Tomi Kuivalainen

Väylätekniikan hyödyntäminen henkilöauton korin sähköjärjestelmän modernisoinnissa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ajoneuvotekniikka

Insinöörityö

29.3.2022

Tiivistelmä

Tekijä:	Tomi Kuivalainen
Otsikko:	Väylätekniikan hyödyntäminen henkilöauton korin sähköjärjestelmän modernisoinnissa.
Sivumäärä:	26 sivua + 2 liitettä
Aika:	29.3.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Ajoneuvotekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Autosähkötekniikka
Ohjaajat:	Lehtori Pasi Kovanen

Tämän insinöörityön tavoitteena oli modernisoida vanhan henkilöauton korisähköjärjestelmä väylätekniikkaa käyttäen, tutkia häviöiden vähenemistä ja tehdä järjestelmästä modulaarinen. Auto, johon järjestelmä rakennettiin, on 30 vuotta vanha.

Vanhanaikaisessa sähköjärjestelmässä oli käytetty paljon yksinkertaisia ratkaisuja, jotka eivät kuitenkaan välttämättä ole mahdollisimman järkeviä. Jos alkuperäiseen järjestelmään haluttaisiin lisätä toimilaitteita tai varusteita, se olisi huomattavasti työläämpää kuin nykyaikaisessa ajoneuvossa.

Työssä korvattiin lähes koko korin sähköjärjestelmä CAN-väylällä ja ohjainlaitteilla. Lisäksi auton sisustan toimintoja varten rakennettiin ohjainlaite Arduino Due -mikrokontrolleria käyttäen sekä ovien toimintoja varten Arduino Nano Every -mikrokontrollereja käyttäen.

Työn tuloksena saatiin nykyaikainen ja toimiva henkilöauton korin sähköjärjestelmä, joka on huomattavasti energiatehokkaampi ja toimintavarmempi kuin alkuperäinen järjestelmä oli. Lisäksi johtosarjojen kokonaismassa puolittui alkuperäiseen verrattuna. Uudessa sähköjärjestelmässä on vähemmän kuluvia osia ja siihen on helppo lisätä komponentteja.

Avainsanat: CAN-väylä, korisähköjärjestelmä, Arduino, johtosarja, tehohäviöt

Abstract

Author:	Tomi Kuivalainen
Title:	Utilizing Bus Technology in the Body Electrical System of a Passenger Car.
Number of Pages:	26 pages + 2 appendices
Date:	29 March 2022
Degree:	Bachelor of Engineering
Degree Programme:	Automotive Engineering
Professional Major:	Automotive Electronics Engineering
Supervisors:	Pasi Kovanen, Senior Lecturer

The aim of this Bachelor's thesis was to modernize the body electrical system of an old passenger car using bus technology, to examine decrease of losses and to make the system modular. The vehicle that the system was built on, is 30 years old.

Many simple solutions had been used in the original outdated electrical system, and they were not necessarily very reasonable. If any devices or accessories had been added to the original system, it would be considerably more laborious than it would be in a modern vehicle.

In this thesis almost the whole vehicle body electrical system was replaced with CAN-bus and control modules. In addition, a control module for the vehicle interior functions was built using an Arduino Due microcontroller and control modules for the door functions using Arduino Nano Every microcontrollers.

The result of this thesis was a modern and functional body electrical system for a passenger car that is significantly more energy efficient and reliable than the original system. Additionally, the total weight of the wiring harnesses was halved compared to the original wiring harness. The new electrical system has fewer wear parts and it is easy to add more components to it.

Keywords: CAN bus, body electrical system, Arduino, wiring harness, power losses

Sisällys

Lyhenteet

CAN: *Controller Area Network.* Ajoneuvoissa ja automaatiojärjestelmissä paljon käytetty tiedonsiirtoväylä.

NRZ: *Non-Return-to-Zero.* Binäärikoodaus ilman perustilaa.

RTR: *Remote transmission request.* Viestikehyksen tyypin määrittelevä bitti.

SRR: *Substitute remote request.* RTR-bitin korvike CAN 2.0B -viestikehyksessä.

IDE: *Identifier Extension bit.* Viestikehyksen tunnisteen pituuden määrittävä bitti.

CRC: *Cyclic Redundancy Check.* Tarkistusalgoritmi.

GPIO: *General Purpose Input/Output.* Yleinen tulo-lähtöliitäntä.

PWM: *Pulse Width Modulation.* Pulssinleveysmodulaatio.

MOSFET: *Metal-Oxide-Semiconductor Field-Effect-Transistor.* Metallioksidi-puolijohdekanavatransistori.

LIN: *Local Interconnect Network.* Ajoneuvoissa käytetty yksinkertainen tiedonsiirtoväylä.

1 Johdanto

Ajoneuvojen sähköjärjestelmät ja varusteet kehittyvät koko ajan. Koska nykyään autoissa on todella paljon sähköllä toimivia laitteita, niiden väliseen kommunikointiin käytetään monesti väylätekniikkaa. Yleisimpiä ajoneuvoissa käytettyjä väyliä ovat CAN, LIN, FlexRay ja MOST.

Väylätekniikkaa hyödyntämällä johtosarjoista saadaan yksinkertaisempia ja kaikista järjestelmistä modulaarisempia. Ilman väylätekniikan hyödyntämistä tarvittaisiin yhden tiedon siirtämiseen jokaiselle sitä tarvitsevalle ohjainlaitteelle oma johdin ja joissakin tapauksissa myös enemmän elektroniikkaa. Väylätekniikalla voidaan myös minimoida häiriöiden syntyminen signaaleihin pitkillä matkoilla.

Tämän insinöörityön tavoitteena oli hyödyntää väyläteknologiaa 30 vuotta vanhan auton korisähköjen modernisoinnissa. Työssä korvataan Mercedes-Benz 190E -mallisen henkilöauton lähes koko korin sähköjärjestelmä CAN-väylällä ja tutkitaan muutoksen vaikutusta häviöihin ja järjestelmän modulaarisuuteen. ABS-jarrujärjestelmä ja turvavöiden esikiristimet jätetään turvallisuussyistä työn ulkopuolelle.

Autoon on tarkoitus asentaa myöhemmin myös uusi moottorinohjausjärjestelmä, joten työssä pyrittiin luomaan edellytykset myös sen helpolle asennettavuudelle.

2 CAN-väylä

CAN-väylä on ensimmäinen massatuotannolla valmistetuissa ajoneuvoissa käytetty väylätyyppi. Nykyään lähes kaikki ajoneuvon ohjainlaitteiden välinen kommunikaatio tapahtuu CAN-väylän välityksellä. CAN-väylää käytetään ajoneuvojen lisäksi myös automaatiojärjestelmissä. CAN-väylässä käytetään NRZ-koodausta. Nykyinen ja tässä työssä hyödynnetty versio CAN-väylästä on CAN 2.0. [1, s. 92.]

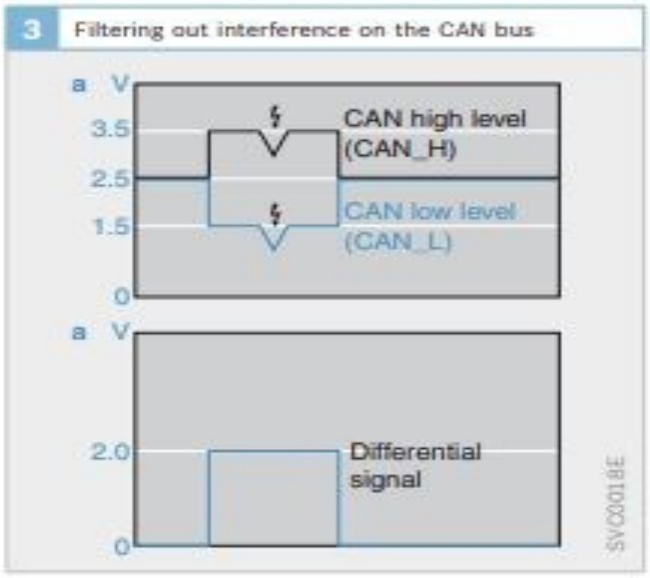
2.1 Standardit ja tiedonsiirtonopeudet

Ajoneuvojen CAN-väylissä käytetään kahta eri standardia riippuen käyttökohteen vaatimuksista. Hidasta CAN-väylää eli CAN-B-väylää käytetään kohteissa, joissa ei tarvita korkeaa tiedonsiirtonopeutta ja kohteissa, joissa tarvitaan parempaa vikasietoisuutta. Hitaan CAN-väylän yleisiä käyttökohteita ovat esim. auton sisustan varusteet ja valaisimet. Nopeaa CAN-väylää eli CAN-C-väylää taas käytetään suurta kaistanleveyttä vaativissa kohteissa, esimerkiksi moottorinohjauksessa ja turvajärjestelmissä. Hitaan CAN-väylän tiedonsiirtonopeus on 5–125 kbit/s ja nopean CAN-väylän tiedonsiirtonopeus on 125–1000 kbit/s. Tässä työssä käytetään ainoastaan nopeaa CAN-väylää komponenttien yhteensopivuuden takaamiseksi. [1, s. 92.]

2.2 Fyysinen kerros

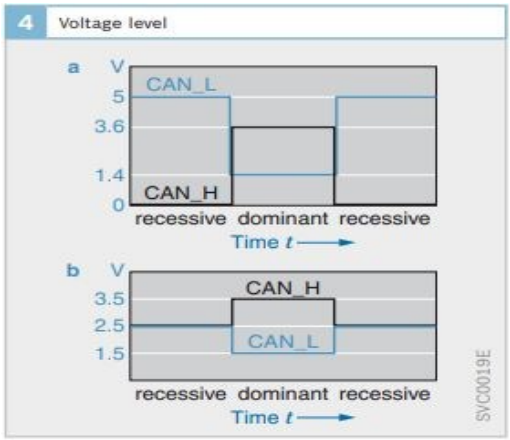
CAN-väylässä käytetään siirtovälineenä kierrettyä parikaapelia, jossa on CAN_H- ja CAN_L-johtimet. Kierretyssä parikaapelissa on etuna häiriösuojaus, sillä johtimiin mahdollisesti indusoituva häiriöjännite on samansuuruinen molemmissa johtimissa eikä muuta niiden välistä jännite-eroa (kuva 1). [1, s. 94.]

