Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Measurements

Master's Thesis

Wireless replacement of CAN based communication

Bc. Denys Chereda

Supervisor: Doc. Ing. Jiří Novák, Ph.D.
Study Programme: Cybernetics and robotics
Field of Study: Cybernetics and robotics
August 2020

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Chereda Denys** Personal ID number: **481749**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Measurement**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Wireless replacement of CAN based communication

Master's thesis title in Czech:

Náhrada CAN bezdrátovou komunikační technologií

Guidelines:

Identify a wireleless technology suitable to replace CAN communication in the cluster of interconnected gen-set controllers. Design the CAN to wireless Gateway module and implement its hardware and software. Demonstrate its functionality in a network of at least three controller nodes.

Bibliography / sources:

[1] ISO11898 standard - Controller Area Network
[2] Etschberger K.: Controller Area Network, IXXAT Automation 2001, ISBN: 978-3000073762
[3] Tanenbaum, A. S., Wethetral, D.J.: Computer Network. Prentice Hall 2010, ISBN-13: 978-0132126953

Name and workplace of master's thesis supervisor:

doc. Ing. Jiří Novák, Ph.D., K 13138 - katedra měření

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **20.02.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until:
by the end of winter semester 2021/2022

doc. Ing. Jiří Novák, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Acknowledgments

I would like to thank my supervisor, Doc. Ing. Jiří Novák, Ph.D., for supervising my thesis and providing consultation all the way through this thesis. I also would like to thank all my colleagues from Research and Development Department of ComAp a.s. for cooperation doting the development of this thesis. Namely Roman Taragel for providing technical support and ideas for system evaluation and Petr Krupansky for giving a bigger picture of product development procedure. Lastly, I want to thank my family and friends for a mental support during this thesis.

Declaration

I declare that the presented work was developed independently and I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, on August, 2020 .....

Abstract

The main goal of this thesis is to research, develop and test the solution that replaces CAN communication in the cluster of interconnected genset controllers or generally suitable for other machine-to-machine communication in such a way that it wouldn't require to introduce any changes in already existing devices. Firstly, a description of a use-case application is declared together with requirements for system design. At the next step system design is specified in terms of used HW, SW and networking model. Finally, testing and verification are done. The result of this work shows that the suggested solution can be applied to some extent and provides a base for further research in the development of the CAN-to-wireless gateway module as a commercial product.

Keywords: CAN bus, M2M communication, genset control system, Wi-Fi, Espressif Systems, ESP32

Abstrakt

Hlavním cílem této práce je výzkum, vývoj a testování řešení, které nahrazuje komunikaci CAN v klastru řadičů generátorů a je obecně vhodné i pro jinou komunikaci mezi stroji tak, aby nebyly nutné žádné změny v již existujících zařízeních. Nejprve je uveden popis aplikace spolu s požadavky na návrh systému. V dalším kroku je navrženo HW a SW řešení a také model síťové komunikace. Nakonec je provedeno testování a ověření. Výsledky této práce ukazují, že navrhované řešení může být s určitými omezeními použito a poskytuje základ pro další vývoj modulu bezdrátové brány CAN jako komerčního produktu.

Klíčová slova: sběrnice CAN, M2M komunikace, řídicí systém generátoru, WiFi, Espressif Systems, ESP32

Contents

List of figures

Abbreviations

AP Access point

BLE Bluetooth Low Energy

CAN Controller Area Network

CCMP Counter Mode Cipher Block Chaining Message Authentication Code Protocol an encryption protocol

CiA CAN in automation

CU Control Unit

ECU Engine Control Unit

EMC Electromagnetic compatibility

FPCC Future-Proof Co-existence Connectivity

GCB Generator Circuit Breaker

GPIO General purpose input-output

HID Human interface device

IDF IoT Development Framework

ISM Industrial, Scientific and Medicine

ISO International Organization for Standardization

ISR Interrupt Service Routine

JSON JavaScript Object Notation

LMK Local Master Keys

MAC Medium access control

MCB Mains Circuit Breaker

MCU Microcontroller Unit

PDR Packet Delivery Ratio

PMK Primary Master Key

RSSI Received signal strength indication

SAE Society of Automotive Engineers

SDK Software development kit

TCP Transmission Control Protocol

WHN Wireless Hosted Network

WLAN Wireless Local Area Network

Chapter 1

Introduction

In today's world, human civilization demand on energy and information transfer is growing at a huge rate to provide more goods, increase the quality of life and establish a more stable society. But even with a lot of innovations in all branches of technology, there are still many cases where we rely on combustion engines to generate electricity for power plants, remote communication points, backup energy sources, off-grid or marine applications. Therefore, industrial power generation via gensets, a unit that combines a combustion engine with a generator, is also growing and evolving to find more efficient and reliable solutions and so the control systems for their use are becoming more advanced. [1]

1.1 Controller Area Network overview

1.1.1 History note

In the past, automotive manufacturers were connecting electronic devices in the vehicles using point-to-point wiring systems and with growth in the number of such devices, those systems become heavy and expensive wire harnesses that are hard to maintain. At the beginning of 1980s engineers at Robert Bosch GmbH were evaluating existing systems for serial bus communication in the scope of their use for the automotive industry and passenger cars especially but found out that non of those available solutions were able to fulfil engineer's requirements therefore in the 1983 Uwe Kiencke started the development of a new serial bus system.

In the early stage of specification development engineers from Mercedes-Benz were involved and as potential integrated circuit manufacturer Intel showed an interest in the new serial bus system. Professor Dr. Wolfhard Lawrenz from the Ostfalia University of Applied Science, Germany, was hired as a consultant and gave the new network protocol name "Controller Area Network"

At the SAE (Society of Automotive Engineers) congress in Detroit, February of 1986, the new bus system was presented as "Automotive Serial Controller Are Network". Uwe Kiencke together with Siegfried Dais and Martin Litschel introduced the multi-master network protocol that was based on a non-destructive arbitration mechanism that grants access to the bus for the frame with the highest priority without any delays. The mentioned father of CAN together with Bosch employees Wolfgang Borst, Wolfgang Botzenhard, Otto Karl, Helmut Schelling and Jan Unruh has developed several error detection mechanisms like automatic disconnecting of faulty bus nodes in order to continue communication between the remaining nodes. Another difference from existing in that time bus system was that the transmitted frames were not identified by addresses of the receiver/transmitter node but rather by their content. The initial presentation was followed by many more papers and

