Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2021

Kalle Nurminen

CAN-VÄYLÄN SUUNNITTELU JA OHJELMOINTI SULAUTETTUUN JÄRJESTELMÄÄN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2021 | 20 sivua

Kalle Nurminen

CAN-VÄYLÄN SUUNNITTELU JA OHJELMOINTI SULAUTETTUUN JÄRJESTELMÄÄN

Työkoneille ei ole asetettu selvää standardia CAN-väylälle, mikä aiheuttaa ongelmia tällä alalla. Tämän takia ei ole selvää tietoa saisiko työkoneista CAN-dataa ulos, esimerkiksi CAN-datan salausten takia tai ei tiedetä, esimerkiksi CAN-väylän nopeutta eri työkoneissa.

Opinnäytetyön tavoitteena on luoda prototyyppiohjelmisto CAN-väylälle, jolla pystytään saamaan eri valmistajien työkoneista CAN-dataa ulos asiakkaiden seurattavaksi. Prototyyppiohjelmisto testataan toimeksiantajan telematiikkalaitteella ja Ruptelan LCV5 -adapterilla, jolla pystyy mittaamaan CAN-väylän liikennettä induktiivisesti. Prototyyppiohjelmiston tarkoituksena on, esimerkiksi maksimoida asiakkaan saama hyöty työkoneesta, vaikka polttoaineen kulutuksen minimoimisella.

Opinnäytetyössä käydään läpi ohjelmistokoodin suunnittelua ja laitteistoa. Suunnittelusta kehitetään ohjelmistokoodi, joka toimii laitteiston kanssa ottaen CAN-dataa ulos työkoneista. Testausten lopputulokset ohjelmistokoodin ja laitteiston kanssa käydään läpi. Ohjelmistokoodista kerrotaan myös oleellisin osa testauksien kannalta ja bittimuunnoksesta eli CAN-datan muuttamisesta ymmärrettävään muotoon, esimerkin kanssa.

Lopputuloksena saadaan prototyyppiohjelmisto, jolla pystyy ottamaan CAN-dataa ulos trukista. Prototyyppiohjelmiston avulla saadaan lisää tietoa työkoneiden CAN-väylästä, esimerkiksi tiedetään, että CAN-dataa saa ulos Hyundain trukista. Ohjelmistoa tullaan kehittämään opinnäytetyön jälkeenkin ja tulevaisuudessa on mahdollista nähdä tämä laite työkonemarkkinoilla lukemassa CAN-dataa eri valmistajien laitteista.

ASIASANAT:

CAN-väylä, CAN-data, testaus, ohjelmointi, ohjelmistokoodi, bittimaskaus

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2021 | 20 pages

Kalle Nurminen

CAN BUS DESIGN AND PROGRAMMING FOR AN EMBEDDED SYSTEM

There is no clear CAN bus standard for work machines, which causes problems in this area. For this reason, it is not clear whether CAN data could be obtained from work machines, for example because of CAN data encryptions or not known, for example, the speed of the CAN bus in different machines.

The goal of the thesis is to create prototype software for the CAN bus, which can be used to get CAN data out of the machines of different manufacturers for customers to monitor. The prototype software is tested with the telematics device provided by the commissioner and Ruptelan LCV5 adapter, which can measure can bus traffic inductively. The purpose of the prototype software is, for example, to maximise the benefits of the customer from the machine, even by minimizing fuel consumption.

The thesis examines software code design and hardware. The design will be developed into a software code that works with the hardware by taking CAN data out of the machines. The results of the tests with the software code and hardware are reviewed. The most important part of the software code is also described in terms of testing and bit conversion for example changing CAN data to an understandable format, is explained with an example.

The result is prototype software that can take CAN data out of the forklift. Prototype software provides more information about the CAN bus for work machines, for example, it is now known that CAN data can be obtained from a Hyundai forklift. The software will continue to be developed after the thesis, and in the future it will be possible to see this device in the machine market reading CAN data from different manufacturers machines.

KEYWORDS:

CAN bus, CAN data, testing, programming, software code, bit conversion

SISÄLLYS

KÄYTETYT LYHENTEET TAI SANASTO

AND	Ja-toiminto
CAN	CAN-väylä on autoissa ja työkoneissa laitteiden ja antureiden kommunikoimiseen käytetty väylä (Controller Area Network)
CAN-H	CAN-väylän johdin, jonka jännitetaso nousee dominantti tilassa
CAN-L	CAN-väylän johdin, jonka jännitetaso laskee dominantti tilassa
ESP32	Espressif:n valmistama kehitysalusta
MQTT	Kevyt verkkoprotokolla, joka käyttää TCP/IP protokollaa viestien kuljetukseen laitteiden välillä (Message Queuing Telemetry Transport)
OBD2	Ajoneuvodiagnostiikkajärjestelmä, joka lukee esimerkiksi vikakoodeja (On-Board Diagnostics)
PGN	Ainutlaatuinen kehystunniste CAN-väylälle J1939 standardissa (Parameter Group Number)
SPN	Tunnus PGN kehystunnisteelle (Suspect Parameter Group)
UART0	Sarjaportti (Universal Asynchronous Receiver/Transmitter)

1 JOHDANTO

Työkoneilla ei ole yhtä kaikkien noudattamaa CAN-väylä standardia, mikä aiheuttaa ongelmia tällä alalla. Tämän takia ei ole selvää tietoa, saisiko työkoneista CAN-dataa ulos, esimerkiksi eri työkonevalmistajien CAN-datan salausten takia tai ei tiedetä, esimerkiksi CAN-väylän nopeutta eri työkoneissa.

Opinnäytetyön tavoitteena on CAN-väylän prototyyppiohjelmisto, jolla todistetaan kyky saada CAN-data ulos eri valmistajien työkoneista toimeksiantajan palveluun asiakkaiden seurattavaksi. CAN-datan etäseurannan tavoitteena on lisätä asiakkaiden kykyä optimoida laitteiden hyötyä telematiikkadatan avulla, esimerkiksi laskemalla polttoaineen keskikulutusta. Tämä mahdollistaisi myös nopeamman reagoinnin erilaisiin häiriö- tai vikatilanteisiin. Toimeksiantajan tavoitteena on tarjota mixed fleet kalustonhallintapalvelua, jossa telematiikkayksikön laite lukisi CAN-dataa eri valmistajien työkoneista. Näin saisi CAN-datan ulos monen eri valmistajan työkoneista asiakkaiden seurattavaksi.