Nopeassa CAN-väylässä väyläkaapelin molemmissa päädyissä on 120 ohmin päätevastus. Päätevastuksilla estetään signaalin mahdollinen heijastuminen johtimissa. Jokaisessa CAN-väylään liitetyssä ohjainlaitteessa on oman logiikkansa lisäksi CAN-väyläohjain sekä lähetin-vastaanotinmoduuli. [1, s. 95.]



Kuva 1. Häiriön vaikutus signaaliin [1, s. 94].

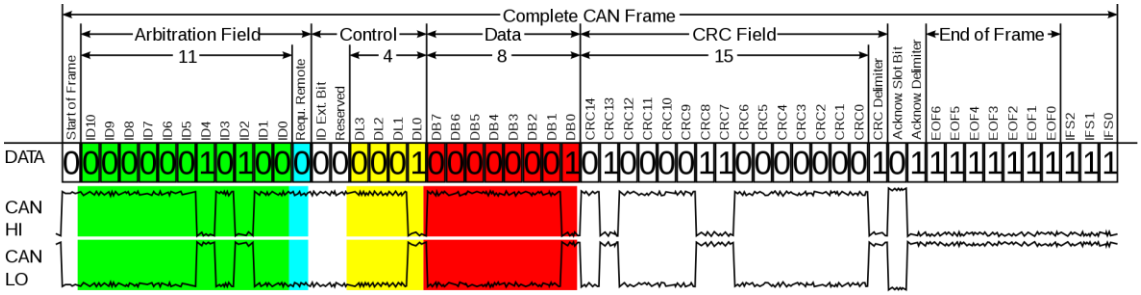
Kuvasta 2 nähdään, että hitaan ja nopean CAN-väylän jännitetasot ovat täysin erilaiset keskenään. Hitaassa CAN-väylässä CAN_L-johtimen jännite on 5 volttia ja CAN_H-johtimen jännite on 0 volttia väistyvässä tilassa (looginen 1). Hallitsevassa tilassa (looginen 0) taas CAN_H-johtimen jännite on 3,6 volttia ja CAN_L-johtimen jännite on 1,4 volttia. Nopeassa CAN-väylässä molempien johtimien jännite on 2,5 volttia väistyvässä tilassa ja hallitsevassa tilassa CAN_H-johtimen jännite on 3,5 volttia ja CAN_L-johtimen jännite on 3,5 volttia. [1, s. 95.]



Kuva 2. CAN-väylän jännitetasot, a = CAN-B, b = CAN-C [1, s. 95].

2.3 Viestikehys ja väylän haltuunotto

CAN-väylä on moni-isäntäinen väylä, joten mikä tahansa siihen liitetty ohjainlaite voi välittää tietoa väylälle väylän ollessa vapaa. Ollessaan vapaana väylä on väistyvässä tilassa. Väylän haltuunotossa korkeimman prioriteetin viestikehys pääsee väylälle. Kuvassa 3 on nopean CAN-väylän viestikehyksen rakenne ja sen bittejä vastaavat jännitetasot. [1, s. 97; 2.]



Kuva 3. Viestikehyksen rakenne [3].

2.3.1 Viestikehyksen aloitus ja haltuunottokenttä

Uusi viestikehys alkaa aina hallitsevalla bitillä, jonka jälkeen tulee viestikehyksen tunniste. Tunnisteen pituus on joko 11 bittiä (CAN 2.0A) tai 29 bittiä (CAN 2.0B). Viestikehyksen tunniste määrittää myös sen prioriteetin väylällä. Mitä pienempi tunnisteen arvo viestikehyksellä on, sitä korkeampi prioriteetti sillä on. [1, s. 98.]

CAN 2.0A -viestikehyksessä 11-bittisen tunnistekentän jälkeen tulee RTR-bitti, jolla määritetään, onko viestikehys data- vai pyyntökehys. RTR-bitti on hallitseva datakehyksissä ja väistyvä pyyntökehyksissä. Haltuunottotilanteessa datakehys menee aina pyyntökehyksen edelle. CAN 2.0B -viestikehyksessä RTR-bitin tilalla on SRR-bitti, joka on aina väistyvä. Tällä varmistetaan, että CAN

2.0A -viestikehyksellä on aina korkeampi prioriteetti kuin CAN 2.0B -viestikehyksellä. [1, s. 99.]

2.3.2 Ohjauskenttä

RTR- tai SRR-bitin jälkeen tulee IDE-bitti, jolla määritetään tunnisteen pituus. IDE-bitti on CAN 2.0A -viestikehyksessä aina hallitseva ja CAN 2.0B -viestikehyksessä väistyvä. CAN 2.0B -viestikehyksessä IDE-bitin jälkeen tulee loput 18 bittiä tunnisteesta ja RTR-bitti.

CAN 2.0A -viestikehyksessä IDE-bitin jälkeinen bitti on varattu tulevia muutoksia varten ja se on aina hallitseva. CAN 2.0B -viestikehyksessä IDE-bitti kuuluu haltuunottokenttään, jolloin varattuja bittejä on kaksi kappaletta. Seuraavat 4 bittiä kertovat viestin pituuden tavuina (0–8 tavua). [1, s. 99.]

2.3.3 Datakenttä, CRC-kenttä ja viestin lopetus

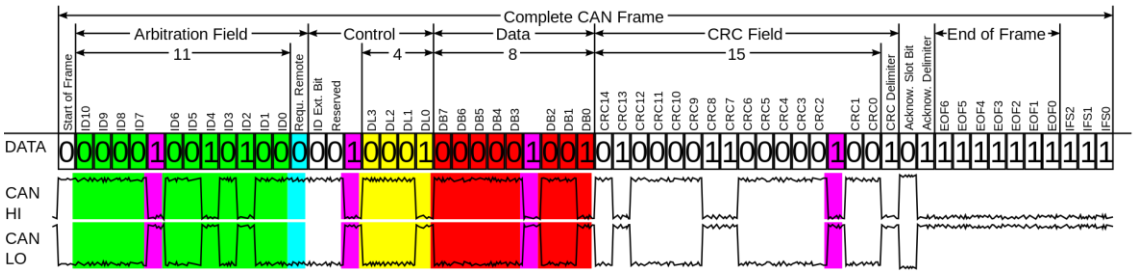
Heti viestin pituuden jälkeen tulee itse viestin sisältö. Viestin maksimipituus on 8 tavua eli 64 bittiä. Datakentän jälkeen tulee CRC-kenttä, joka sisältää viestin 15-bittisen tarkisteavaimen. Tarkisteavaimella voidaan varmistaa, että viesti on oikea, eikä ole muuttunut häiriön seurauksena. CRC-kentän 16. bitti on aina väistyvä ja lopettaa tarkisteavaimen.

CRC-kentän jälkeen lähettävä ohjainlaite lähettää vahvistusbitin väistyvänä ja vastaanottava ohjainlaite korvaa sen hallitsevalla bitillä, jos viesti on vastaanotettu ilman virheitä. Vahvistusbitin jälkeen tulee väistyvä bitti, jolla lopetetaan vahvistuskenttä. Viestikehys lopetetaan lähettämällä seitsemän väistyvää bittiä peräkkäin. [1, s. 99–100.]

2.3.4 Bit stuffing

CAN-väylällä tulee usein tilanteita, joissa tulee monta samanarvoista bittiä peräkkäin. Koska CAN-väylässä käytetään NRZ-koodausta, ei voida tietää, onko

kyseessä virhe vai normaali tilanne. Ongelman välttämiseksi käytetään bit stuffing -menetelmää, jossa aina viiden peräkkäisen samanarvoisen bitin jälkeen syötetään väylälle yksittäinen eriarvoinen bitti. Kuvassa 4 näkyy stuff-bitit viestikehyksessä. [2]



Kuva 4. Viestikehys stuff-biteillä [3].

2.4 Virheenhallinta

Jos vastaanottava laite havaitsee virheen viestissä, se lähettää virhekehyksen heti väylälle. Aktiivinen virhekehys koostuu hallitsevista biteistä ja sillä keskeytetään virheellinen viesti, jotta muut vastaanottajat eivät hyödyntäisi sitä. [2]

CAN-väylässä käytetään seuraavia virheentunnistusmenetelmiä:

- **Väylämonitorointi.** Lähettävä ohjainlaite vertaa lähetettyä signaalia signaaliin, joka väylällä liikkuu fyysisesti.
- **CRC.** Ohjainlaitteet vertaavat CRC-tarkisteavainta viestiin
- **Kehyksen tarkistus.** Ohjainlaitteet tarkistavat viestikehyksen rakenteen.
- **Bit Stuffing.** Stuff-bittien virheet CRC-kentän loppuun asti.
- **Vahvistus.** Vahvistusbitin tilan muutos.
- **Virheilmoitus.** Koska viestin rakenne on virheilmoituksen jälkeen väärä, muut vastaanottavat laitteet ”pitkittävät” sitä.

Väyläkommunikaatiossa tapahtuvia mahdollisia virheitä ovat:

- **Bittivirhe.** Yhden tai useamman bitin arvo on muuttunut häiriön seurauksena.

- **Bit Stuffing -virhe.** Viestikehyksessä on enemmän kuin viisi samaa peräkkäistä bittiä.
- **Vahvistusvirhe.** Vastaanottava laite ei muuta vahvistusbittiä hallitsevaksi tai lähettävä laite on ainoa väylällä.
- **CRC-virhe.** CRC-tarkistusavain ei täsmää.

Jokaisella väylään liitetyllä laitteella on virhelaskuri sekä lähetetylle että vastaanotetulle datalle. Virhetilanteessa virheen tunnistanut laite lähettää väylälle aktiivisen virheilmoituksen. Kun laskuri saavuttaa ”varoitusrajan” (96), useimmat laitteet reagoivat toistuviin virheisiin. Kun virhelaskurin arvo on pienempi kuin 128, laite on aktiivisessa virhetilassa.

Laskurin saavuttaessa arvon 128 laite siirtyy passiiviseen virhetilaan. Passiivisessa virhetilassa laite yrittää kommunikoida normaalisti, mutta lähettää passiivisia virheilmoituksia. Passiivinen virheilmoitus ei keskeytä viestikehystä kuten aktiivinen virheilmoitus.

Kun laskuri saavuttaa arvon 255, laite menee ”bus off” -tilaan. Tässä tilassa laite kytkeytyy kokonaan pois väylältä. Laite voi palata aktiiviseen virhetilaan, jos se lukee väylältä 128 virheetöntä 11:n väistyvän bitin sarjaa. Kyseinen sarja syntyy viestikehyksen virheettömästä lopetuksesta. [2.]