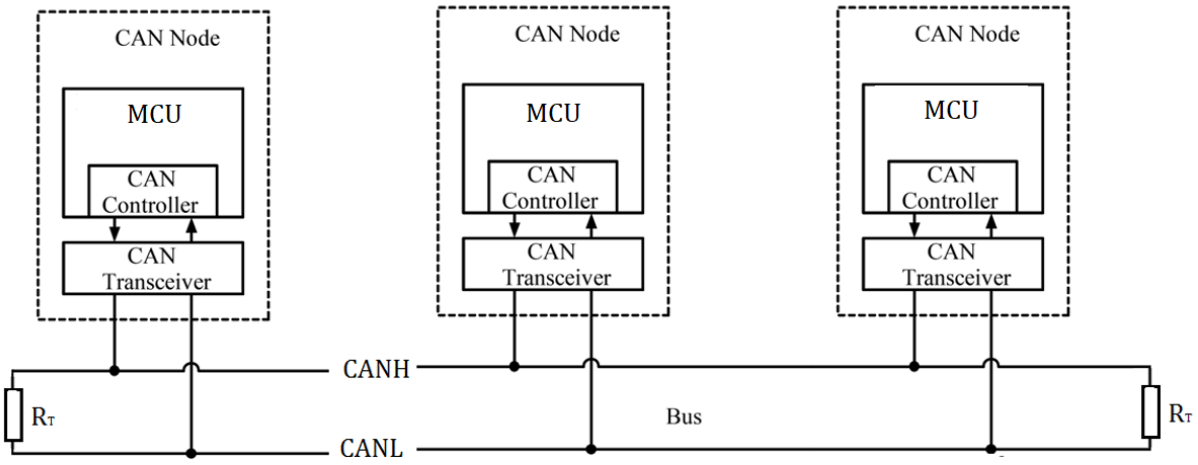
publications that were describing this innovative communication protocol and in the mid of 1987 Intel delivered the first CAN controller chip the 82526, in a few years an idea had become reality. Shortly after that Philips Semiconductor introduced the 82C200 chip.

In 1991 Bosch published CAN 2.0 specification that has two parts: part A is for a standard frame format with an 11-bit identifier, and part B with an extended 29-bit identifier. In 1993, the International Organization for Standardization (ISO) published the CAN standard 11898 which was later restructured in a few parts: ISO 11898-1 - covers the data-link layer, 11898-2 - covers physical layer for high-speed CAN, 11898-3 - describes physical layer for low-speed, fault-tolerant CAN. While Bosch CAN 2.0 specification is freely available, the ISO 11898-2 and ISO11989-3 can be purchased from the ISO.

For around 20 years after CAN 2.0 and ISO 11898 Bosch continued to develop CAN and in 2012 the CAN FD 1.0 or CAN with Flexible Data-Rate was released. This specification incorporates a different frame format that allows to the extent the payload from 8 to 64 bytes with an optional switching toa faster bit rate but still compatible with existing CAN 2.0 networks. The third generation of CAN is also on the way. In the end of 2018, the CiA started the development of CAN XL - the third generation of CAN-based data link. [2]

1.1.2 CAN network architecture

A CAN network consists of several CAN Nodes which are linked via a two-line physical transmission medium called CAN Bus. Each CAN Node consists of CAN Transceiver - a unit that acts as an interface between physical CAN bus and CAN Controller and the MCU/CAN Controller itself.



CANH - CAN high line
CANL - CAN low line
R_T - termination resistance

Figure 1-1. CAN network architecture

The two lines of the CAN bus, CANH and CANL, are pair of twisted wires over which symmetrical signal runs - meaning that signal on the CANH is always inverted to signal on CHNL and vice versa. In such a way electromagnetic disturbance caused by motors, switching contacts or ignitions systems are minimized. Also, such configuration improves EMC (electromagnetic compatibility).

Besides initial design application in vehicles, nowadays the CAN bus communication is heavily used in an even more variety of scenarios, like agriculture machinery, industrial automation and machine control, aviation, medical devices, etc, due to its robustness, cost of implementation and bus architecture. But a global movement to the so-called wireless era and other constraints like the convenience of use and maintenance, the necessity of maintaining additional redundancy layer to repeat CAN bus in some critical use-cases are causing motivation for this research.

1.2 Control systems of industrial genset

Each genset by itself is a quite complex electromechanical system with build-in ECU (engine control unit) but to achieve more automated and robust control an additional control unit (CU) is introduced. Such CU is monitoring and controlling the engine parameters by directly communicating to its ECU via CAN bus and continues monitoring of generator parameters such as voltage, current, phase together with generator/mains circuit breakers (GCB/MCB) states control.

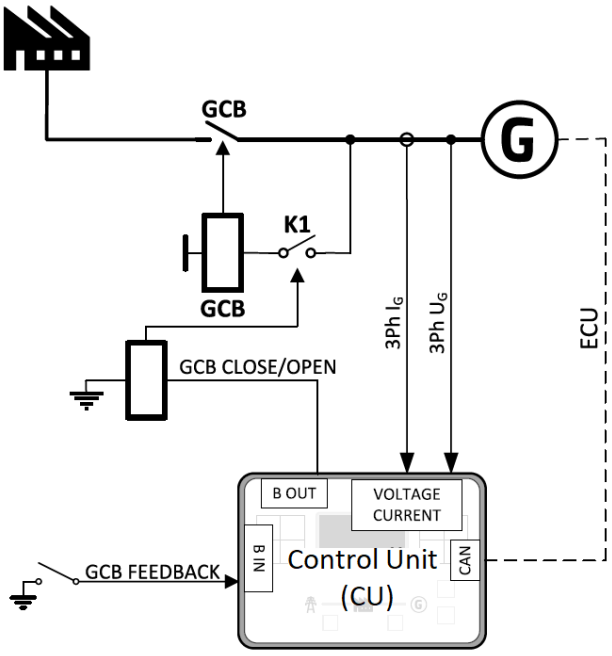


Figure 1-2. Connection between genset and CU

But in many cases, there is a necessity for multiple gensets to work synchronously on one bus due to the requirement of total power distribution or to have a backup reserve in case of any unit shutdown

or to have a communication between genset CUs and mains CU for example in applications where gensets are standby measure.

1.3 Power management

The Power management function decides how many gensets should run and selects particular gensets to run. The power management is applicable in cases multiple gensets run in parallel to mains or in the island operation. The function is based on the load evaluation in order to provide enough available running power. Since it allows the system to start and stop gensets based on the load demand, it can vastly improve the system fuel efficiency. In other words, an additional genset starts when the load of the system rises above a certain level. The additional genset stops when the load of the system drops down below a certain level. The process of determining genset start and stop is done in each controller; there is no "master-slave" system.

Therefore, the system is very robust and resistant to failures of any unit in the system. Each of the controllers can be switched off without influencing the whole system. Except for the situation, the respective genset is not available for power management. The power management evaluates the so-called load reserve. The load reserve is calculated as the difference between the actual load and a nominal power of running gensets. The reserve is calculated as an absolute value (in kW / kVA) or relative to the nominal power of genset(s) (in %). Priority swapping function focuses on an efficient run of genset in regard to running hours and genset size. [3]

One of the typical application scenarios with power management is the following: consider a plant that is connected to mains power on one hand and to the cluster of gensets on the other. The plant power consumption from mains is limited to e.g. 200 kW due to one of the reasons like overall mains capabilities or significantly higher cost for each kWh after some level. But on some occasions plant can draw up to 500 kW therefor three gensets with a nominal power of 100 kW each are connected in parallel to mains. So, depending on the current power consumption, available load reserve and other configurable parameters starting or stopping sequence is initiated. Detailed relation between those values and gensets starting/stopping sequences is shown in Figure 1-3

1.4 CAN-based intercontroller communication

When each unit in the system knows about the current state of all other units the possibilities for advanced control emerges. This CAN-based machine-to-machine communication line provides a foundation for a complex automation system that manages power management function.