Opinnäytetyössä käsitellään ohjelmistokoodia, laitteistoa ja testauksien lopputuloksia. Ohjelmistokoodin suunnittelusta kerrotaan ja näytetään oleellinen osa koodia testausten kannalta. Laitteistosta kerrotaan toimeksiantajan telematiikkalaitteesta ja Ruptelan Easy-CAN adapterista. Testaukset käydään läpi lopputuloksineen. Yhteenvedossa käsitellään onnistunutta tavoitetta ja prototyyppiohjelmiston jatkosuunnitelmaa.

Tavoitteena olevan prototyyppiohjelmiston samankaltaisia laitteita on olemassa, mutta ne ovat hintavia. Seuraavaksi tullaan esittelemään vaihtoehtoinen laite CAN-väylälle, joka pystyy lukemaan CAN-väylää ja muuttamaan datan ymmärrettävään muotoon asiakkailleen.

CSS Electronicsin CANedge2 on yrityksen uusin laite, joka pystyy lukemaan CAN-väylää erilaisilla adaptereilla. CANedge2 pystyy myös varastoimaan dataa, joko nettiin tai SD-kortille ja tukee SAE J1939 standardia. CANedge2:sta löytyy myös ennakoivan huollon mahdollisuus. (CSS Electronics)

2 TAUSTATEORIAN MÄÄRITTÄMINEN

2.1 Yritystietoa

Opinnäytetyön toimeksiantajana toimii Fleethub Oy, joka tarjoaa Saas-kalustonhallintapalvelua omassa pilvipalvelussaan asiakkailleen pääasiassa teollisuuteen. Tällä tuotteella asiakkaat pääasiassa tavoittelevat kustannussäästöjä, esimerkiksi tarkkailemalla polttoainekulutusta tai ennakoimalla huoltoa. Fleethub Oy:n telematiikkayksikkö on työkoneisiin asennettava laite, jolla pystyy seuraamaan reaaliajassa työkoneita, esimerkiksi sijaintia. Laitteella pystyy myös hallitsemaan sähköisiä ajolupia. (Fleethub Oy 2020)

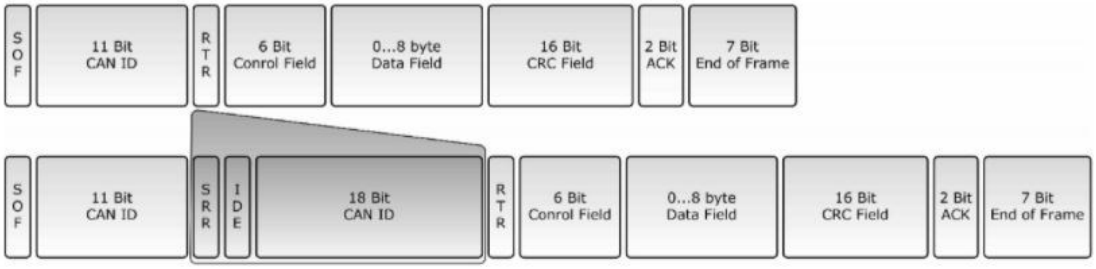
Telematiikkayksikön data on rajallista, eikä vastaa kaikkien asiakkaiden tarpeisiin, mutta elektroniikkasuunnitelussa varauduttiin tähän sisäänrakennetulla CAN-väylälukijalla. CAN-väylä laajentaa telematiikkayksikön asiakaskohderyhmää esimerkiksi, kuljetusyrityksiin.

2.2 CAN-väylä

CAN-väylä on automaatioväylä, joka kehitettiin vuonna 1983 Robert Boschin toimesta aluksi autojen ohjausjärjestelmien reaaliaikaiseen tiedonsiirtoon ja myöhemmin laajalti muualle, esimerkiksi työkoneisiin (Alanen 2000, 1). CAN-väylää käytetään pääsääntöisesti sulautetuissa ohjelmistoissa, koska CAN-väylä takaa nopean yhteyden mikro-ohjainten välillä reaaliaikaisiin vaatimuksiin asti. CAN-väylällä saadaan laitteista erilaista tietoa, esimerkiksi autoissa polttoaineen kulutuksen. Tiedonsiirto nopeus on 125 kb/s – 1 Mb/s erilaisista sovelluksista riippuen. CAN-väylässä on kaksi johtoa, jotka ovat jaettu solmuiksi, joista dataliikenne kulkee analogisessa muodossa. Jokaisella solmulla on vastaanottolähetin ja lähetyksenvahvistus kommunikoimaan CAN-väylään. Trukeissa yleensä käytetään CAN-väylänä tavallista kierrettyä suojaamatonta kaksijohdinväylää, CAN-L, CAN-H ja väylä on päätetty yhdellä 120:n ohmin vastuksella. Tällainen väylä mahdollistaa trukeissa jopa 1Mb/s siirtonopeuden (Gustafsson & Blomqvist 2016, 12). CAN-väylästä on ylemmän kerroksen protokolla, joka on CANopen. CANopen on joustava ratkaisu reaaliaikaisiin sovelluksiin, koska siinä on valmiiksi määritellyt sovellusobjektit suoraan laatikossa (Gustafsson & Blomqvist 2016, 12). CAN-väylän data on linkitetty molempiin viestikehyksiin.