3 Häviöiden tarkastelu ennen muutoksia

3.1 Jännitehäviö

Kirchhoffin jännitelain mukaan virtapiirissä olevien jännitteiden summa on aina 0. Tämä tarkoittaa sitä, että virtapiirissä olevien toimilaitteiden ja häviöiden yli oleva jännite on sama kuin piirin käyttöjännite. Jokaisella johtimella ja liitoksella on tietty resistanssi, jonka aiheuttama jännitehäviö voidaan laskea Ohmin lain mukaan virran funktiona kaavalla 1

$$U = R * I, \tag{1}$$

jossa U on jännitehäviö, R on resistanssi ja I on virta. Jännitehäviö aiheuttaa aina myös tehohäviötä ja sen myötä liitosten ja johtimien lämpenemistä ja ennenaikaista kulumista. Tehohäviö voidaan laskea myös Ohmin lakia käyttäen kaavalla 2

$$P = U * I, \tag{2}$$

jossa P on tehohäviö, U on jännitehäviö ja I on virta. Johtimen resistanssiin ja sen aiheuttamaan jännitehäviöön vaikuttaa sen pituus, materiaali ja poikkipinta-ala. Johtimen resistanssi voidaan laskea kaavalla 3

$$R = \rho \frac{l}{A}, \tag{3}$$

jossa R on johtimen resistanssi, ρ on materiaalin ominaisvastus, l on pituus ja A on poikkipinta-ala.

Ominaisvastus riippuu käytetystä materiaalista ja sen lämpötilasta. [4, s. 420.] Taulukossa 1 on yleisimpien käytettyjen johdinmateriaalien ominaisvastusarvoja.

Taulukko 1. Johdinmateriaalien ominaisvastusarvot. [4, s. 420.]

Materiaali	**Ominaisvastus (20 °C)**	**Lämpötilakerroin**
Kupari	0,0168 * 10^{-6} Ωm	3,9 * 10^{-3} / K
Alumiini	0,027 * 10^{-6} Ωm	4,3 * 10^{-3} / K
Tina	0,110 * 10^{-6} Ωm	4,6 * 10^{-3} / K
Kulta	0,022 * 10^{-6} Ωm	4,0 * 10^{-3} / K
Hopea	0,16 * 10^{-6} Ωm	3,8 * 10^{-3} / K

Autossa käytetyt johtimet on pääosin valmistettu kuparista. Häviöt kasvavat myös johtimien ja liittimien korroosion ja kulumisen seurauksena. Tämä vääristää hieman mittaustuloksia, sillä alkuperäisellä järjestelmällä on ikää noin 30 vuotta. Alkuperäisessä järjestelmässä on käytetty paljon yksinkertaisia ratkaisuja, joissa on potentiaalinen kehittämisen mahdollisuus nykytekniikkaa käyttäen.

3.2 Mittaus

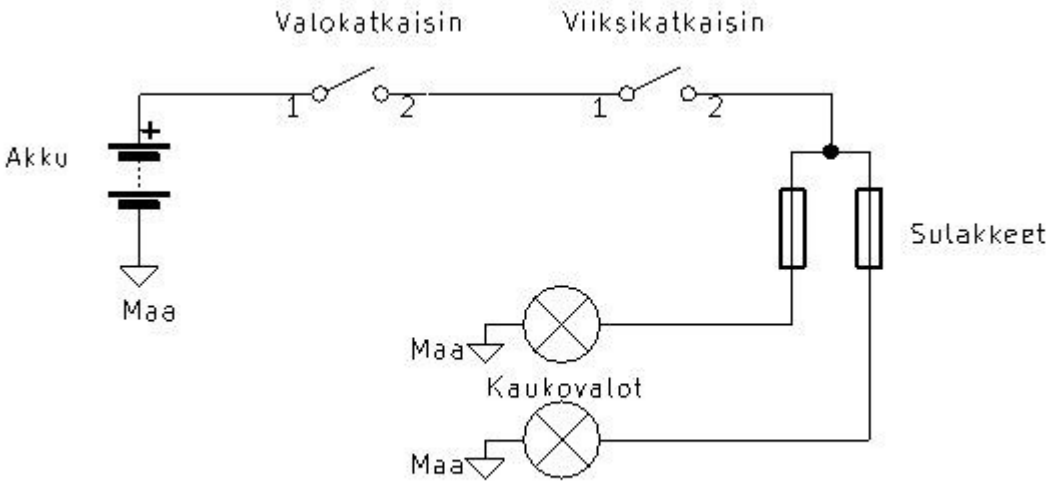
Alkuperäisestä järjestelmästä mitattiin jännitehäviöitä akulta erinäisille laitteille ennen muutoksia. Taulukossa 2 on mittaustuloksia laitteilta, joiden kytkentä muutetaan.

Taulukko 2. Mittaustuloksia.

Toimilaite	**Virta (A)**	**Jännitehäviö (mV)**			**Tehohäviö (W)**
Vasen lähivalo	4,73	Akku–valokatkaisin 184	Valokatkaisin–sulake 80	Sulake–valo 385	3,07
Oikea lähivalo	4,75	Akku–valokatkaisin 184	Valokatkaisin–sulake 120	Sulake–valo 486	3,75
Pyyhin	10	Akku–sulake 300	Sulake–viiksikatkaisin 97	Viiksi–moottori 503	9,00
Vasen kaukovalo	5,15	Akku–viiksikatkaisin 377	Viiksikatkaisin–sulake 112	Sulake–valo 340	4,27
Oikea kaukovalo	5,11	Akku–viiksikatkaisin 377	Viiksikatkaisin–sulake 100	Sulake–valo 540	5,20
Jarruvalo	1,85	Akku–sulake 130	Sulake–kytkin 55	Kytkin–valo 500	1,30
Peruutusvalot	1,5	Akku–sulake 80	Sulake–kytkin 120	Kytkin–valo 360	0,84
Takalasinlämmitin	5,5	Akku–sulake 91	Sulake–rele 200	Rele–vastus 390	3,75

3.3 Mittaustulosten tarkastelu

Mittaustuloksista nähdään, että tehohäviöt ovat verrattain suuria. Tämä johtuu siitä, että järjestelmässä on paljon yksinkertaisia kytkentöjä, joita on ollut pakko käyttää ilman mikrokontrollereita. Hyvä esimerkki tästä on kaukovalojen kytkentä, jossa virta kiertää akulta valokatkaisimen ja viiksikatkaisimen kautta valaisimelle (kuva 5).



Kuva 5. Kaukovalojen alkuperäinen kytkentä.

Kyseisistä kytkennöistä johtuen järjestelmässä on myös paljon pitkiä johdotuksia suurta virtaa käyttävissä kohteissa, joita voidaan vähentää huomattavasti väylätekniikkaa käyttäen. Johtimien pituuden vaikutuksen huomaa mittaustuloksissa vertaamalla vasemman ja oikean puolen valojen jännitehäviöitä sulakkeen jälkeen keskenään. Väylätekniikkaa soveltamalla saadaan vähennettyä johtimien pituuksia ja siten myös niiden poikkipinta-alaa.

4 Työhön käytetyt laitteet

4.1 Arduino Due

Työssä rakennetun ohjainlaiteen perustaksi valittiin Arduino Due -mikrokontrollerialusta, sillä siinä on riittävästi GPIO-liitäntöjä ja kaksi sisäänrakennettua CAN-väyläkontrolleria. Arduino Duessa käytetään 32-bittistä Atmelin SAM3X8E-mikrokontrolleria. Mikrokontrollerissa on 54 digitaalista GPIO-liitäntää, 12 analogista tuloa ja 2 analogista lähtöä. Digitaalisista liitännöistä 12 tukee PWM-ohjausta. [5.]

GPIO-liitännöissä käytetään jännitetasona 3,3 volttia, joten osassa tuloista on auton oman sähköjärjestelmän 12 voltin jännite muutettava sopivalle tasolle. Lisäksi tarvitaan erillinen lähetin-vastaanotinmoduuli, sillä sitä ei ole mikrokontrollerissa valmiina. [5.]

Lähetin-vastaanotinmoduulina käytettiin Microchipin valmistamaa MCP2562-moduulia. MCP2562 sisältää jännitetasomuuntimen I/O-liitännöille, joten sitä voidaan käyttää 5 voltin käyttöjännitteellä 3,3 voltin mikrokontrollerissa. 5 voltin käyttöjännite on pakollinen, jotta CAN-väylän jännitetasot ovat yhteensopivat muiden laitteiden kanssa. [6.]

4.2 Arduino Nano Every (ovet)

Jotta GPIO-liitännät riittäisivät kaikille toiminnoille, etuovien toimintojen ohjaamiseen tarvittiin erilliset ohjainlaitteet. Näihin toimintoihin kuuluu oven lukon asennon tunnistaminen, sähköikkunoiden ohjaus ja oikealla puolella peruutuspeilin säätömoottoreiden ohjaus.

Nämä ohjainlaitteet perustuvat Arduino Nano Every -mikrokontrollerialustaan. Arduino Nano Every perustuu Atmelin ATMega4809-mikrokontrolleriin. [7.]

Suurien virtojen ohjaukset toteutettiin releitä ja transistoreja hyödyntäen. CAN-väyläkommunikaatioon käytettiin väyläohjaimena Microchipin valmistamaa MCP2515-ohjainta ja lähetin-vastaanotinmoduulina saman yrityksen valmistamaa MCP2562-moduulia.

4.3 Ecumaster PMU16

Pääohjainlaitteina auton etu- ja takaosien laitteille käytettiin Ecumasterin valmistamia PMU16-virranhallintayksiköitä. PMU16 sisältää 16 kappaletta 10-bittisiä analogisia tuloja (0–5 V) ja 16 kappaletta lähtöjä. Laitteella pystyy myös hyödyntämään tuloja CAN-väylältä. Laitteesta löytyy valmiit funktiot suuntavilkkujen ja tuulilasinpyyhkimen ohjaamiseen. [8.]