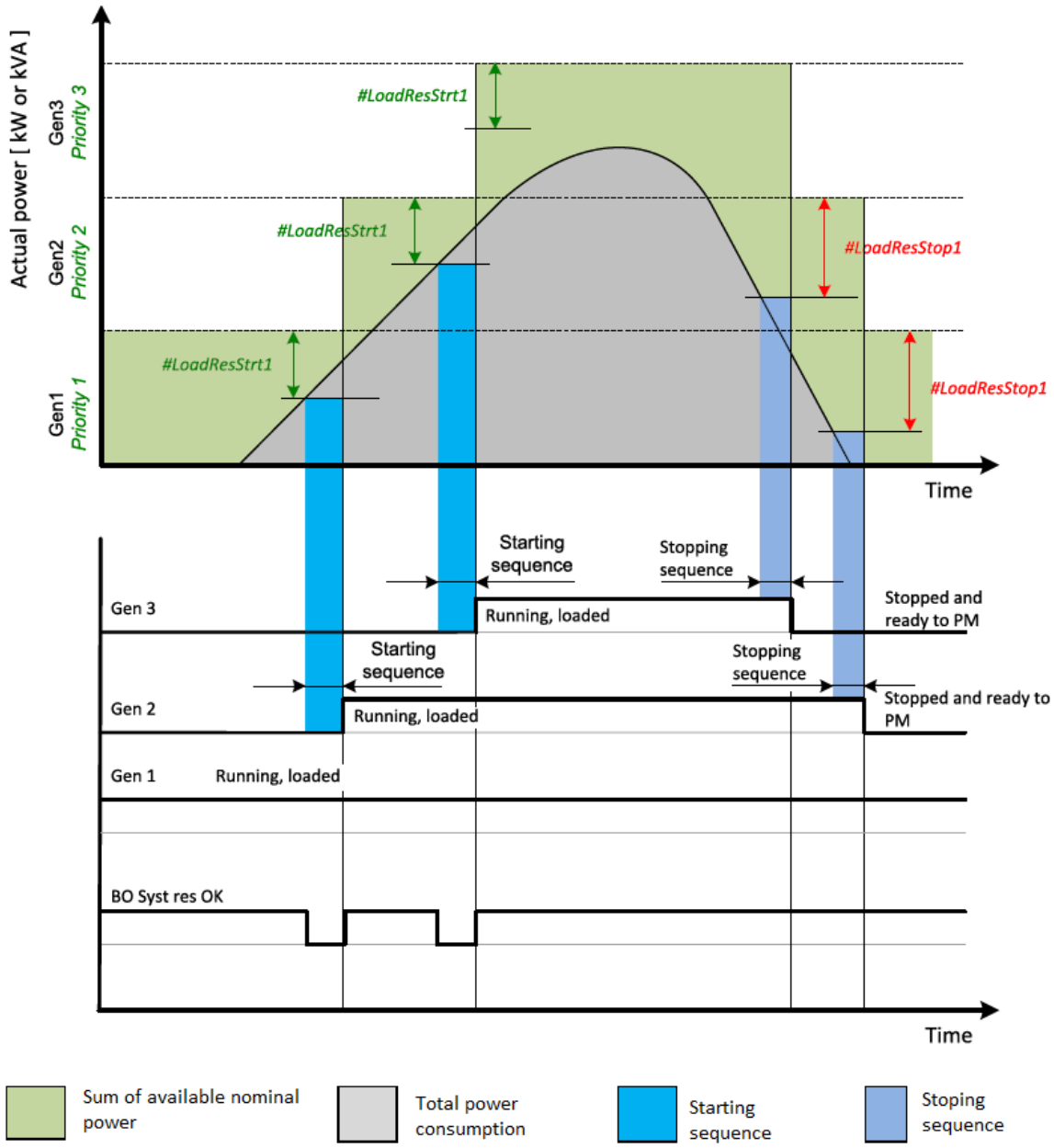


Figure 1-3. Gensets start/stop diagram in power management [3]

According to the details of applications such as a number of connected CUs, the distance between them and the relationship between distance and number of connected CUs, the requirements about communication speed and acceptable latencies may vary.