2.2.1 CAN-väylän viestikehys

CAN-väylän viestikehys koostuu kahdesta erilaisesta viestitunnisteesta (Kuva 1.). CAN 2.0A on tarkoitettu pääsääntöisesti autoihin ja tässä versiossa on 11 b:n viestitunniste. CAN 2.0B:ssa on 29 b:n viestintunniste, joka koostuu 11 b:n perustunnisteesta ja 18 b:n tunnisteen laajennuksesta (Kuva 1.). Näiden kahden CAN-version erottamisessa käytetään IDE-bittiä ohjauskentän sisällä. Matala (hallitseva) IDE-bitti käyttää 11 bittistä tunnistetta ja korkea (resessiivinen) IDE-bitti käyttää 29 bittistä tunnistetta. (Wilfried 2008, 5)



Kuva 1. CAN-väylän molemmat viestikehykset (Wilfried 2008).

CAN-protokollassa on neljä erilaista kehystä, sanomakehys, kyselykehys, virhekehys ja ylikuormituskehys (Alanen 2000, 6). Kehyksen alku (engl. SOF) aloittaa kehyksen. Tunnistekenttä (engl. CAN ID) kuuluu sanomakehyksen pääosaan, joka kertoo viestin tunnisteen. IDE-bitti määrittää käyttääkö kehys normaalimuotoista 11 bittistä vai laajennettua 29 bittistä kehystä. Hallintakenttä (engl. Control field) ilmaisee tietokentän pituuden. Tietotavun kentän maksimipituus on 8 B. CRC-kentässä tarkistetaan sisältääkö kehys oikean määrän bittejä. ACK-kentässä kuitataan sanoma. (Alanen 2000, 6)

Kyselykehyksellä voidaan pyytää CAN-väylälle haluttua sanomaa ja se on muodoltaan lähes samallainen kuin sanomakehys, mutta tietokenttää tässä kehyksessä ei ole (Alanen 2000, 6).

Virhekehys lähetetään, jos havaitaan virhe sanomassa. Virhekehys koostuu kuudesta dominanttisesta ja kahdeksasta resessiivisestä bitistä. Tietynlaisissa olosuhteissa virhekehys voi mennä passiiviseen virhetilaan, jossa kuuden dominanttisen bitin sijaan lähetetään resessiiviä bittejä sama määrä. (Alanen 2000, 7)

Jos vastaanottava solmu tarvitsee lisäaikaa sanoman käsittelyyn, niin ylikuormakehys lähetetään sanomien välisenä aikana. Tämä kehys on rakenteeltaan samantapainen

kuin virhekehys. Nykyään uudet protokollapiirit ovat niin kehittyneitä, että ylikuormakehystä ei enää tarvitse. (Alanen 2000, 7)

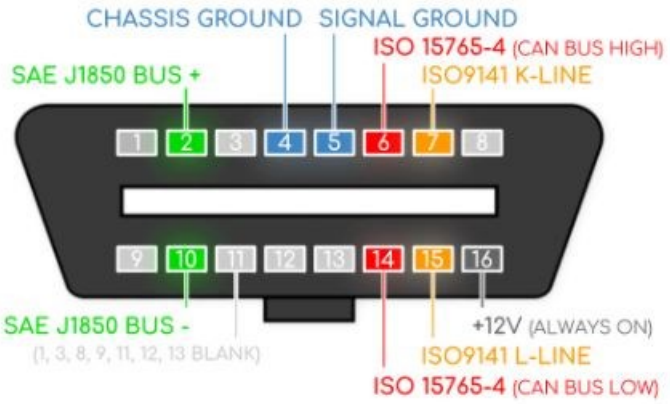
2.2.2 CAN-väylän standardit

CAN-väylälle on määritelty erilaisia standardeja, jotta erilaisten valmistajien laitteet olisivat yhteensopivia, esimerkiksi työkoneille on määritelty standardi SAE J1939. CAN-väylä on kansainvälisesti standardisoitu ISO 11898 standardilla, esimerkiksi ISO 11898-2:2016, joka määrittelee nopean fyysisen kerroksen standardin (Kelechava 2017). Erilaiset standardit määrittelevät erilaisia asioita ja näiden standardien pohjalta on helpompaa ymmärtää ja lukea CAN-väylän dataa ja saada oikeanlaista tietoa ulos laitteista.

Kaikissa autoissa on sama standardi ja niistä saa ulos samaa dataa, mutta trukeilla ei ole yhteistä CAN-standardia, vaan eri trukeista voi saada erilaista dataa ulos, joka aiheuttaa ongelmia tällä alalla. Eri trukkien valmistajat tekevät CAN-väylän eri tavalla laitteisiinsa, jolloin data on erilaista ja yritykset, jotka tekevät CAN-väylän lukijaa työkoneisiin, voivat olla ongelmissa juuri tämän datan kanssa. Joissakin työkoneissa voi olla myös salattua dataa, josta ei ole tietoa, saadaanko tätä dataa ulos ulkopuolisilla laitteilla ollenkaan. (O’Connell 2016)

2.2.3 CAN-väylän datan lukijat

CAN-väylälle on erilaisia lukijoita, joilla saadaan dataa ulos laitteista. Yksi lukijoista on OBD2-laite (Kuva 2.), jolla saadaan esimerkiksi dataa ulos autoista. OBD2-lukija tukee monia standardeja, jolloin sillä saadaan monista muistakin laitteista dataa ulos, esimerkiksi se tukee J1939 standardia, jolloin saadaan dataa ulos työkoneista. OBD2-lukijassa on 16-lokeroinen liitin, joka tukee molempia CAN-versioita (CAN2.0A ja CAN2.0B). OBD2-lukijalla saadaan CAN-dataa ulos suoraan esimerkiksi Bluetoothin tai jonkinlaisen sovelluksen kautta. OBD2-lukijan paikka on vuodesta 2008 määrätty kaikkiin USA:n tehtailla tehtyihin autoihin ja vuodesta 2003 EU:ssa tehtyihin autoihin. (CSS Electronics 2021)



Kuva 2. OBD2-lukijan rakenne (CSS Electronics 2021).