5 Uuden järjestelmän fyysinen rakentaminen

5.1 Sulakkeet ja releet

Ohjainlaitteiden päävirtakaapeleiden suojaamiseksi akun positiiviseen napaan hankittiin jakoyksikkö, jossa on kolme sulakoitua ja yksi sulakoimaton lähtö. Sulakoimaton lähtö kytkettiin käynnistinmoottorille, sillä se tarvitsee käynnistystilanteessa hetkellisesti suuren virran. Sulakoiduista lähdöistä kaksi käytettiin PMU-yksiköille ja yksi sulakerasialle.

Ohjaamattomien laitteiden käyttöjännitteet kytkettiin muokatun alkuperäisen sulakerasian kautta. Näitä laitteita ovat esimerkiksi sisustan valaisimet, savukkeensytytin ja radio. Sulakerasia muokattiin ”busbar”-tyyppiseksi, sillä sen sisäiset kytkennät korvattiin älykkäällä ohjauksella. Sulakerasiaan rakennettiin releohjatut virtakiskot virtalukon eri asentoja varten. Alkuperäinen sulakerasia painoi 1,87 kg ja uusi muokattu sulakerasia 0,765 kg.

5.2 Johtosarjat

Työtä varten hankittiin vastaavasta autosta puretut käytetyt johtosarjat, joita muokattiin yksinkertaisemmiksi. Autossa käytettyjä liittimiä ei saatu hankittua uutena, mutta kaikkia alkuperäisiä liittimiä pystyttiin hyödyntämään työssä, sillä johtimet on juotettu niihin kiinni.

Alkuperäisissä johtosarjoissa oli paljon toimilaitteiden välisiä kytkentöjä sekä laitteiden käyttöjännitteiden ja maadoitusten edestakaisia johdotuksia. Lisäksi takavalojen ja kattoluukun moottorin johtosarjat olivat koko auton pituisia. Kyseiset kytkennät korvattiin CAN-väylällä ja käyttöjännitteiden yksinkertaisemmilla johdotuksilla. Maadoituspisteitä myös lisättiin, jolloin niihin saatiin huomattavasti lyhyemmät johdotukset.

Päävirtakaapeleina käytettiin auton etuosan ja sulakerasian kaapeleina 16 mm^2:n kuparikaapelia ja takaosan kaapelina 25 mm^2:n kuparikaapelia. Kaapeleiden mitoituksessa on huomioitu mahdolliset tulevat lisävarusteasennukset.

Diagnostiikka- ja ohjelmointiliitäntään valittiin liittimeksi SAE J1962 -standardin mukainen liitin, jota käytetään OBD2-järjestelmässä nykyaikaisissa ajoneuvoissa. Liittimeen kytkettiin jatkuva 12 voltin jännite, maadoitus sekä CAN-väyläjohtimet.

Alkuperäisillä johtosarjoilla oli massaa yhteensä 10 kg. Uudet johtosarjat ja ohjainlaitteet painavat yhteensä 6 kg.

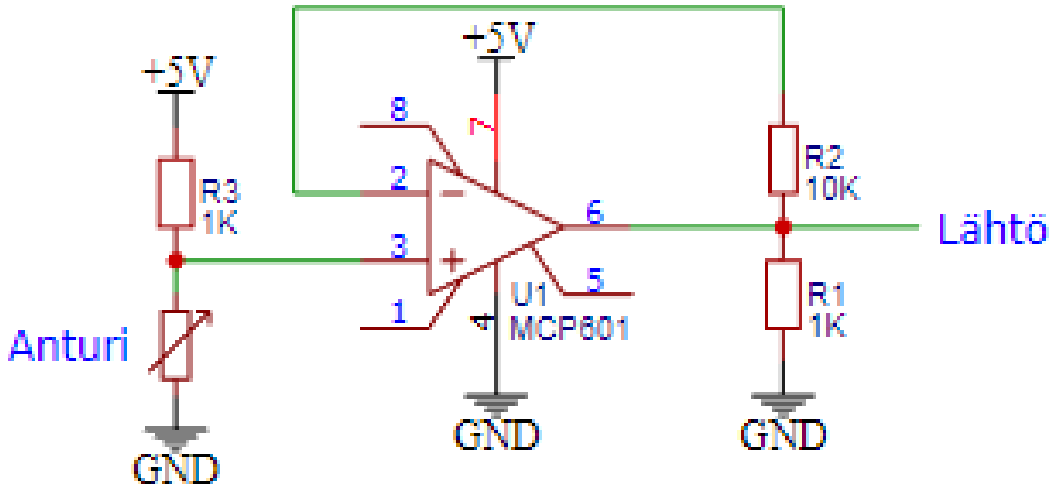
5.3 PMU-ohjainlaitteiden toiminnot

PMU-ohjainlaitteisiin kytkettiin auton kaikki ulkopuoliset valot, äänimerkinantolaite, tuulilasinpyyhkijä, tuulilasinpesurin pumppu, lämmityslaitteen puhallin, kattoluukku ja anturit, jotka sijaitsevat moottoritilassa tai auton takapäässä. Näihin antureihin kuuluu tuulilasin pesurin, jarrunesteen, jäähdytinnesteen ja polttoaineen tasoanturit sekä jarrupalojen kuluneisuustunnistimet.

Polttoaineen tasoanturille rakennettiin muunnospiiri, jonka tehtävä on muuntaa yksinapainen resistanssiin perustuva anturi kaksinapaiseksi jännitetasoon perustuvaksi anturiksi. Koska alkuperäinen polttoainemittari on perinteinen analoginen mittari, tasoanturin impedanssi on todella matala (0–100 ohmia). Jotta tällä impedanssilla saataisiin järkevästi mitattavissa oleva jännitetaso, se kuluttaisi paljon tehoa. Muunnospiiri perustuu ei-invertoivaan operaatiovahvistinpiiriin. Anturi kytketään sarjaan 1 kΩ:n vastuksen kanssa, jolloin muodostuu jännitteenjakopiiri, jonka jännitetaso on 0–0,45 volttia 5 voltin käyttöjännitteellä. Tämä jännitetaso vahvistetaan takaisin 0–5 voltin jännitetasolle. Vahvistinpiirin lähtöjännite voidaan laskea kaavalla 4:

$$U_{OUT} = \left(1 + \frac{R_2}{R_1}\right) * U_{IN} , \quad (4)$$

jossa U_{OUT} on lähtöjännite, U_{IN} on tulojännite, R_2 on lähdön ja invertoivan tulon välinen vastus ja R_1 on invertoivan tulon ja maadoituksen välinen vastus. Muunnospiirin kytkentä näkyy kuvassa 6.



Kuva 6. Operaatiovahvistinpiiri.

5.4 Rakennetut ohjainlaitteet

Koska työssä käytetyillä mikrokontrollerialustoilla ei voi suoraan ohjata toimilaitteita, niiden ympärille teetettiin piirilevyt. Piirilevyille asennettiin lähtöjen ohjaamiseen, tiettyjen tulojen jännitetasomuunnoksiin tarvittavat komponentit ja väylälähkommunikaatioon tarvittavat komponentit.

5.4.1 Lähdöt

Lähdöt, joissa ohjataan maadoitusta, on toteutettu käyttäen IRLL2705-mallisia N-kanavaisia avauskanava-MOSFET-transistoreja. Transistori maadoittaa kuorman, kun sen hilan ja lähteen välinen jännite ylittää raja-arvon, joka on kyseisissä transistoreissa korkeintaan 2 volttia.

Transistorin lähes olemattoman tuloimpedanssin takia hilalle tarvitaan myös alasvetovastus estämään ylimääräisiä kytkeytymisiä. Lisäksi kontrollerin ja hilan välillä on vastus, jolla rajoitetaan transistorin tulokapasitanssin varaamiseen vaadittavaa virtaa. Tulokapasitanssi on huomioitava varsinkin PWM-

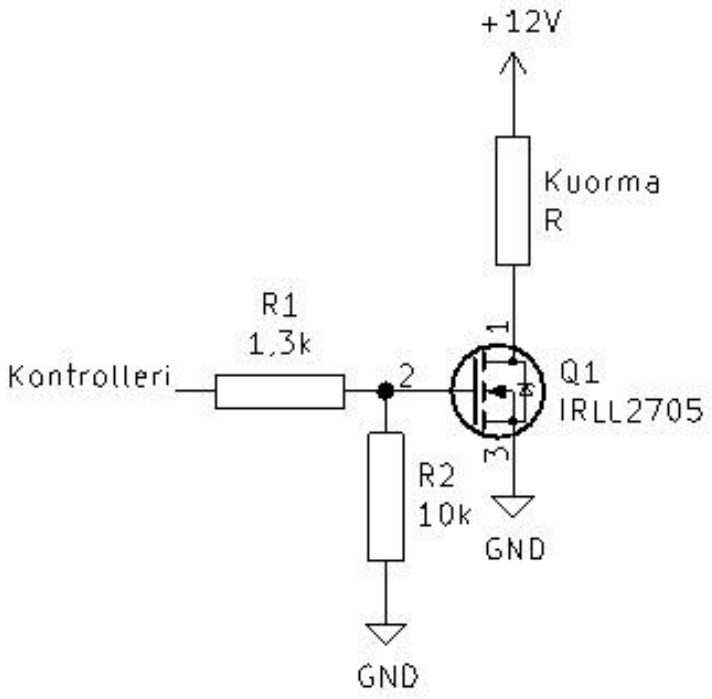
ohjauksissa, sillä transistoria kytketään päälle ja pois jatkuvasti. Kyseinen kytkentä näkyy kuvassa 7. Vastuksen minimikoko on laskettu kaavalla 5:

$$\frac{E-U_0}{I_{max}} = \frac{3{,}3\ V-0\ V}{3\ mA} = 1{,}1\ k\Omega, \quad (5)$$

jossa E on mikrokontrollerin syöttämä hilajännite, U_0 on kapasitanssin varautumisen alkujännite ja I_{max} on mikrokontrollerin GPIO-liitännän maksimivirta. Vastukseksi valittiin 1,3 kΩ:n vastus hyvän saatavuuden vuoksi. PWM-ohjauksessa on lisäksi huomioitava kapasitanssin ja hilavastuksen vaikutus varautumisaikaan, joka voidaan määrittää karkeasti kaavalla 6:

$$t = 5 * R_g * C_{iss} = 5 * 1{,}3\ k\Omega * 870\ pF \approx 1{,}1\ \mu s, \quad (6)$$