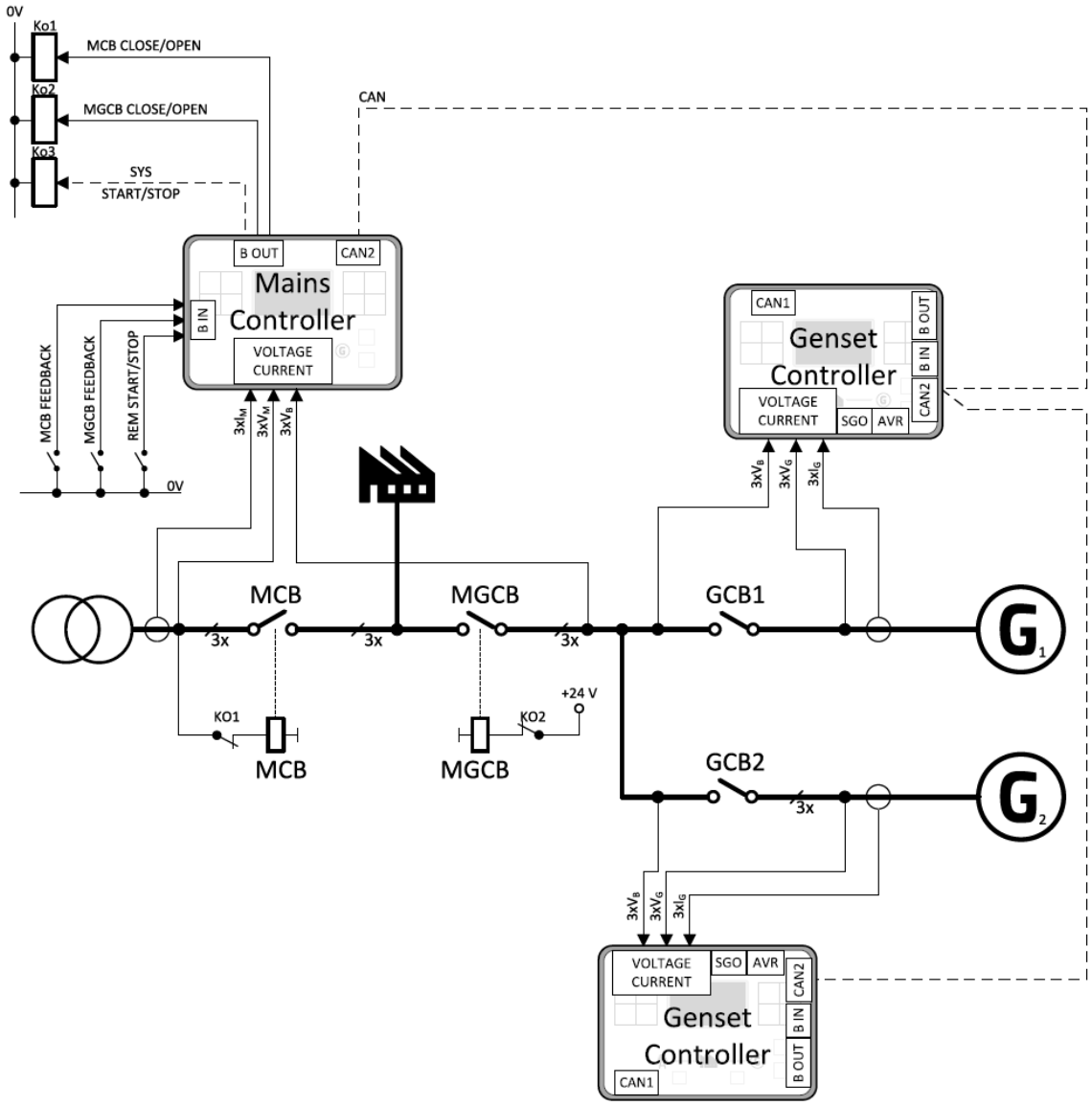


Figure 1-4. Example of system with intercontroller communication [3]

At the limit, such intercontroller communication capable to work with 64 CUs at the same time. For such a case the communication speed is 250 Kbit/s, the maximum length of a segment is 200 m and the overall bus length can be up to 800 m with a repeater. Each controller sends a heartbeat (PQ) message every 30-60 ms which is also the latency for recognizing the presence of CU on the common bus. More common and simpler situations like 16 CUs, the communication speed can be 125 Kbit/s, the maximum length of a segment is 400 meters and overall bus length up to 1600 meters with a repeater.

It is also worth mentioning that quite a typical CAN connection on the site can be done inside one switchboard or between few evenly spaced genset blocks.

Figure 1-5. Typical installations

1.5 Intercontroller communication protocol

The backbone of this communication consists of 4 messages that are sent and evaluated from each CU, those messages and their periods are:

- PQ data: current active and reactive power production, genset status (ready, running, loaded etc.), priority in power management. The period can vary from 30 to 60 ms.
- Nominal data message: nominal power ratings of a particular unit (P, Q, I) with a period of 5s
- Group data: group status, group position in the whole system (left/right relative group) sent every second or in response to changes in circuit breakers state
- Run hours time data message with period of 10 s

Each controller in the system sends one frame (ext. ID + 8 bytes) of PQ data with a period $T = 2 \times \frac{1}{f}$ where f is a nominal frequency of 50/60 Hz or in some cases period is predefined in the range 30-60 ms. At the same time, each controller awaits this frame from other controllers for $2 \times T$. If the frame does not arrive, the sending controller is considered as disconnected and the bit that corresponds to this CU address is set to 0 in the CAN16/32 register. This is the most crucial message for stable power management process control. Therefore, in average one node is generating from 30 to 40 messages per second.

At this stage, this communication is implemented via the CAN 2.0 specification but considering current trends about CAN FD, the design of the wireless system aimed to be future-proof at least in terms of the main enhancement of CAN FD - increase of maximum payload length from 8 bytes to 64.

1.6 The requirement specification for system design

Saying it simple - the main idea is to create CAN-to-wireless gateway modules with a network configuration such that when the CUs are equipped with those modules their behaviour is still the same as prior. Therefor following requirements for most common use cases can be specified:

- all gateway modules are equivalent similar to the CAN bus
- a working area of 100 meters circle in diameter
- communication speed up to 1 Mbps
- up to 16 CU communication
- latency up to 60 ms

As a proof of concept, we assumed a demonstration of a working network with 3 CUs equipped with the gateway modules.

1.7 CAN-to-wireless gateways on the market

It is worth mentioning that the price of the module is also playing the role. Besides the fact that the gateway is planned to be as a mass-produced unit, there is no sense to have an extension with a price comparable to the price of CU (starting from 150EUR) that is equipped with it or even to a half of that.

By a simple Google search “can to wireless” a few devices can be found:

1.7.1 GCAN-211 Wi-Fi to CAN from Shenyang Guangcheng Technology Co., Ltd

The first issue that is raised is the price of 118USD but the main problems, in this case, are limitation of configuration and applications that were considered in the design, two use-cases are mentioned from the manufacturer webpage: [4]

- Build a connection between Pad, mobile phone or PC and CAN bus network. The user can receive and transmit data to CAN bus wirelessly
- Using a pair of GCAN-211 can realize CAN bus wireless relay, with one node configured as AP and another as a client(station)

Therefore, this solution can act as a bridge between not more than two CAN buses.

1.7.2 HD67644-Wi-Fi-B2 from ADFweb.com S.r.l

In this case, the price is not available directly on the manufacturer's website. Looking into the user manual, "EXAMPLE OF CONNECTION" part, the situation similar to the previous gateway is observed. It is suggested that the module is used as an Ethernet bridge between two CAN buses or as a Wi-Fi monitor/logger of the CAN bus. [5]

1.7.3 CAN-Wi-Fi WIRELESS CAN INTERFACE from Grid Connect,Inc.

Besides the price of 150USD the device datasheet from the product webpage describes that there are 2 Wi-Fi modes available: [6]

1. Wi-Fi access point (server) when in the field and using a mobile PC
2. Wi-Fi client for remote monitoring and diagnostics

The CAN Wi-Fi can be utilized in several ways:

- Implemented as a service tool for diagnosing CAN networks and uploading new CAN parameters.
- Employed as a wireless development tool.
- Embedded permanently into a mobile device for 24/7 wireless remote monitoring.

Still not applicable for the purpose of this research.

1.7.4 PCAN-Wireless Gateway from PEAK-System Technik GmbH

Ignoring the price of 300EUR this gateway looks more promising. Going into the user manual of product that is available on a product webpage, chapter 5.5.2 WLAN describes such operational modes: [7]

- Infrastructured Mode (Client): The Gateway must connect to a WLAN network of an existing access-point.
- Ad-Hoc Mode: The Gateway provides a WLAN network of its own. Other devices can connect to this network as an equal participant. The SSID entered in the following form is used as the network's name. This operation mode does not require an additional access-point. Note: Since Windows 8.1, the connection to Ad-Hoc networks is no longer supported.
- Micro Access Point Mode (Host): The PCAN-Gateway hosts a WLAN network of its own. Up to seven devices can connect as a client. The SSID entered in the following form is used as the network's name. This operation mode does not require an additional access-point.

From this description, it looks feasible to create a network of up to 8 nodes with Micro Access Point Mode, which is best seen so far, but yet still this configuration will rely on one node acting as a host for the network.

1.7.5 Kvaser BlackBird v2

The Kvaser AB is known for a variety of devices that are working with CAN bus so two devices int their portfolio are related to so to say wireless CAN bus: Kvaser Air Bridge Light HS [8] and Kvaser BlackBird v2. While the Air Bridge Light HS provides a functionality similar to already mentioned gateways - wireless connection of two separate CAN buses, the BlackBird v2, on the other hand, provides more options for configuration and the product webpage leads directly to the blog article "How to connect multiple Kvaser BlackBirds" [9]. From this post, it is seen that multiple BlackBirds can be paired together in infrastructure mode, WHN or through Wi-Fi direct and even so it isn't designed in the first place to provide the desired system but more focused for the monitoring/development purposes with a connection to PC.