2.2.4 PGN- ja SPN-viestit SAE J1939 standardissa

Opinnäytetyössä keskitytään CAN-väylän osalta tähän SAE J1939 standardin PGN ja SPN viesteihin, joista ohjelmoidaan bittimaskauksella raaka CAN-data ymmärrettäväksi. SAE J1939 käyttää hyödyksi CAN:n 29-bittistä viestikehystä. SAE J1939 käyttää ennalta määrättyjä parametritaulukoita, jotka pitävät protokollan ymmärrettävällä tasolla. Parametritaulukoina ovat esimerkiksi, moottorin lämpötila, joka sisältää itsessään, esimerkiksi jäähdytysnesteen lämpötilan (Kuva 3.). Ohjelmoidessa näitä on tärkeää tietää PGN:n viestin prioriteetti ja sovelluksen solmun osoite, joka esittää CAN 29-bittisen viestikehyksen tunnistetta. SPN kuvaa parametriryhmän tiettyä parametria, jossa niitä kuvataan yksityiskohtaisemmin. (Copperhilltech 2021)

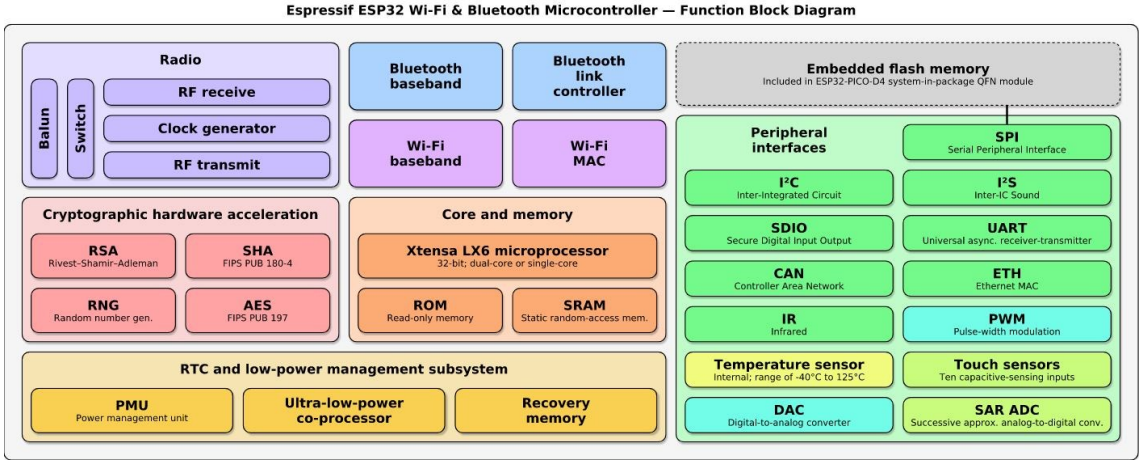
SPN 110 - Engine Coolant Temperature
Temperature of liquid engine cooling system
- Data Length: 1 Byte
- Resolution: 1 deg C / Bit
- Offset: -40 deg C
- Data Range: -40 to 210 deg C
- Type: Measured
- Reference: PGN 65262

Kuva 3. SPN 110 kuvaa jäähdyttimen lämpötilaa, jossa näkyy kuvauksen tiedot ja viittaus PGN-koodiin (Copperhilltech 2021).

2.3 Kehitysalusta ja laiteympäristö

Opinnäytetyössä käytettävä laiteohjelmisto CAN-väylälle on Fleethub Oy:n telematiikkayksikkö, jossa on Espressif:n valmistama ESP32 mikro-ohjain kahdella 32-bittisellä Xtensa LX6 prosessorilla (Kuva 4.). Telematiikkayksikkö tarjoaa myös LTE/GPRS modeemin matkapuhelinverkkoyhteydelle, kiihtyvyysanturin, GPS:n, Bluetoothin, Wi-Fin, RFID-lukijan, releohjauksen ja ADC-muuntimen.

Telematiikkayksikön laitteessa on kaapeloinnit CAN-H ja CAN-L johtimille, jotka liitetään työkoneen CAN-väylään. Laitteessa on Texas Instumentin valmistama CAN-lähetin-vastaanotin, jossa on kansainväliset standardit ISO 11898 ja ISO 11898-2. Laiteohjelmiston CAN kommunikoi lähetin-vastaanottimen kanssa ESP32:n UART0-sarjaportilla. CAN-viestiliikenteestä otetaan viestit, jotka bittimaskataan ja prosessoidaan laiteohjelmistossa ja sen jälkeen lähetetään Fleethub Oy:n palvelimelle MQTT-viestein.

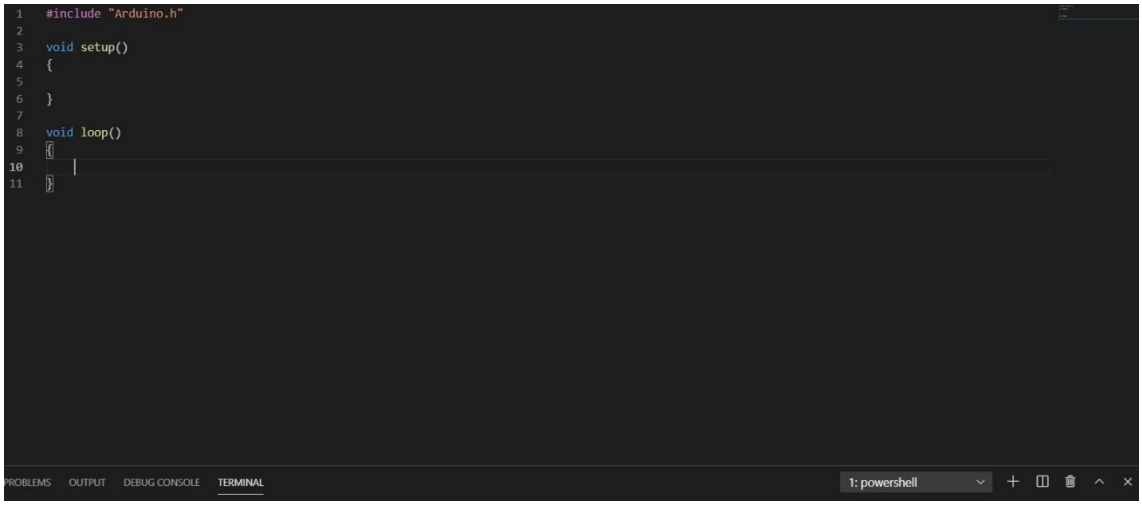


Kuva 4. ESP32:n arkkitehtuuri. (Wikipedia 2018).