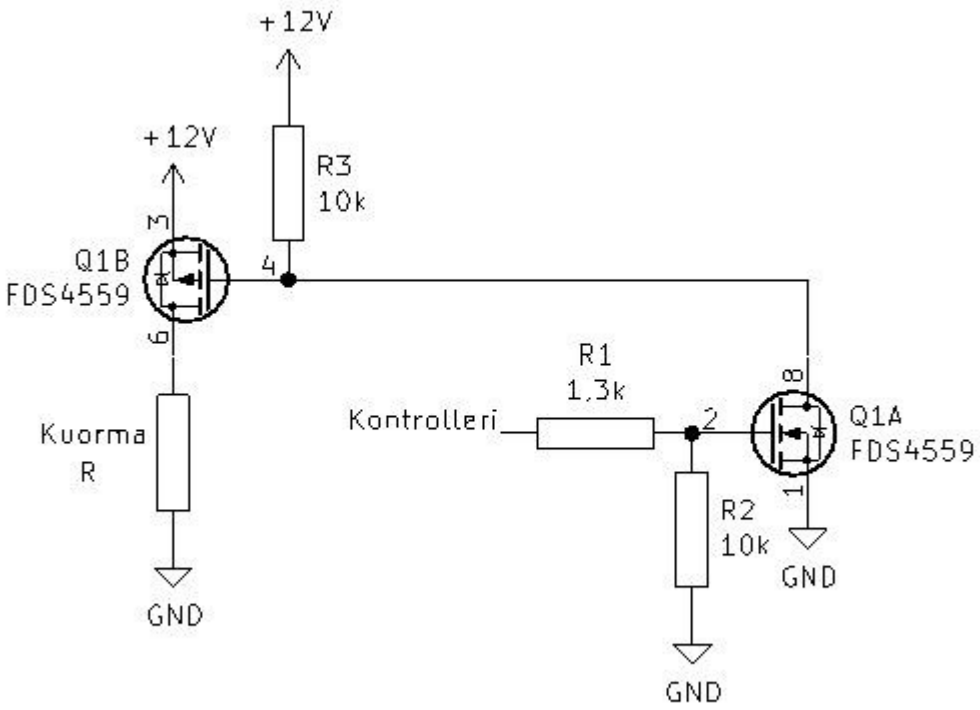
jossa R_g on hilavastuksen resistanssi, C_{iss} on transistorin tulokapasitanssi ja t on latausaika. PWM-ohjauksessa käytetään 1 kHz:n taajuutta, joten jaksonaika on 1 ms. Varautumisaika ei tässä tilanteessa vaikuta käytännössä ohjauksen toimintaan.



Kuva 7. Maadoituksen ohjaus.

Lähdöissä, joissa ohjataan jännitepuolta, käytetään taas P-kanavaisia avauskanava-MOSFET-transistoreja, joita ohjataan N-kanavaisilla MOSFETeillä. Tähän kytkentään valittiin FDS4559 MOSFET-pari. FDS4559 sisältää sekä P-kanavaisen, että N-kanavaisen transistorin.

Transistori johtaa, kun hilan ja lähteen välinen jännite on vähintään -3 V. Hilan ja käyttöjännitteen välillä on ylösvetovastus, jolla transistori pidetään estotilassa, kunnes hila maadoitetaan N-kanavaisella transistorilla. Ylösvetovastus rajoittaa myös maadoittavan transistorin läpi kulkevan virran. Kyseisen N-kanavaisen transistorin hilajännitteen raja-arvo on korkeintaan 3 volttia. Kytkentä näkyy kuvassa 7.

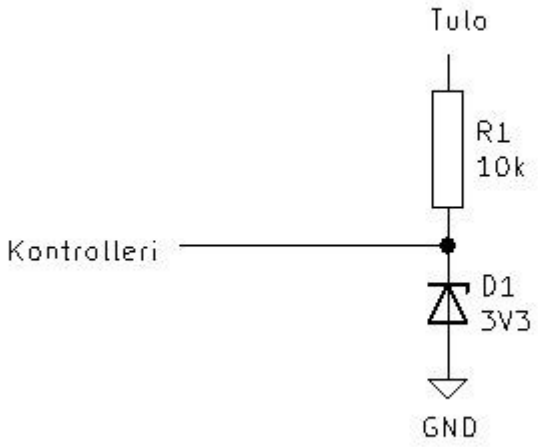


Kuva 8. Jännitepuolen ohjaus

5.4.2 Jännitetasomuunnokset

Tuloissa, joissa joudutaan käyttämään 12 voltin jännitettä, jännite on muunnettava mikrokontrollerille sopivaan korkeintaan 3,3 volttiin. Digitaalisissa tuloissa muunnos toteutettiin vastuksen ja zenerdiodin avulla. Zenerdiodin estosuuntainen kynnysjännite eli zenerjännite rajoittaa tulon jännitteen 3,3 volttiin, ja sen läpi kulkema virta rajoitetaan 10 kΩ:n vastuksella. Lisäksi on varmistettava, että tulo maadoittuu ollessaan pois päältä, sillä zenerdiodi ei johda jännitteen ollessa alle 3,3 volttia, jolloin tuloa ei vedä mikään alas. Jos kytkentä ei maadoitu ollessaan pois päältä, sille tarvitaan alasvetovastus zenerdiodin rinnalle.

Kahdella vastuksella toteutetussa jännitteenjakopiirissä ongelmaksi muodostuu akkujännitteen vaihteluväli. Matalalla jännitteellä mikrokontrolleri ei välttämättä tulkitsisi tulon tilaa oikein. Zenerdiodilla toteutettu kytkentä näkyy kuvassa 8.

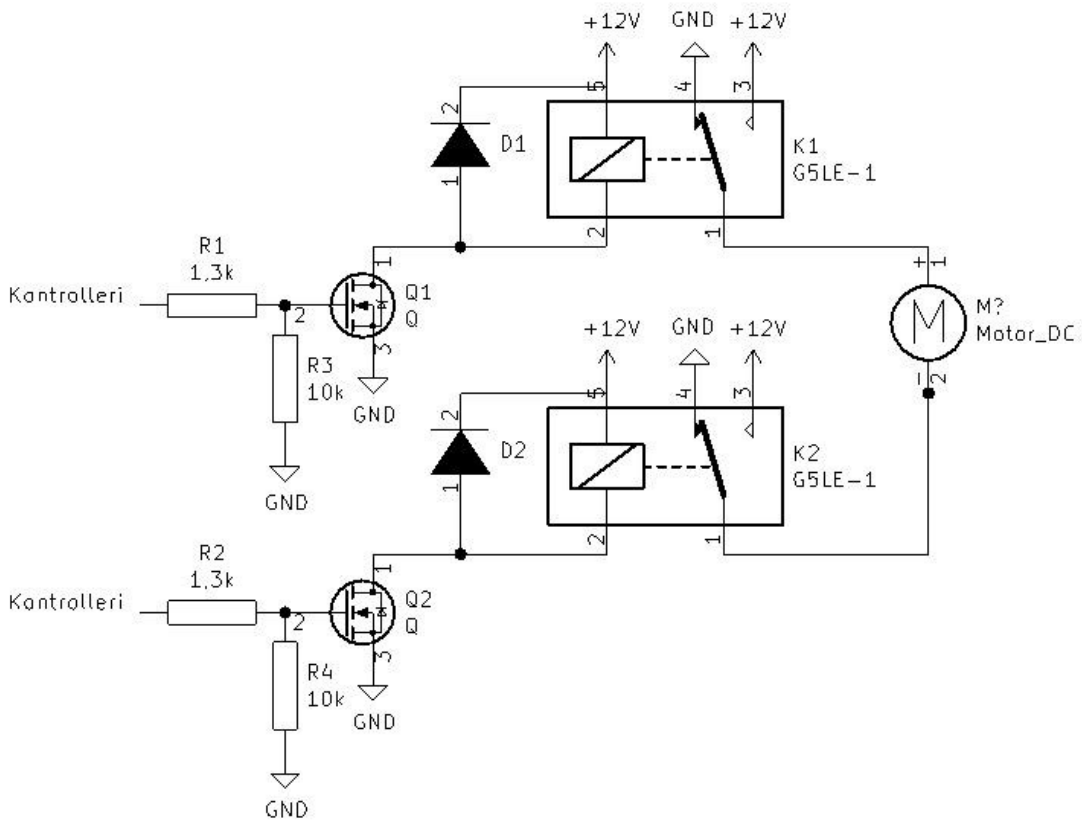


Kuva 9. Jännitetasomuunnos zenerdiodilla.

Analogisten tulojen muunnoksessa taas ei voida käyttää zenerdiodia, sillä mikrokontrollerin mittaama jännite on tällöin maksimissaan 3,3 volttia. Lisäksi tarvittaisiin vertailujännite, joka muuttuu akkujännitteen mukaan. Työssä ei käytetä analogisia tuloja näillä ohjainlaitteilla lainkaan.

5.4.3 Ikkunoiden ja peilien ohjaus

Ikkunoiden ohjaus on toteutettu kahdella releellä ja niiden transistoriohjauksella. Normaalitilassa ikkunannostimen moottorin molemmat navat ovat maadoitettuna. Ohjattaessa relettä, rele kytkee toiseen napaan 12 voltin jännitteen. Releiden rinnalla on diodit, joilla suojataan ohjaustransistori releen kelan induktanssin aiheuttamalta jännitepiikiltä. Kytkentä näkyy kuvassa 9.



Kuva 10. Ikkunan ohjaus.

Säätömoottoreissa ei ole sisäistä virranrajoitusta, joten kytkennässä on myös INA181-virranmittausvahvistin, joka mittaa 5 mΩ:n vastuksen yli olevaa jännitehäviötä ja vahvistaa sen 20-kertaiseksi, jolloin sen voi mitata mikrokontrollerin sisäisellä A/D-muuntimella.

Peilin kääntöä ohjataan samalla tavalla kuin ikkunoita, mutta siinä on kaksi säätömoottoria. Näitä säätömoottoreita ohjataan kolmella releellä.

6 Ohjainlaitteiden ohjelmointi

6.1 PMU-ohjainlaitteet

PMU-laitteet ohjelmoitiin CAN-väylän välityksellä Ecumasterin omaa CAN-väylämoduulia ja ohjelmistoa käyttäen. Lähtökohtaisesti ohjainlaitteiden toiminnot ohjelmoitiin kytkeytymään CAN-väylältä saatavan tiedon mukaan.