This particular unit looks the most featured and feasible of providing the required network, but the price point of 814 EUR makes it not only unreasonable for application of interest but even costly to try out.

Figure 1-6. CAN-to-Wi-Fi gateways on the market

1.7.6 LumenRadio AB

LumenRadio AB is a company that provides a wireless networking solution that may suit the purpose of this research, their description says: [10]

“We develop, market and sell low-power and ultra-reliable operating systems, radio modules and products for wireless IoT-applications. From high-profile professional lighting to heavy-duty industrial equipment, we have a solution that works in environments where others fail.

Based on our patented ultra-reliable Cognitive Coexistence technologies, we offer radio modules and an operating system that can be integrated straight into your products. Using our technology your products will be Future-Proof as they co-exist along other wireless technologies, not being interfered or causing interference – we call this FPCC (Future-Proof Co-existence Connectivity).”

To be more specific, the MiraOS is claimed to provide a reliable and scalable mesh network with a free application layer that can be utilized for the purpose of the target use-case.

Mira offers unrivalled scalability with thousands of nodes in a self-healing and self-organizing network. Mira utilizes LumenRadio's patented coexistence technology to provide maximum robustness to RF disturbances and minimum effect on other wireless networks.

The main blocker of trying out this solution is the cost, MiraOS is a commercial product and the evaluation kit price is 2000 USD.

Chapter 2

System design

The focus of this chapter is to go through all choices regarding overall system structure, HW components used, SW and networking technology that were utilized to achieve the desired result.

2.1 System overview

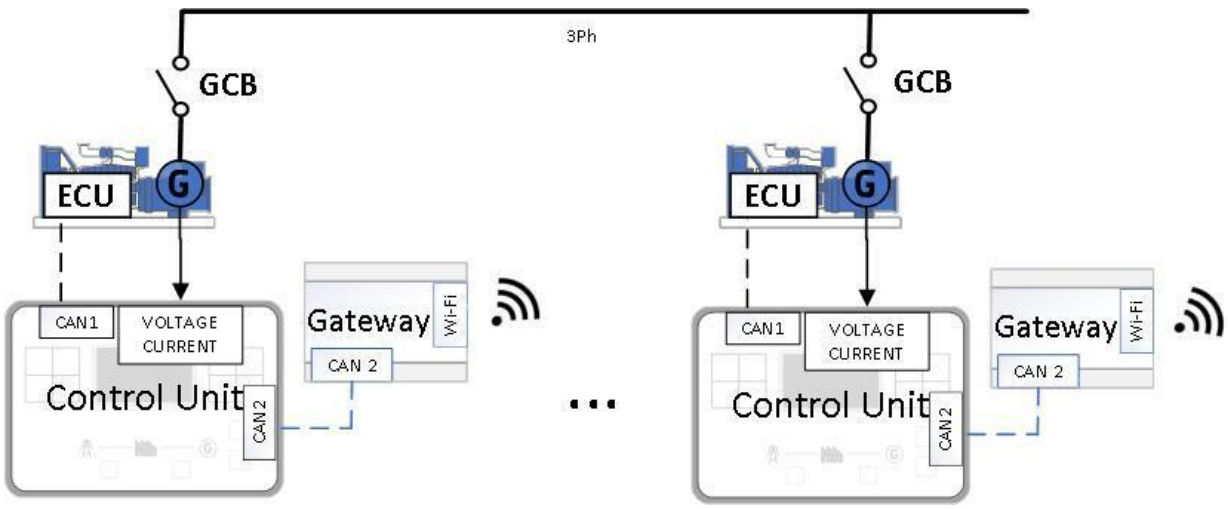


Figure 2-1. System concept

The system is consisting of gateway modules and the network produced by them. This system serves as a wireless bridge between the CUs in which intercontroller communication is required. The input data is taken from the CAN2 port of CU and then transferred to all nodes in the network. The moment gateway receives a packet it converts it to CAN message and put it to CAN2 port of the corresponding CU.

2.2 Search for suitable technology

Our current progress in a variety of different ways to transfer data is astonishing - water, air, metal or plasma as a medium for signal, long-range and low power communication points around the whole world or transfer rates up to hundreds of Gbits in devices of a consumer electronics market.

A variety of existing technologies or their combinations can cover almost any application that humans can think about therefore in this work we are going to use well defined and standardized technology. According to known data about speed, throughput, and common working ranges [11] [12] a rough comparison graph can be drawn.

Another important point that should be considered is the architectural details that each technology can support. In wireless networks there are two modes for configuring a wireless architecture – ad-hoc and infrastructure [13] wherein the ad hoc mode devices transmit directly peer-to-peer while in infrastructure mode devices communicate through an AP.

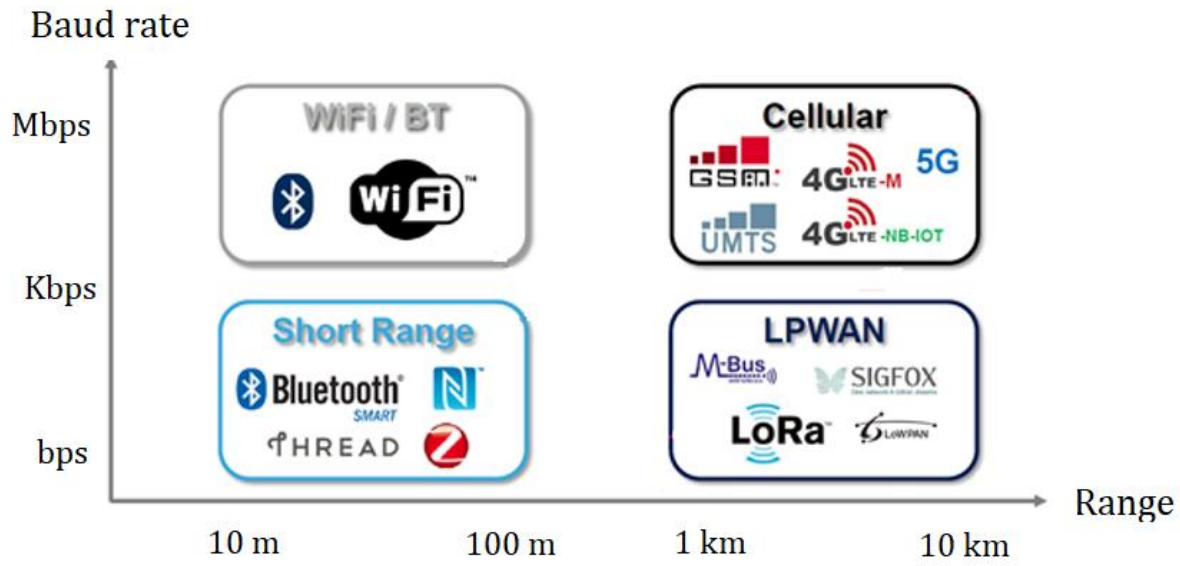


Figure 2-2 . Wireless technologies overview

Considering the requirements about speed and range we can cut off the bottom part of the graph in Figure 2-2, so we left with BLE, Wi-Fi and cellular. The next point to consider is the level of autonomy that can be reached in each of the technology. While the cellular networks are quite good in many parameters, like throughput and latency, the main problem is relying on the cellular network provider because there are applications where genset control systems are working in a very remote location with bad or even no cellular network coverage. Also, cellular communications are based on infrastructure mode of network architecture which makes the overall system more complex and expensive. BLE technology is similar to Wi-Fi in many cases and even provides some advantages like lower power consumption but the overall range that can be reached is greater in Wi-Fi.