2.4 Ohjelmointi

Opinnäytetyössä käytetään PlatformIO laajennusta, joka on ladattavissa ja käytettävissä Visual Studio Codessa (Kuva 5.). PlatformIO on avoimen lähdekoodin systeemi kehitykselle, mikä tarjoaa hyvän debuggausalustan, testauksen, sarjaporttiliikenteen seurannan ja työkalupakin, jolla voi kääntää ja siirtää koko laiteohjelmiston laitteeseen. PlatformIO toimii samoin tavoin kuin Arduino IDE -sovellus, mutta on monipuolisempi tarjonnaltaan. Visual Studio Codessa käytetään C++ -ohjelmointikieltä CAN-väylän ohjelmoimiseen.

Visual Studio Codessa voi ladata lisäosia myös C++ -ohjelmointikieleen, esimerkiksi C/C++ Extension Pack. Bittimuunnoksella muutetaan CAN-väylän raakadata sopivaksi ohjelmistoihin ymmärrettäväksi. Raakadata on yleensä binäärimuodossa eli 0 tai 1 arvoisia ja niitä tulee ulos laitteista CAN-väylän viestikehyksen mukaisesti, mutta se voi olla myös heksadesimaalijärjestelmän muodossa ulos ottaessa.



Kuva 5. Visual Studio Coden näkymä, jossa on setup ja loop kirjoittettu.

2.5 FreeRTOS

Laiteohjelmistossa käytetään FreeRTOS reaaliaikakäyttöjärjestelmää, joka on C-kielen avoin lähdekoodi, joka mahdollistaa monipuolisen tavan hallita synkronointia ja kommunikaatiota säikeiden ja taskien välillä. FreeRTOS on lisensioitu GPL-lisenssillä, joka mahdollistaa sen käytön kaupallisen käytön suljetussa lähdekoodin projektissa ja kaupallisen jakelun ilman, että tuotteiden oma koodi tulisi avoimeksi lähdekoodiksi. FreeRTOS ohjelmassa on taskeja, jotka toteutetaan aliohjelmina ohjelmistossa. Opinnäytetyössä käytetään FreeRTOS:n taskeja eli tehtäviä ja queueita eli jonoja.

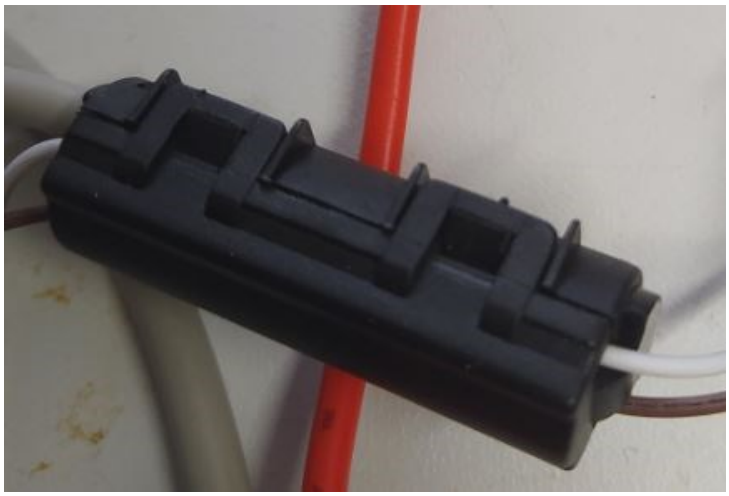
Taskit ovat eriarvoisia, joita pystyy suorittamaan erilaisissa järjestyksissä. Ohjelmistokoodin tehtäviä pystyy laittamaan tehtäväksi tiettyyn järjestykseen. Isomman arvon omaavat taskit suoritetaan ensiksi. Taski luodaan xTaskCreate() funktiolla. (Shawnnymel 2020)

Queue eli jono on tehtävienvälisen viestinnän muoto. Niitä käytetään viestien lähettämiseen tehtävien välillä. Jono tehdään xQueueCreate() funktiolla ja xQueueSendToBack() lähettää jonoon tehtäviä järjestyksessä, jolloin xQueueReceive() vastaanottaa niitä siinä järjestyksessä ja suorittaa ne. Näin jono ei tule koskaan täyteen vaan saadaan suoritettua tehtävät järjestyksessä. (FreeRTOS Queues)

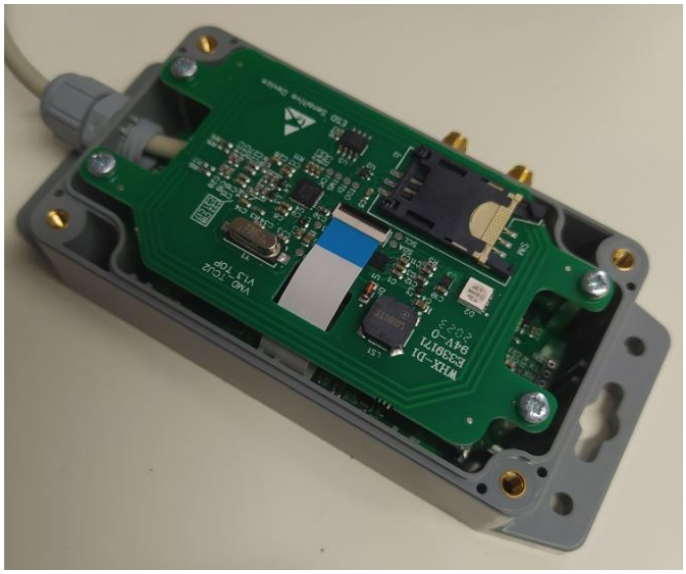
3 SUUNNITTELU JA TESTAUS

3.1 Laitteisto

Opinnäytetyössä käytetään telematiikkayksikön laitteistoa (Kuva 7.), jossa on ESP32:n mikro-ohjain kahdella 32-bittisellä Xtensa LX6 prosessorilla. Telematiikkalaitteesta löytyy tarvittavat asiat CAN-väylälle, kuten CAN-L ja CAN-H johtimet ja CAN-lähetin-vastaanotin, jossa on tarvittavat standardit. Tähän lisänä on Ruptelan EasyCAN LCV5 adapteri (Kuva 6.), joka pystyy mittaamaan CAN-väylän liikennettä induktiivisesti ja se käyttää SAE J1939 standardia hyväkseen. LCV5 adapterilla pystyy myös seuraamaan liikennettä GPS:n avulla, käyttämään tietoverkkoja ja Bluetoothia hyväksi (Ruptela 2021).