Ajovalojen ja kaukovalojen jännite rajoitettiin latausjännitteen funktiona 13,5 volttiin PWM-ohjauksella polttimoiden käyttöiän säästämiseksi. Palaneiden polttimoiden tai virtapiirien vikojen varalle luotiin funktiot ”varavaloille”, eli jos jonkin valon virtapiirin virrankulutus on asetettujen raja-arvojen ulkopuolella, ohjainlaite kytkee toisen valon viallisen tilalle tilapäisesti.

Kattoluukun moottorille kopioidaan kattoluukun kytkimen tila, joka luetaan CAN-väylältä. Antureiden arvot ja lähtöjen tilat lähetetään CAN-väylälle, josta muut ohjainlaitteet lukevat niitä tarvittaessa.

6.2 Arduino Due

Arduino Due -pohjaisen ohjainlaitteen päätehtävä on välittää sisustan katkaisimien tila CAN-väylälle ja ohjata mittariston toimintoja. Ohjelmassa luodaan CAN-viestikehys sekä funktiot mittariston toiminnoille.

Analogisille mittareille luotiin funktiot ja kertoimet, joilla muunnetaan alkuperäisten anturien resistanssit PWM-signaaleiksi. Tässä työssä käytettiin vain polttoainemittaria PWM-ohjauksella.

Lisäksi ohjelmaan luotiin funktiot ikkunoiden ja peilin säädöille.

7 Häviöiden tarkastelu muutosten jälkeen

Järjestelmän rakentamisen jälkeen mitattiin häviöt samoilta laitteilta kuin ennen järjestelmän rakentamista. Taulukossa 3 voidaan nähdä uudet mittaustulokset.

Taulukko 3. Mittaustulokset uudella järjestelmällä.

Toimilaite	**Virta (A)**	**Jännitehäviö (mV)**		**Tehohäviö (W)**
Vasen lähivalo	4,8	Akku–ohjainlaite 7	Ohjainlaite–valo 203	1,00
Oikea lähivalo	4,8	Akku–ohjainlaite 7	Ohjainlaite–valo 129	0,65
Pyyhkijä	12	Akku–ohjainlaite 15	Ohjainlaite–moottori 154	2,03
Vasen kaukovalo	4,5	Akku–ohjainlaite 10	Ohjainlaite–valo 195	0,92
Oikea kaukovalo	4,5	Akku–ohjainlaite 10	Ohjainlaite–valo 171	0,81
Jarruvalo	1,6	Akku–ohjainlaite 10	Ohjainlaite–valo 110	0,19
Peruutusvalo	1,6	Akku–ohjainlaite 10	Ohjainlaite–valo 140	0,24
Takalasinlämmitin	5	Akku–ohjainlaite 10	Ohjainlaite–vastus 140	0,75

Tuloksista voidaan päätellä, että häviöt pienenivät huomattavasti. Häviöitä saataisiin vähennettyä vielä enemmän uusimalla itse virtaa kuluttavat laitteet, sillä varsinkin valoissa on liitoksia, joissa on kulumaa ja korroosiota.

Mittaustuloksissa on otettu valojen akkujännitteen mukainen PWM-säätö pois käytöstä, jotta tuloksista saadaan vertailukelpoisia alkutilanteeseen. Jos PWM-

säätö olisi käytössä, se aiheuttaisi mittaustuloksiin ylimääräisen näennäisen jännitehäviön.

Taulukossa 4 verrataan tehohäviöitä vanhan ja uuden järjestelmän välillä.

Taulukko 4. Tehohäviöiden vertailu

Toimilaite	**Vanha tehohäviö (W)**	**Uusi tehohäviö (W)**	**Ero (W)**
Vasen lähivalo	3,07	1,00	2,07
Oikea lähivalo	3,75	0,65	3,10
Pyyhin	9,00	2,03	6,97
Vasen kaukovalo	4,27	0,92	3,35
Oikea kaukovalo	5,20	0,81	4,39
Jarruvalo	1,30	0,19	1,11
Peruutusvalo	0,84	0,24	0,60
Takalasin lämmitin	3,75	0,75	3

8 Yhteenveto

Työssä korvattiin Mercedes-Benz 190E -mallisen henkilöauton korin sähköjärjetelmä nykyaikaisella väylätekniikalla. Työn tuloksena saatiin toimiva ja moderni korin sähköjärjestelmä, jonka johtosarjoista ja kytkennöistä saatiin huomattavasti yksinkertaisempia kuin ne alun perin olivat. Lisäksi auton massaa saatiin pudotettua 5 kg. Jos autossa olisi ollut enemmän elektroniikkaa, massan pudotus olisi ollut vielä merkittävämpi.

Lähes kaikkien paljon virtaa kuluttavien laitteiden sulakoinnit ja relekytkennät saatiin korvattua virran mittaukseen perustuvalla ohjelmallisella suojauksella. Tämä vähensi kuluvien liitosten ja komponenttien määrää huomattavasti. Tämä lisää järjestelmän luotettavuutta ja helpottaa mahdollista vianhakua merkittävästi tulevaisuudessa.

Sähköjärjestelmän tehohäviöt putosivat myös merkittävästi, mikä osaltaan myös pidentää johtosarjojen käyttöikää. Tehohäviöiden pudottamisella on myös marginaalinen vaikutus polttoaineenkulutukseen generaattorin kuorman vähentyessä.

Järjestelmään on tulevaisuudessa helppoa lisätä toimilaitteita ja varusteita.

9 Jatkokehitysmahdollisuudet

9.1 Lepovirrankulutus

Työssä käytetyt mikrokontrollerit voidaan mahdollisesti vaihtaa energiatehokkaampiin ja paremmilla virransäästöominaisuuksilla varustettuihin malleihin. Lisäksi MCP2562-piirien virransäästötilat voidaan ohjelmoida käyttöön.

PMU-ohjainlaitteet herätetään kytkemällä jännite heräteliitäntään. Jotta kaikki tarvittavat toiminnot olisivat tarvittaessa käytettävissä, tarvittaisiin lisäksi erillinen ohjainlaite herättämään PMU-ohjainlaitteet.

9.2 LIN-väylän hyödyntäminen

Käytettyjä johdotuksia voidaan vähentää hyödyntämällä jokaisessa moniasentoisessa katkaisimessa tai katkaisinpaneelissa mikrokontrolleria ja LIN-väylää. LIN-väylän käyttäminen vähentäisi myös johdotuksia, sillä katkaisimien asennot voidaan tuoda yhtenä LIN-viestikehyksenä ohjainlaitteelle, joka toimisi tässä tilanteessa yhdyskäytävänä LIN-väylien ja CAN-väylän välillä.

9.3 Lisävarusteasennukset

Väyläteknologiaa käyttäessä voidaan CAN-väylällä olevia signaaleja hyödyntää myös muihin tarkoituksiin, esimerkiksi perävaunun valojen ohjaamiseen.

Lähteet

1 Bosch Automotive Electrics and Automotive Electronics. 2007. Robert Bosch GmbH. Plochingen: John Wiley & Sons. Inc. and Bentley.

2 Paret, Dominique. 2007. Multiplexed Networks for Embedded Systems. Hoboken: John Wiley & Sons. Inc.

3 CAN bus. 2003. Muokattu 16.4.2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/CAN_bus>. Luettu 1.5.2016.

4 Valtanen, Esko. 2016. Tekniikan taulukkokirja 21. painos. Mikkeli: Genesis-Kirjat Oy.

5 Arduino Due. Verkkoaineisto. Arduino. <https://store.arduino.cc/arduino-due>. Luettu 8.6.2021.

6 High-Speed CAN Transceiver. 2013. Verkkoaineisto. Microchip Technology Inc. <https://ww1.microchip.com/downloads/en/DeviceDoc/20005167C.pdf>. Luettu 10.6.2021.