Summing up, for the purpose of this project two protocols based on Wi-Fi would be tested to evaluate the difference and find the most suitable solution.

2.3 Gateway HW structure

While it is possible to find a proper combination of Wi-Fi modules together with simple MCU without Wi-Fi to reduce the overall cost of the device but on the other hand with more parts chances of faults are growing. Luckily nowadays we have easy access to all possible sorts of MCUs so it is no problem to find MCU with CAN and Wi-Fi/BLE onboard for a reasonable price therefore as a brain of the gateway the development board based on ESP32 SoC from Espressif Systems [14] is chosen. To be more specific, the development board based on ESP32-WROOM-32 module is taken, the main specs of the module are listed below:

- CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 240 MHz and performing at up to 600 DMIPS
- Memory: 520 KiB SRAM
- Wi-Fi: 802.11 b/g/n (802.11n up to 150 Mbps), Frequency range 2.4 GHz ~ 2.5 GHz

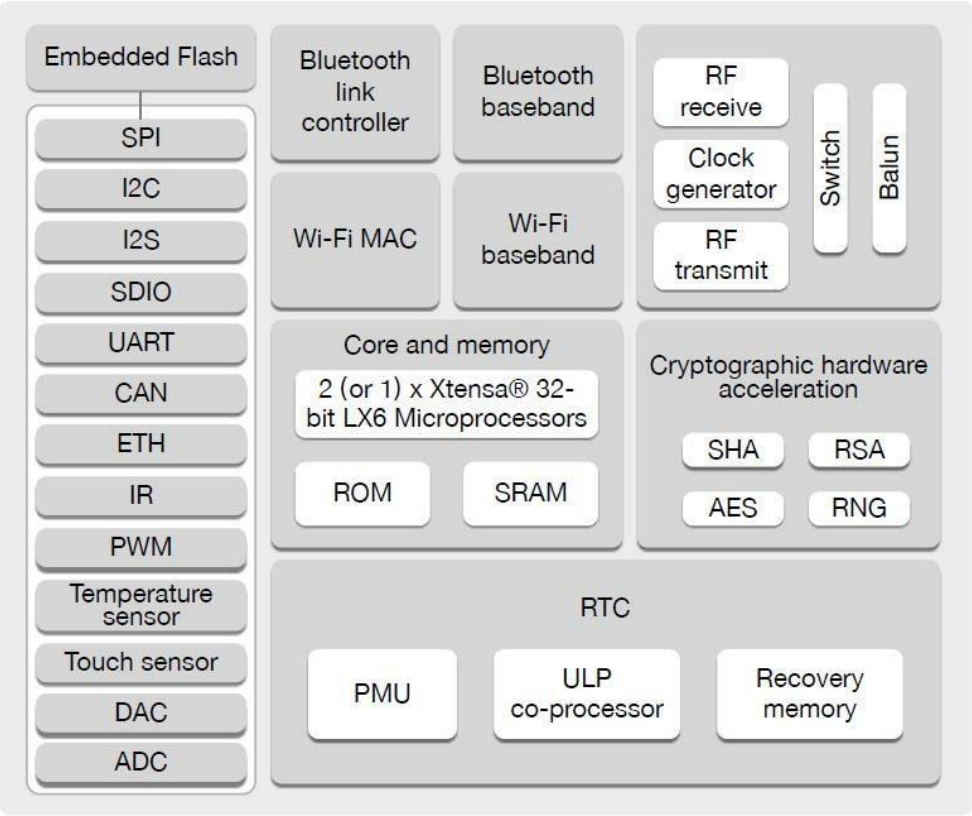


Figure 2-3. ESP32 SoC functional block diagram [14]

The block scheme of gateway module is on the Figure 2-4. To connect the physical CAN bus with the ESP32 (U1) the CAN transceiver module based on TJA1050 chip (U2) is used. Such a module with all passive components that are necessary for the proper functionality of IC and pin headers already installed increases the speed of the development. The only inconvenience about this particular solution is voltage level differences, the ESP32 isn't 5V tolerant so the 5V-3.3V level shifter based on

BSS138 (U3) chip is used to step down signals from CAN bus physical driver. Definitely, another CAN transceiver that operates on 3.3V logic levels, like SN65HVD230, can be used for this purpose but mentioned TJA1050 modules together with level shifter were in the disposal.

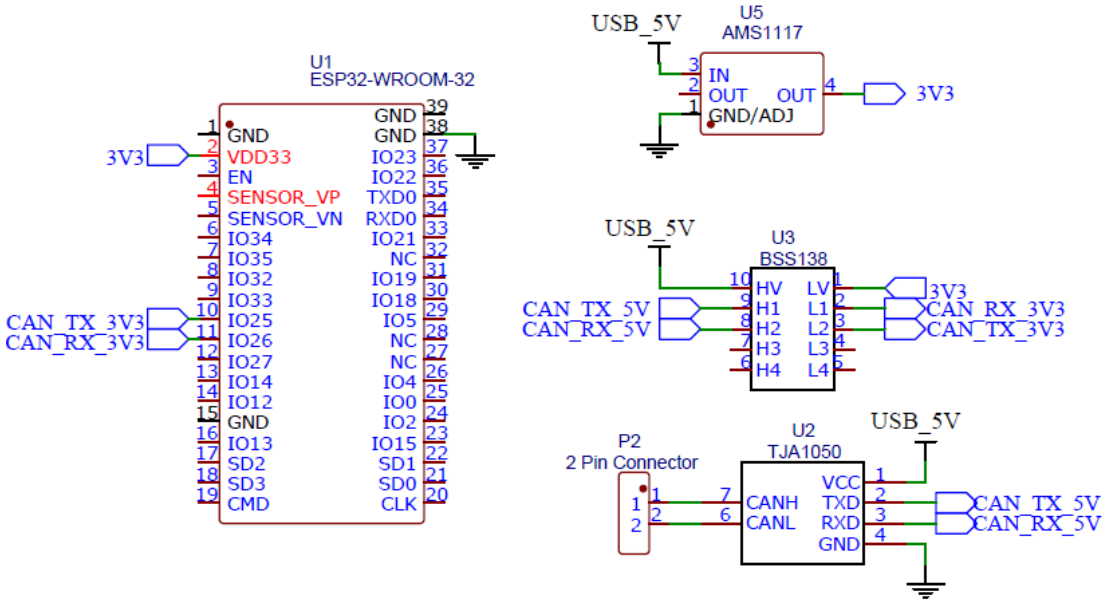


Figure 2-4. Gateway schematics

A breadboard and jumper wires are used to create a platform for the whole gateway and to provide ease of repair and debug. On the figure below photo of the complete gateway assembly is provided. The development board can be power from the micro-USB port and build-in AMS1117 (U4) voltage regulator providing the 3.3 V reference for the level shifter.