Kuva 6. Kuvassa musta Ruptelan EasyCAN LCV5 adapteri, jossa on johdot sisällä CAN-väylän mittaamista varten.



Kuva 7. Kuvassa on telematiikkalaite, jota käytetään opinnäytetyössä.

3.2 Ohjelmistokoodin suunnittelu

CAN-vastaanottimen ohjelmistokoodin suunnittelussa käytetään sekvenssi- ja aktiviteettikaavioita pohjana. Tavoitteena on ottaa tehtävistä (engl. tasks) CAN-tehtävä, kun CAN-dataa tulee työkoneesta. Tämä viedään CAN-vastaanotin kohtaan ja sieltä laitetaan se jonoon xQueueCreate() komennolla. Luetaan CAN-data UART0-sarjaportilta ja palautetaan se xQueueReceive() komennolla takaisin CAN-vastaanottimelle. Luettu CAN-data viedään bittimaskaukseen ja siellä maskataan se oikeaan muotoon. Maskattu arvo palautetaan CAN-tehtävälle, josta se lähetetään telematiikkayksikön MQTT serverille ja myös modeemeille ja takaisin. Tämä bittimaskattu arvo löytyy sen jälkeen MQTT serveriltä. Koko tämän ajan ollaan loopissa xCreateTask komennon alla. Tällä logiikalla tehdään koko ohjelmistokoodi toimimaan CAN-väylälle. Ohjelmistokoodissa tullaan käyttämään ESP32:n valmiita kirjastoja ja tarvittaessa muitakin kirjastoja helpottamaan ja nopeuttamaan ohjelmistokoodin tekoa.

3.3 Testaukset

CAN-väylän lähettävyys ja vastaanottavuus testattiin kahdella Fleethub Oy:n telematiikkayksikön laitteilla. Toiseen asennettiin lähettävä- ja toiseen vastaanottavakoodi, jossa lähettävä laite oli esimerkiksi trukki ja vastaanottava laite oli CAN-väylän lukija.

Opinnäytetyössä oleva laite on vain vastaanottava, koska lähettävä tulisi aina olemaan jokin työkone.

Testauksessa asennettiin laitteet paikoilleen ja aloitettiin testaamaan. Vastaanottava laite onnistui saamaan lähettävän laitteen CAN-datan kiinni ja tulostamaan sitä Bluetoothin monitoriin.

Sama testi tehtiin myös toisen kerran, mutta tällä kertaa käytettiin Ruptelan EasyCAN LCV5 adapteria, jolla pystyy mittaamaan CAN-väylän liikennettä induktiivisesti eli mittaamaan CAN-dataa häiritsemättä CAN-väylän liikennettä. Myös tällä kertaa vastaanottava laite sai CAN-dataa ja tulosti sitä Bluetoothin monitoriin, joten ohjelmistokoodi toimi molemmissa testeissä hyvin.

Onnistuneiden testausten ansiosta, mentiin testaamaan ohjelmistokoodia ja laitteistoa Hyundain sähkötrukkiin. Laitteet liitettiin kiinni trukin CAN-väylään, CAN-L- ja CAN-H-johtoihin ja ohjelmistokoodi onnistui löytämään CAN-väylän oikean taajuuden tähän trukkiin, joka oli tässä tapauksessa 250-kb/s. Induktiivisesti saatiin CAN-dataa ulos trukista Bluetoothin sovellukseen, jonka jälkeen todettiin testaus onnistuneeksi näillä laitteilla ja ohjelmistokoodilla.

4 OHJELMOINTI

Ohjelmoinnissa käsitellään testauksissa olevaa ohjelmistokoodia, josta näytetään oleellinen osa. Bittimaskauksen toiminnasta ja tarkoituksesta kerrotaan esimerkin avulla.

4.1 Bittimaskaus

Opinnäytetyössä käytettävä bittimaskaus tarkoittaa SAE J1939 -standardin raa-an CAN-datan muuntamisen käytettäväksi ohjelmistoihin. Näin saadaan erilaisten standardien mukaan CAN-tietoja ulos laitteista, esimerkiksi trukeista. Nämä ryhmät bittimaskaukselle löytyvät standardin SAE J1939 dokumenteista ja sieltä katsotaan ryhmät ja lasketaan niiden mukaan arvot eri tarkoituksille.

Esimerkkinä moottorin kierrosluku. Kierrosluvun ryhmät ovat PGN61444 (Kuva 8.) ja sieltä nähdään SPN, joka on SPN190 (Kuva 9.). CAN-väylältä saadaan silloin SAE J1939 standardin muotoisen viestin, joka on 64 bittiä. Tästä otetaan vasemmalta ensimmäiset 29 bittiä, joka on ”identifier” ja loput bitit ovat ”data field” eli SPN. PGN saadaan, kun bittimaskataan tästä identifierestä tarvittavat bitit eli ne ovat yhteensä kahdeksantoista bittiä alkaen yhdeksännestä bitistä. Maskauksessa käytetään AND-toimintoa eli ja-toimintoa PGN:n saamiseksi. Tämän jälkeen shiftataan eli siirretään PGN oikealle tarvittavan monta kertaa, jotta ylimääräiset nollat eivät häiritse lopputulosta eli tässä tapauksessa shiftataan kahdeksan bittiä oikealle. Saadaan desimaalina 61444 eli haluttu PGN (Kuva 8.).