7 Arduino Nano Every. Verkkoaineisto. Arduino. <https://store.arduino.cc/arduino-nano-every>. Luettu 8.6.2021.

8 PMU. 2021. Verkkoaineisto. Ecumaster. <https://www.ecumaster.com/products/pmu/>. Luettu 10.6.2021.

Arduino Duen ohjelmiston lähdekoodi

```
#include <Scheduler.h>
#include <due_can.h>
#include <variant.h>
#include <math.h>

#define txCanId 0x400 //lähetettävän viestikehyksen ID
#define id_etu 0x500 //etu-PMU
#define id_taka 0x501 //taka-PMU

#define Serial SerialUSB
byte viat = 0b00000000; // 7: CAN-väylä 1: ikkunat 0: peilit

byte data_0 = 0b00000000;
byte data_1 = 0b00000000;
byte data_2 = 0b00000000;
byte data_3 = 0b00000000;
byte data_4 = 0b00000000;

byte datain_0 = 0b00000000;
byte datain_1 = 0b00000000;
byte datain_2 = 0b00000000;
byte datain_3 = 0b00000000;
byte datain_4 = 0b00000000;
byte datain_5 = 0b00000000;
byte datain_6 = 0b00000000;
byte datain_7 = 0b00000000;

byte datain2_0 = 0b00000000;
byte datain2_1 = 0b00000000;
byte datain2_2 = 0b00000000;
byte datain2_3 = 0b00000000;
byte datain2_4 = 0b00000000;
byte datain2_5 = 0b00000000;
byte datain2_6 = 0b00000000;
byte datain2_7 = 0b00000000;

bool highBeam = 0;
bool brakeWear = 0;
bool brakeFault = 0;
bool oilLevel = 0;
bool coolantLevel = 0;
bool washerLevel= 0;
bool fuelRes = 0;
unsigned int fuelLevel_r = 0b0000000000;
float fuelLevel = 0;
bool leftBlink = 0;
bool rightBlink = 0;

bool window_lu = 0;
bool window_ld = 0;
bool window_ru = 0;
bool window_rd = 0;

bool mirror_left = 0;
bool mirror_right = 0;
bool mirror_up = 0;
bool mirror_down = 0;

const float fuel_a = 312.47; //polttoainemäärä ax^2+bx+c
const float fuel_b = 164.72;
const float fuel_c = 493.51;
```

```
unsigned int fuel_dc = 0;

const float engTemp_a = -0.069; //moottorin lämpötila ax^2+bx+c
const float engTemp_b = 17.754;
const float engTemp_c = -152.58;
unsigned int engTemp_dc = 0;

const float oilPres_a = 26.624; //öljynpaine ax^2+bx+c
const float oilPres_b = -264.81;
const float oilPres_c = 892.31;
unsigned int oilPres_dc = 0;

void setup() {
// GPIO-asetukset

//Mittaristo:
pinMode(2,OUTPUT);  //vasemman vilkun merkkivalo
pinMode(3,OUTPUT);  //oikean vilkun merkkivalo
pinMode(5,OUTPUT);  //polttoainemittari (PWM)
pinMode(6,OUTPUT);  //moottorin lämpötila (PWM)
pinMode(7,OUTPUT);  //öljynpainemittari (PWM)
pinMode(8,OUTPUT);  //kierroslukumittari (PWM)
pinMode(9,OUTPUT);  //kaukovalojen merkkivalo
pinMode(10,OUTPUT); //latausvalo
pinMode(11,OUTPUT); //jarrupalojen varoitusvalo
pinMode(12,OUTPUT); //jarrujärjestelmän varoitusvalo
pinMode(13,OUTPUT); //öljytason merkkivalo
pinMode(14,OUTPUT); //jäähdytysnestetason merkkivalo
pinMode(15,OUTPUT); //tuulilasin pesunestemäärän merkkivalo
pinMode(16,OUTPUT); //polttoainemäärän varoitusvalo
pinMode(A0,INPUT);  //ajonopeus

//Valokatkaisin:
pinMode(17,INPUT_PULLUP);  //vasen seisontavalo
pinMode(18,INPUT_PULLUP);  //oikea seisontavalo
pinMode(19,INPUT_PULLUP);  //seisontavalot
pinMode(20,INPUT_PULLUP);  //ajovalot
pinMode(21,INPUT);  //etusumuvalo
pinMode(22,INPUT);  //takasumuvalo

//Viiksikatkaisin:
pinMode(23,INPUT_PULLUP); //vasen vilkku
pinMode(24,INPUT_PULLUP); //oikea vilkku
pinMode(25,INPUT_PULLUP); //kaukovalo
pinMode(26,INPUT_PULLUP); //ohitusmerkki
pinMode(27,INPUT_PULLUP); //pyyhkijä 1
pinMode(28,INPUT_PULLUP); //pyyhkijä 2
pinMode(29,INPUT_PULLUP); //pyyhkijä 3
pinMode(30,INPUT_PULLUP); //pesuri
pinMode(31,INPUT_PULLUP); //äänimerkki

//vasen ovi ja jarruvalo:
pinMode(32,INPUT); //jarruvalokytkin
pinMode(33,INPUT_PULLUP); //vasemman oven tila


//keskiosa:
pinMode(34,OUTPUT);   //sisävalo (ovi)
pinMode(35,INPUT_PULLUP);   //kattoluukku
pinMode(36,INPUT_PULLUP);   //kattoluukku
pinMode(37,INPUT_PULLUP);   //kattoluukku
pinMode(38,INPUT); //takalasin lämmitin off
pinMode(39,INPUT);  //takalasin lämmitin on
pinMode(40,INPUT_PULLUP); //vasen ikkuna ylös
pinMode(41,INPUT_PULLUP); //vasen ikkuna alas
pinMode(42,INPUT_PULLUP);  //hätävilkku
```

```
pinMode(43,INPUT_PULLUP); //peruutusvalo
pinMode(44,INPUT_PULLUP); //seisontajarru
pinMode(45,INPUT_PULLUP); //oikea ikkuna ylös
pinMode(46,INPUT_PULLUP); //oikea ikkuna alas
pinMode(A1,INPUT); //peilin säätö nasta 1
pinMode(A2,INPUT);  //peilin säätö nasta 4
pinMode(A3,INPUT_PULLUP); //peilin säätö nasta 5
pinMode(47,INPUT_PULLUP); //puhallin asento 1
pinMode(48,INPUT_PULLUP); //puhallin asento 2
pinMode(49,INPUT_PULLUP); //puhallin asento 3
pinMode(50,INPUT_PULLUP); //puhallin asento 4
pinMode(4,OUTPUT); //hätävilkun valo

//oikea ovi:
pinMode(A4,INPUT_PULLUP); //oikean oven tila

//CAN-väylät:
pinMode(51,OUTPUT); //Standby 0
pinMode(52,OUTPUT); //Standby 1

Serial.begin(115200);
analogWriteResolution(10);
Scheduler.startLoop(debug_info);


//CAN-väylä
digitalWrite(51,LOW);
if (Can0.begin(CAN_BPS_1000K,51)) { //CAN-väylän alustus nopeudella 1 Mbit/s
}
else {
  bitWrite(viat,7,1);
}
Can0.watchFor();
}

//void mirror_adj() {
// if (digitalRead(A3) == 1) {
//  if (analogRead(A2) < 300 && analogRead(A1) < 700 && analogRead(A1) > 300) {
//    bitWrite(data_3,1,1);
//  }
//  else if (analogRead(A8) < 300 && analogRead(A9) < 700 && analogRead(A9) > 300) {
//    digitalWrite(52,HIGH);
// }
// else {
// digitalWrite(49,LOW);
// digitalWrite(52,LOW);
// digitalWrite(50,LOW);
// digitalWrite(51,LOW);
// }
// }
// else {
//    if (analogRead(A9) > 700 && analogRead(A8) < 700 && analogRead(A8) > 300) {
//    digitalWrite(50,HIGH);
//  }
//  else if (analogRead(A8) > 700 && analogRead(A9) < 700 && analogRead(A9) > 300) {
//    digitalWrite(51,HIGH);
// }
// else {
// digitalWrite(49,LOW);
// digitalWrite(52,LOW);
// digitalWrite(50,LOW);
// digitalWrite(51,LOW);
// }
// }
//  yield();
//}
```

```
//Viestikehyksen bitit:
void refresh() {
bitWrite(data_0,0,!digitalRead(42)); //hätävilkku
bitWrite(data_0,1,!digitalRead(31)); //äänimerkki
bitWrite(data_0,2,digitalRead(22)); //takasumuvalo
bitWrite(data_0,3,digitalRead(21)); //etusumuvalo
bitWrite(data_0,4,!digitalRead(20)); //ajovalot
bitWrite(data_0,7,!digitalRead(19)); //seisontavalot
bitWrite(data_0,5,!digitalRead(18)); //oikea seisontavalo
bitWrite(data_0,6,!digitalRead(17)); //vasen seisontavalo

bitWrite(data_1,0,!digitalRead(30)); //pesuri
bitWrite(data_1,1,!digitalRead(29)); //pyyhkijä3
bitWrite(data_1,2,!digitalRead(28)); //pyyhkijä2
bitWrite(data_1,3,!digitalRead(27)); //pyyhkijä1
bitWrite(data_1,4,!digitalRead(26)); //ohitusmerkki
bitWrite(data_1,5,!digitalRead(25)); //kaukovalo
bitWrite(data_1,6,!digitalRead(24)); //oikea vilkku
bitWrite(data_1,7,!digitalRead(23)); //vasen vilkku

bitWrite(data_2,0,digitalRead(39)); //takalasi E
bitWrite(data_2,1,digitalRead(38)); //takalasi A
bitWrite(data_2,2,!digitalRead(43)); //peruutusvalo
bitWrite(data_2,3,digitalRead(32)); //jarruvalo
bitWrite(data_2,4,!digitalRead(47)); //puhallin 1
bitWrite(data_2,5,!digitalRead(48)); //puhallin 2
bitWrite(data_2,6,!digitalRead(49)); //puhallin 3
bitWrite(data_2,7,!digitalRead(50)); //puhallin 4


bitWrite(data_3,0,!digitalRead(35)); //kattoluukku
bitWrite(data_3,1,!digitalRead(36)); //kattoluukku
bitWrite(data_3,2,!digitalRead(37)); //kattoluukku
bitWrite(data_3,4,0);
bitWrite(data_3,5,0);
bitWrite(data_3,6,0);
bitWrite(data_3,6,0);
bitWrite(data_3,7,0);

bitWrite(data_4,0,window_lu);
bitWrite(data_4,1,window_ld);
bitWrite(data_4,2,window_ru);
bitWrite(data_4,3,window_rd);
bitWrite(data_4,4,mirror_left);
bitWrite(data_4,5,mirror_right);
bitWrite(data_4,6,mirror_up);
bitWrite(data_4,7,mirror_down);
}

void debug_info() {
   Serial.print(datain_0,HEX);
   Serial.print(datain_1,HEX);
   Serial.print(datain_2,HEX);
   Serial.print(datain_3,HEX);
   Serial.print(datain_4,HEX);
   Serial.print(datain_5,HEX);
   Serial.print(datain_6,HEX);
   Serial.print(datain_7,HEX);
   Serial.println();
   Serial.print(datain2_0,HEX);
   Serial.print(datain2_1,HEX);
   Serial.print(datain2_2,HEX);
   Serial.print(datain2_3,HEX);
   Serial.print(datain2_4,HEX);
   Serial.print(datain2_5,HEX);
```

```
    Serial.print(datain2_6,HEX);
    Serial.print(datain2_7,HEX);
    delay(1000);

}
void window() {
  if (digitalRead(40) == 1) {
    if (digitalRead(41) == 0) {
      window_lu = 1;
      window_lu = 0;
      window_ld = 0;
      window_ru = 0;
      window_rd = 0;
    }
    else {
      bitWrite(viat,1,1);
    }
  }
  else if (digitalRead(41) == 1) {
    if (digitalRead(40) == 0) {
      window_ld = 1; //vasen alas
    }
    else {
      bitWrite(viat,1,1);
      window_lu = 0;
      window_ld = 0;
      window_ru = 0;
      window_rd = 0;
    }
  }
  if (digitalRead(45) == 1) {
    if (digitalRead(46) == 0) {
      window_ru = 1; //oikea ylös
    }
    else {
      bitWrite(viat,1,1);
      window_lu = 0;
      window_ld = 0;
      window_ru = 0;
      window_rd = 0;
    }
  }
  else if (digitalRead(46) == 1) {
    if (digitalRead(45) == 0) {
      window_rd = 1; //oikea alas
    }
    else {
      bitWrite(viat,1,1);
      window_lu = 0;
      window_ld = 0;
      window_ru = 0;
      window_rd = 0;
    }
  }
  else {
    window_lu = 0;
    window_ld = 0;
    window_ru = 0;
    window_rd = 0;
  }
}



void int_light() {
  if (digitalRead(33) == 1 || digitalRead(A4) == 1) {
```

```
    digitalWrite(34,HIGH);
  }
}

void send_CAN() {
  CAN_FRAME txFrame;
  txFrame.id = txCanId;
  txFrame.priority = 4;
  txFrame.length = 5;
  txFrame.data.byte[0] = data_0;
  txFrame.data.byte[1] = data_1;
  txFrame.data.byte[2] = data_2;
  txFrame.data.byte[3] = data_3;
  txFrame.data.byte[4] = data_4;
  //txFrame.data.byte[5] = data_5;
  //txFrame.data.byte[6] = data_6;
  //txFrame.data.byte[7] = data_7;
  Can0.sendFrame(txFrame);
  //delay(100);
}

void loop() {
  refresh();
  //Serial.println(viat);
  CAN_FRAME rxFrame;
  static unsigned long aika = 0;
  if (Can0.available() > 0) {
    Can0.read(rxFrame);
  }
  if (rxFrame.id == id_etu) {
    datain_0 = rxFrame.data.byte[0];
    datain_1 = rxFrame.data.byte[1];
    datain_2 = rxFrame.data.byte[2];
    datain_3 = rxFrame.data.byte[3];
    datain_4 = rxFrame.data.byte[4];
    datain_5 = rxFrame.data.byte[5];
    datain_6 = rxFrame.data.byte[6];
    datain_7 = rxFrame.data.byte[7];
  }
  else if (rxFrame.id == id_taka) {
    datain2_0 = rxFrame.data.byte[0];
    datain2_1 = rxFrame.data.byte[1];
    datain2_2 = rxFrame.data.byte[2];
    datain2_3 = rxFrame.data.byte[3];
    datain2_4 = rxFrame.data.byte[4];
    datain2_5 = rxFrame.data.byte[5];
    datain2_6 = rxFrame.data.byte[6];
    datain2_7 = rxFrame.data.byte[7];
  }

  for (int i=0; i<10; i++) {
    if (i < 8) {
      bitWrite(fuelLevel_r,i,bitRead(datain2_0,i));
    }
    else {
      bitWrite(fuelLevel_r,i,bitRead(datain2_1,i-8));
    }
  }
  fuelLevel = fuelLevel_r/1024;
  fuel_dc = round(fuel_a * fuelLevel * fuelLevel + fuel_b * fuelLevel + fuel_c); //polttoainemittarin duty cycle
  highBeam = bitRead(datain_0,6);
  brakeWear = bitRead(datain_0,7);
  coolantLevel = bitRead(datain_0,4);
  washerLevel = bitRead(datain_0,5);
  fuelRes = bitRead(datain2_2,0);
  brakeFault = bitRead(datain_0,3)||digitalRead(44);
```

```
 leftBlink = bitRead(datain_3,0);
 rightBlink = bitRead(datain_3,1);

 //moottorinohjaus:
 //oilLevel
 //if (engineTemp > 35.00) {
 //  engTemp_dc = round(engTemp_a * engineTemp * engineTemp + engTemp_b * engineTemp + eng-
Temp_c); //lämpömittarin duty cycle
 //}
 //analogWrite(6,engTemp_dc);
 //if (oilPressure < 3.00) {
 //oilPres_dc = round(oilPres_a * oilPressure * oilPressure + oilPres_b * oilPressure + oilPres_c);
 //}
 //analogWrite(7,oilPres_dc);
 //digitalWrite(13,oilLevel);
 //latausvalo ph
 //RPM Placeholder

 digitalWrite(9,highBeam);
 digitalWrite(11,brakeWear);
 digitalWrite(12,brakeFault);
 digitalWrite(14,coolantLevel);
 digitalWrite(15,washerLevel);
 digitalWrite(16,fuelRes);
 analogWrite(5,fuel_dc);
 digitalWrite(2,leftBlink);
 digitalWrite(3,rightBlink);
 if (!digitalRead(42)) {
   digitalWrite(4,leftBlink);
 }
 int_light();
 window();
 send_CAN();
 //Debug-osio
 Serial.println("OK");
 Serial.println(datain_0);
 Serial.println(datain2_0);

 delay(10);
}
```