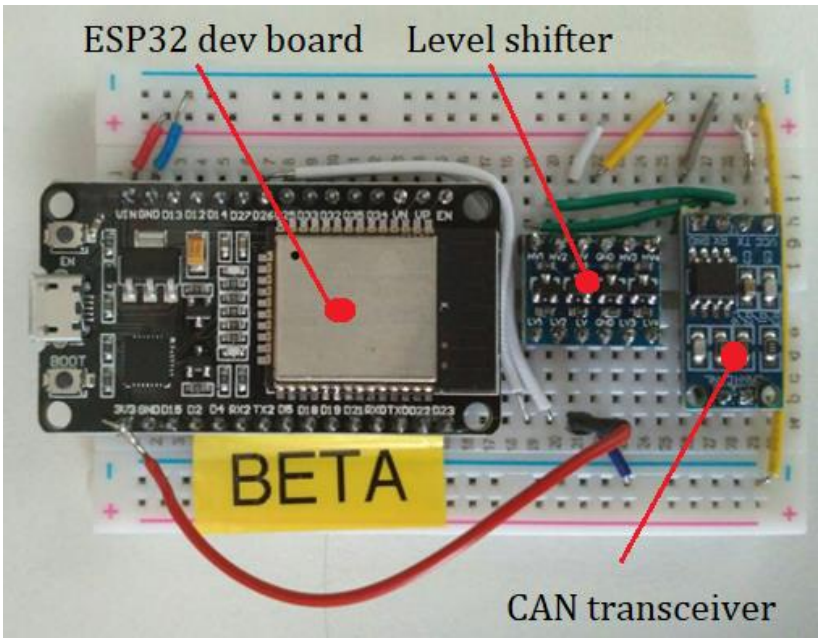


Figure 2-5. Gateway prototype

2.3.1 Gateway cost

Taking as a reference price from e-shop Techfun s.r.o [15] a cost of gateway module is following:

- ESP32 Node MCU 9.1 EUR
- TJA1050 CAN bus transceiver 1.7 EUR
- BSS138 based level shifter 0.8 EUR
- Breadboard 1.9 EUR

In total – 13.5 EUR.

2.4 SW structure

The implementation specifics of a particular protocol may vary but knowing the target application we can assume such a working principle. Firstly, after booting up the initialization process taking place configuring the peripherals for CAN and network parameters such as peering the nodes or detecting other nodes with proper SSID and connect to them. In case of some problems with these steps, error handling is taking place by log record of the issue or running self-repair procedure. If everything succeeded properly the main application is taking place.

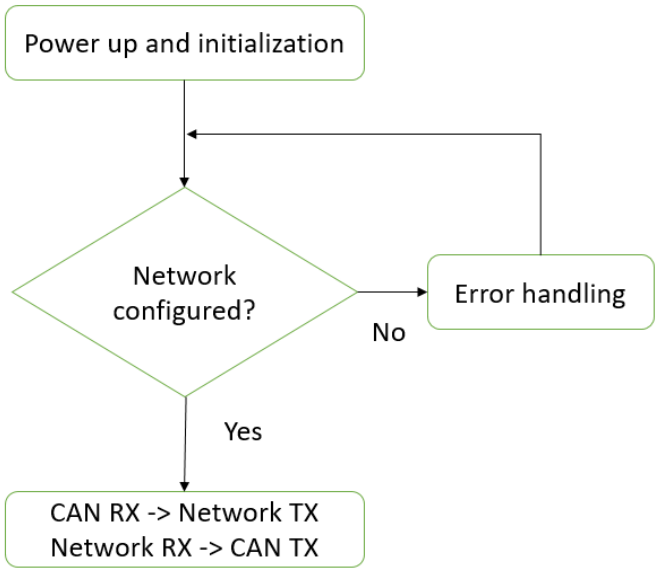


Figure 2-6. Software operation principle

2.4.1 Development environment

Espressif Systems SDK, called ESP-IDF (IoT Development Framework) is the official development framework for the ESP32 and ESP32-S Series SoCs. Natively ESP-IDF requires quite complicated procedure for correct functioning but luckily for all makers and enthusiasts Espressif Systems provides an Arduino core for the ESP32 and this work will be based on that.

The light-weight operating system FreeRTOS [16] is used to operate multiple threads of the target application. The RTOS kernel scheduler switches between tasks, making it appear as if each task had been executing simultaneously. This functionality will be used for to have the Wi-Fi task operating the network, while the CAN task continuously pools messages from the CAN network and also convenient for some testing purposes when strict periodicity is needed.

2.4.2 ESP32 CAN driver

The initial release of Espressif Systems SDK for ESP32, has been published without a proper CAN drive. Luckily, Thomas Barth, a German PhD student, developed a library for such a driver and published in his own blog Barth Development [17].

There are four main parts:

1. API Configuration. Here the structures of the frame information and frame itself together with main routines declarations, such as CAN_Init(), CAN_write_frame(*frame), CAN_stop are defined

2. The configuration part contains all the settings needed for proper initialization of the CAN driver: the available CAN speed values, the ESP32 pins that are used for connection with a CAN transceiver and the FreeRTOS queue are declared.

3. Register definitions of the actual CAN Port installed on the chip. They are assigned in a way similar to definitions mentioned in SJA1000 standalone CAN controller datasheet [18]

4. Functions and routine part contains all the necessary functions to initialize and communicate through the CAN bus together with Interrupt Service Routine (ISR)

- CAN_init().Before calling this function the configuration structure can_device_t must be filled with respective values. CAN_Init() function sets the CAN clock bit and deactivating the reset bit the CAN module is enabled, after that the TX and RX pins are assigned to GPIOs specified in can_device_t. Then all the necessary registers are set for respective value, like BTR1.B.TSEG1 that is dependent on the Baud Rate. Afterall MOD.B.RM is set to 0 forcing to exit the reset mode and start message receive/transmit.
- CAN_isr(void *arg_p). The Interrupt Service Routine is enabled. For this application, the only interrupt of interest is the one associated with the reception flag and every time a message is received the CAN_read_frame() function is called.

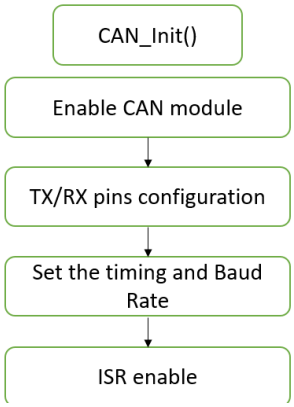


Figure 2-7. CAN driver initialization

- CAN_read_frame() function copies the message received into the queue. At the end of this routine CMR.B.RRB register is set to 1 saying to the hardware that the message has been received and read correctly. The RXFIFO occupied by the message is released so the next incoming message sets the reception flag and function is called again.
- CAN_write_frame(const CAN_frame_t* p_frame). The p_frame argument is an actual frame that required to be sent via the CAN bus. The argument must be compline with the CAN_frame_t structure defined in the API. At the end of the transmission, the CMR.B.TR register is set to 1 letting the hardware know that a new frame can be transmitted.
- CAN_stop(). The MOD.B.RM register is set to 1 forcing the CAN module to enter the reset mode.