Tämä sama operaatio tehdään myös SPN:n osalle. Dokumentistä katsottuna (Kuva 8.) tuossa SPN190:stä saadaan moottorin kierroslukuarvot tavuista neljä ja viisi. Bittimaskauksen jälkeen shiftataan taas oikealle ja tässä tapauksessa 24 bittiä eli 3 tavua. Desimaaliarvo on 4968. Dokumentistä selviää (Kuva 9.), että kierroslukunopeus lasketaan offset lisättynä rpm/bit, kerrottunu SPN arvolla. offset ja rpm/bit arvot löytyvät dokumentistä (Kuva 9.) eli tässä tapauksessa ne ovat 0 offset ja 0.125 rpm/bit. Kierrosnopeus on näillä arvoilla 621RPM.

PGN 61444 ***Electronic Engine Controller 1***

Engine related parameters

Transmission Repetition Rate:	engine speed dependent		
Data Length:	8		
Extended Data Page:	0		
Data Page:	0		
PDU Format:	240		
PDU Specific:	4	PGN Supporting Information:	
Default Priority:	3		
Parameter Group Number:	61444 (0x00F004)		

Start Position	Length	Parameter Name	SPN
1.1	4 bits	Engine Torque Mode	899
1.5	4 bits	Actual Engine - Percent Torque High Resolution	4154
2	1 byte	Driver's Demand Engine - Percent Torque	512
3	1 byte	Actual Engine - Percent Torque	513
4-5	2 bytes	Engine Speed	190
6	1 byte	Source Address of Controlling Device for Engine Control	1483
7.1	4 bits	Engine Starter Mode	1675
8	1 byte	Engine Demand – Percent Torque	2432

Kuva 8. PGN61444 tiedot dokumentissa. (SAE International 2011, 928)

SPN 190 ***Engine Speed***

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

Data Length:	2 bytes	
Resolution:	0.125 rpm/bit, 0 offset	
Data Range:	0 to 8,031.875 rpm	Operational Range: same as data range
Type:	Measured	
Supporting Information:		
PGN reference:	61444	

Kuva 9. SPN190 tiedot dokumentissa. (SAE International 2011, 50)

4.2 Lähettävä- ja vastaanottava ohjelmistokoodi

4.2.1 Lähettävä ohjelmistokoodi

Lähettävä koodi on CAN-väylän lähettävyyden testaukseen tehty ohjelmistokoodi, jota ei tulla käyttämään enää, ellei testauksia tehdä keskenään Fleethub Oy:n telematiikkalaitteilla, koska lähettävä laite on muuten jokin työkone. Koodi käyttää ESP32:n valmiita CAN-kirjastoja ja sarjaporttia arduinon omalla kirjastolla. Koodin setup-osiossa määritellään muuttujat ja portit laitteille sopiviksi, tässä tapauksessa paikat yksi ja kolme. CAN-väylän nopeudeksi määritellään 500kbs.

```
void loop() {
```

```
  CAN_frame_t tx_frame;
  tx_frame.FIR.B.FF = CAN_frame_std;
  tx_frame.MsgID = 0xFFF;
  tx_frame.FIR.B.DLC = 8;
  tx_frame.data.u8[0] = 0xFF;
  tx_frame.data.u8[1] = 0xFF;
  tx_frame.data.u8[2] = 0x00;
  tx_frame.data.u8[3] = 0xFF;
  tx_frame.data.u8[4] = 0x00;
  tx_frame.data.u8[5] = 0xFF;
  tx_frame.data.u8[6] = 0x00;
  tx_frame.data.u8[7] = 0xFF;
  ESP32Can.CANWriteFrame(&tx_frame);

  delay(2000);
}
```

Esimerkkikoodi 1. Lähettävän ohjelmistokoodin loop-osio.

Loop-osiossa (Esimerkkikoodi 1.) määritellään CAN-datan jokaiselle bitille omat arvot ja myös sen viestintunnuksellekin. Lopputuloksena lähettävä telematiikkalaite lähettää Bluetoothin kautta ohjelmistokoodin mukaista CAN-dataa kahden sekunnin välein, jonka vastaanottava laite yrittää vastaanottaa.

4.2.2 Vastaanottava ohjelmistokoodi

Vastaanottavassa laitteessa oleva ohjelmistokoodi käyttää myös Bluetoothia ja tulostaa lähettävän ohjelmistokoodin CAN-datan Bluetooth sovelluksen terminaaliin. CAN-väylän laite on vastaanottava laite, joka pystyy lukemaan työkoneiden CAN-dataa.

Koodin setup-osiossa määritellään muuttujat ja portit laitteille sopivaksi, jotka ovat samat kuin lähettävässä koodissa eli yksi ja kolme. Setup-osio on melkein samanlainen loppuosassa kuin lähettävä, mutta rakennetaan jono CAN-dataa varten eli xQueueCreate komennolla tehdään jono ohjelmistokoodiin. Koodissa myös alustetaan led, koska telematiikkayksikön laitteesta löytyy sellainen. Led on punainen, mutta kun se saa CAN-dataa, se vilkkuu sinisenä.

```
void loop(){


  CAN_frame_t rx_frame;
```

```
  unsigned long currentMillis = millis();

  // Seuraava CAN-data jonosta
  if (xQueueReceive(CAN_cfg.rx_queue, &rx_frame, 3 * portTICK_PE-
RIOD_MS) == pdTRUE) {

    rgb.setColor(0,0,20); // Sininen
    delay(200);
    if (rx_frame.FIR.B.FF == CAN_frame_std) {
      SerialBT.println("New standard frame");
    }
    else {
      SerialBT.println("New extended frame");
    }

    if (rx_frame.FIR.B.RTR == CAN_RTR) {
      Se-
rialBT.printf(" RTR from 0x%08X, DLC %d\r\n", rx_frame.MsgID,  rx_frame.F
IR.B.DLC);
    }
    else {

      for (int i = 0; i < rx_frame.FIR.B.DLC; i++) {
        SerialBT.printf("0x%02X ", rx_frame.data.u8[i]);
      }
      SerialBT.println(" \n");
    }
  }
  else {
    rgb.setColor(20,0,0); // Punainen
  }
}
```

Esimerkkikoodi 2. Vastaanottavan ohjelmistokoodin loop-osio.