Oven ohjainlaitteen ohjelmiston lähdekoodi

Seuraavilla riveillä on oikean oven ohjainlaitteen ohjelmiston lähdekoodi. Vasemman oven ohjainlaite on muuten vastaava, mutta siinä ei ole peilin säätötoimintoja.

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

const int CS = 10;
Watchdog wd;
byte inputs = 0b00000000;
byte viat = 0b00000000;

bool window_lu = 0;
bool window_ld = 0;
bool window_ru = 0;
bool window_rd = 0;

bool mirror_left = 0;
bool mirror_right = 0;
bool mirror_up = 0;
bool mirror_down = 0;

float rsense = 0.005;
float maxCur_0 = 15.00;
float maxCur_1 = 5.00;
unsigned int senseGain = 20;
unsigned int cur_0 = 0;
unsigned int cur_1 = 0;

unsigned int wait_time = 1000;
unsigned int trip_time = 0;
unsigned int trip_time2 = 0;

MCP_CAN CAN(CS);

void setup() {
  Serial.begin(115200);
  Serial.println("Setup");
  cur_0 = 1023 / 5 * rsense * maxCur_0 * senseGain;
  cur_1 = 1023 / 5 * rsense * maxCur_1 * senseGain;
  pinMode(3,OUTPUT); //peili 1
  pinMode(4,OUTPUT); //peili 4
  pinMode(5,OUTPUT); //peili 5
  pinMode(6,OUTPUT); //peilin lämmitys
  pinMode(7,OUTPUT); //ikkuna 2
  pinMode(8,OUTPUT); //ikkuna 1
  pinMode(9,OUTPUT); //transceiver stby
  pinMode(A0,INPUT); //ikkunoiden virranmittaus
  pinMode(A1,INPUT); //peilin säädön virranmittaus
  pinMode(A2,INPUT); //yleistulo (lukon tila?)

  digitalWrite(9,LOW);
  wd.enable(Watchdog::TIMEOUT_500MS);

  CAN_INIT:
    if(CAN_OK == CAN.begin(CAN_1000KBPS)) {
      Serial.println("CAN controller init ok!");
    }
```

```
    else {
      Serial.println("CAN controller init failed!");
      delay(100);
      goto CAN_INIT;
    }
}

void window() {
  if (window_lu == 1) {
    if (window_ld == 0) {
      if (analogRead(A0) <= cur_0) {
        if (trip_time+wait_time <= millis()) {
          digitalWrite(8,HIGH);
        }
        else {
          digitalWrite(8,LOW);
        }
      }
      else {
        trip_time = millis();
        digitalWrite(8,LOW);
        }
    }
    else {
      bitWrite(viat,0,0);
      digitalWrite(8,LOW);
      digitalWrite(7,LOW);
    }
  }

  else if (window_ld == 1) {
    if (window_lu == 0) {
      if (analogRead(A0) <= cur_0) {
        if (trip_time+wait_time <= millis()) {
          digitalWrite(7,HIGH);
        }
        else {
          digitalWrite(7,LOW);
        }
      }
      else {
         trip_time = millis();
         digitalWrite(7,LOW);
         }
      }
      else {
        bitWrite(viat,0,0);
        digitalWrite(8,LOW);
        digitalWrite(7,LOW);
    }
  }
  if (window_lu == 0) {
    digitalWrite(8,LOW);
  }
  else if (window_ld == 0) {
    digitalWrite(7,LOW);
  }

}

void mirror() {
  if (analogRead(A1) <= cur_1) {
    if (trip_time2+wait_time <= millis()) {
      if (mirror_up == 1) {
        if (mirror_down  == 0 && mirror_left == 0 && mirror_right == 0) {
          digitalWrite(4,HIGH);
```

```
        digitalWrite(3,LOW);
        digitalWrite(5,LOW);
       }
       else {
        bitWrite(viat,0,1);
        digitalWrite(3,LOW);
        digitalWrite(4,LOW);
        digitalWrite(5,LOW);
       }
      }

      else if (mirror_down == 1) {
       if (mirror_up  == 0 && mirror_left == 0 && mirror_right == 0) {
        digitalWrite(3,HIGH);
        digitalWrite(4,LOW);
        digitalWrite(5,HIGH);
       }
       else {
        bitWrite(viat,0,1);
        digitalWrite(3,LOW);
        digitalWrite(4,LOW);
        digitalWrite(5,LOW);
       }
      }
      else if (mirror_left == 1) {
       if (mirror_down  == 0 && mirror_up == 0 && mirror_right == 0) {
        digitalWrite(3,LOW);
        digitalWrite(4,HIGH);
        digitalWrite(5,HIGH);
       }
       else {
        bitWrite(viat,0,1);
        digitalWrite(3,LOW);
        digitalWrite(4,LOW);
        digitalWrite(5,LOW);
       }
     }
      else if (mirror_right == 1) {
       if (mirror_down  == 0 && mirror_left == 0 && mirror_up == 0) {
        digitalWrite(3,HIGH);
        digitalWrite(4,LOW);
        digitalWrite(5,LOW);
       }
       else {
        bitWrite(viat,0,1);
        digitalWrite(3,LOW);
        digitalWrite(4,LOW);
        digitalWrite(5,LOW);
       }
      }
      else {
      digitalWrite(3,LOW);
      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
      }

    }
    else {
     digitalWrite(3,LOW);
     digitalWrite(4,LOW);
     digitalWrite(5,LOW);
    }
    }
    else {
      trip_time2 = millis();
      digitalWrite(3,LOW);
```

```
    digitalWrite(4,LOW);
    digitalWrite(5,LOW);
  }
}
void loop() {
  unsigned char recId = 0;
  unsigned char recDlc = 0;
  unsigned char rec_frame[8];

  if (CAN_MSGAVAIL == CAN.checkReceive()) {
    CAN.readMsgBuf(&recDlc, rec_frame);
    recId = CAN.getCanId();
    if (recId == 0x400) {
      inputs = rec_frame[4];
    }
  }
  window_lu = bitRead(inputs,0);
  window_ld = bitRead(inputs,1);
  window_ru = bitRead(inputs,2);
  window_rd = bitRead(inputs,3);

  mirror_left = bitRead(inputs,4);
  mirror_right = bitRead(inputs,5);
  mirror_up = bitRead(inputs,6);
  mirror_down = bitRead(inputs,7);

  window();
  mirror();



}
```