Loop-osiossa (Esimerkkikoodi 2.) odotetaan, että saadaan CAN-dataa ja jos se saadaan, niin led vilkkuu sinisenä ja CAN-data tulostetaan Bluetoothin monitoriin. Sieltä löytyy CAN-data, jos sellainen on onnistuttu saamaan. Jos dataa ei saada, led pysyy punaisena. Datassa on kahdeksan CAN-bittiä ja myös niiden sanoma ja viestintunnus.

Ohjelmistokoodia muokataan, kun testataan työkoneeseen. Ohjelmistokoodiin lisätään välivaiheet, jossa telematiikkalaite käy 25kb/s välein kymmenen kertaa CAN-väylän taajuuksia läpi, kunnes oikea taajuus löytyy. Oikean taajuuden löytyessä tämä alkaa normaalisti tulostamaan CAN-dataa Bluetoothin monitoreihin.

5 YHTEENVETO

Projektin tavoitteena oli ohjelmoida CAN-väylälle prototyyppiohjelmisto, jolla pystyisi ottamaan CAN-dataa eri valmistajien työkoneista asiakkaiden seurattavaksi. Tämä prototyyppiohjelmisto toimisi Fleethub Oy:n telematiikkalaitteella, ja näin asiakkaat saisivat CAN-datan tiedon työkoneistaan helposti ulos. Opinnäytetyössä käsiteltiin laitteistoa ja ohjelmistokoodin suunnittelua ja ohjelmistokoodista käytiin läpi oleelliset asiat.

Ohjelmistokoodi testattiin, ensimmäiseksi telematiikkalaitteiden kesken ja toiseksi Ruptelan EasyCAN LCV5 -adapterilla ja nämä testaukset todettiin onnistuneiksi eli saatiin CAN-dataa ulos Bluetoothin kautta. Viimeinen testaus tehtiin Hyundain sähkötrukilla jolloin todettiin testaus onnistuneeksi, koska saatiin CAN-dataa ulos trukista. Testausten onnistumisten merkitys oli suuri, koska saatiin ratkaistua ongelma työkoneiden CAN-datan lukemisen kanssa, koska aikaisemmin ei ollut varmaa tietoa, että CAN-dataa saisi edes työkoneista ulos.

Opinnäytetyössä saatiin arvokasta tietoa projektin jatkokehitystä varten. Työtä tullaa jatkamaan todennäköisesti ohjelmistokoodin muuttamisella paremmaksi ja yksinkertaisemmaksi. Tulevaisuudessa on myös todennäköistä nähdä Fleethub Oy:n valmis CAN-dataa lukeva laite työkonemarkkinoilla.

LÄHTEET

Alanen, Jarmo. 2000. CAN-ajoneuvojen ja koneiden sisäinen paikallisväylä. Verkkodokumentti. VVT AUTOMAATIO. Viitattu 9.1.2021

https://docplayer.fi/10821742-Can-ajoneuvojen-ja-koneiden-sisainen-paikallisvayla.html

Kelechava, Brad. 2017. Controller Area Network (CAN) Standards. Verkkoblogi. Viitattu 1.3.2021

https://blog.ansi.org/2017/02/controller-area-network-can-standards-iso-11898/#gref

Wilfried, Voss. 2008. A Comprehensive Guide to J1939. Greenfield, MA, USA. Copperhill Media Corporation. Viitattu 1.3.2021

http://read.pudn.com/downloads344/ebook/1503048/A%20Comprehensi-ble%20Guide%20to%20J1939.pdf

OBD2 Explained – A Simple Intro (2021). CSS Electronics. Viitattu 9.1.2021

https://www.csselectronics.com/screen/page/simple-intro-obd2-explained/language/en

CANedge2: 2x CAN Bus Data Logger (SD + WiFi). CSS Electronics. Viitattu 10.5.2021

https://www.csselectronics.com/screen/product/can-lin-logger-wifi-canedge2/language/en#void

O'Connell, Neil. 2016. The CAN Bus dilemma: Will there ever be a standard? Verkkoblogi. Viitattu 1.3.2021

http://info.totaltraxinc.com/blog/the-can-bus-dilemma-will-there-ever-be-a-standard

Gustafsson, Robin & Blomqvist, Niklas. 2016. Options handling using external devices in forklift trucks. Opinnäytetyö. Viitattu 2.3.2021

http://www.diva-portal.org/smash/get/diva2:907584/FULLTEXT01.pdf

Espressif ESP32 Chip Function Block Diagram. 2018. Verkkodokumentti. Wikipedia. Viitattu 5.3.2021

https://commons.wikimedia.org/wiki/File:Espressif_ESP32_Chip_Function_Block_Diagram.svg

A Brief Introduction to the SAE J1939 Protocol. 2021. Verkkodokumentti. Copperhilltech. Viitattu 2.3.2021

https://copperhilltech.com/a-brief-introduction-to-the-sae-j1939-protocol/

Shawnnymel. 2020. Introduction to RTOS – Solution to Part 3 (Task Scheduling). Viitattu 15.3.2021

https://www.digikey.com/en/maker/projects/introduction-to-rtos-solution-to-part-3-task-schedu-ling/8fbb9e0b0eed4279a2dd698f02ce125f

FreeRTOS Queues. Verkkodokumentti. FreeRTOS. Viitattu 30.3.2021

https://www.freertos.org/Embedded-RTOS-Queues.html

Surface Vehicle Recommended Practice. 2011. Verkkodokumentti. SAE International. Viitattu 1.4.2021

Ruptela. 2021. Ruptela Transport Telematics. Viitattu 14.4.2021

https://www.ruptela.com/product/easycan/

Fleethub Oy. 2020. Fleethub Analytics. Viitattu 19.5.2021

https://fleethub.io/fi-fi/