2.5 Networking

2.5.1 PainlessMesh

The painlessMesh [19] library was evaluated in this project. All nodes of the network are interconnected and equal and the network itself is self-organizing and self-repairing with a tree topology. The JSON format is used for messages between nodes for ease of understanding and processing.

Each node behaves as an AP and client at the same time to provide connection and be connected. All nodes which are not already or anymore connected to an AP periodically scans for available APs around it and connects to the one with the strongest signal but considering the list of already present connections or sub-connections. Such procedure, connecting only to unknown APs, creates the mesh without network loops so there is only a single route between each pair of nodes. Every node is aware of the complete network topology. Messages between different nodes are sent in JSON format, making them easy to understand and produce.

The mesh network gives two types of messages to work with - a single(unicast) and broadcast. For the purpose of this project, single messages aren't a type of interest because from its name - it is the messages between two specific nodes only. On the other hand, the broadcast messages are moving iteratively through all connections, so all nodes are exchanging data with all other nodes. On the below JSON schema of single addressed and broadcast message is shown. Both are virtually identical with only difference that for broadcast type is set to 8 and the destination is equal to the receiving node id.

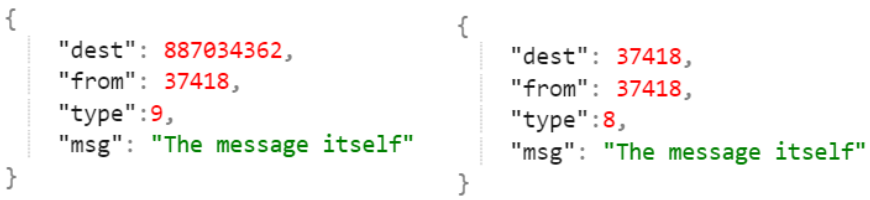


Figure 2-8. painlessMesh message JSON scheme, single(left) and broadcast(right)

After including the library, the painlessMesh object must be created. For this project following steps with the corresponding API functions are going to be used.

2.5.1.1 PainlessMesh initialization

void painlessMesh::init(String ssid, String password, uint16_t port = 5555, enum nodeMode connectMode = STA_AP, _auth_mode authmode = AUTH_WPA2_PSK, uint8_t channel = 1, phy_mode_t phymode = PHY_MODE_11G, uint8_t maxtpw = 82, uint8_t hidden = 0, uint8_t maxconn = 4)

This function is called in the setup() of Arduino sketch initializing the mesh network. This routine does the following things.

- Starts a wifi network
- Begins searching for other wifi networks that are part of the mesh
- Logs on to the best mesh network node it finds… if it doesn't find anything, it starts a new search in 5 seconds.

ssid = the name of mesh that is shared for all nodes. They are distinguished by BSSID. password = wifi password for mesh. port = the TCP port that you want the mesh server to run on. Defaults to 5555 if not specified. connectMode = switch between AP_ONLY, STA_ONLY and STA_AP (default) mode also used in this project.

2.5.1.2 Updating the mesh

void painlessMesh::update(void)

This function must be added to the loop() of Arduino sketch. This routine runs various maintenance tasks and without it mesh functionality disrupts. For a proper performance sketch should be free of any delay() function calls.

2.5.1.3 Receiving the messages

void painlessMesh::onReceive(&receivedCallback)

Sets a callback routine for any messages that are addressed to this node. Callback routine has the following structure.

void receivedCallback(uint32_t from, String &msg)

Every time this node receives a message, this callback routine will the called. "from" is the id of the original sender of the message, and "msg" is a string that contains the message. For the purpose of this project the callback function takes received data and sends it to the CAN bus.

2.5.1.4 Message broadcast

bool painlessMesh::sendBroadcast(String &msg)

Sends msg to every node on the entire mesh network and returns true if everything works, false if not. Prints an error message to Serial.print, if there is a failure.In case of taget application it is called when there is a frame in the rx queue of CAN bus.

2.5.2 ESP-NOW

ESP-NOW is a kind of connectionless Wi-Fi communication protocol defined by Espressif System, which enables multiple devices to communicate with one another without using Wi-Fi AP or stations. The protocol is similar to the low-power 2.4GHz wireless connectivity that is often deployed in wireless HIDs. So, the pairing between devices is needed prior to their communication. After the pairing is done, the connection is safe and peer-to-peer, with no handshake being required.

In ESP-NOW, application data is encapsulated in a vendor-specific action frame and then transmitted from one Wi-Fi device to another without connection. CTR with CBC-MAC Protocol(CCMP) is used to protect the action frame for security. ESP-NOW is widely used in smart light, remote controlling, sensor, etc.

ESP-NOW uses a vendor-specific action frame to transmit ESP-NOW data. The default ESP-NOW bit rate is 1 Mbps. The format of the vendor-specific action frame is as follows: [20]

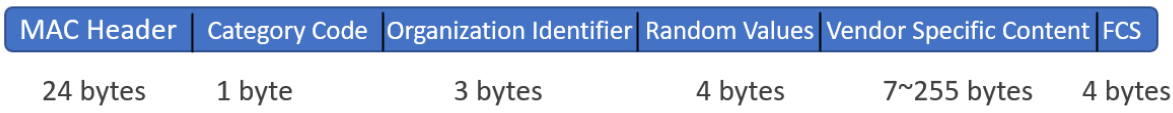


Figure 2-9. ESP-NOW frame

- Category Code: The Category Code field is set to the value (127) indicating the vendor-specific category.
- Organization Identifier: The Organization Identifier contains a unique identifier (0x18fe34), which is the first three bytes of MAC address applied by Espressif.
- Random Value: The Random Value filed is used to prevents relay attacks.
- Vendor Specific Content: The Vendor Specific Content contains vendor-specific fields as follows:

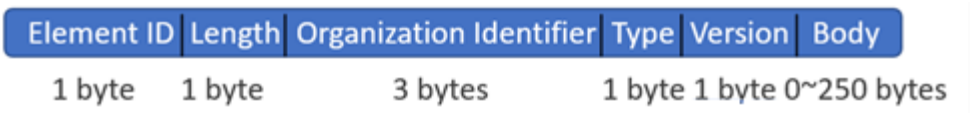


Figure 2-10. Vendor Specific Content

- Element ID: The Element ID field is set to the value (221), indicating the vendor-specific element.
- Length: The length is the total length of Organization Identifier, Type, Version and Body.
- Organization Identifier: The Organization Identifier contains a unique identifier(0x18fe34), which is the first three bytes of MAC address applied by Espressif.
- Type: The Type field is set to the value (4) indicating ESP-NOW.
- Version: The Version field is set to the version of ESP-NOW.
- Body: The Body contains the ESP-NOW data

As ESP-NOW is connectionless, the MAC header is a little different from that of standard frames. The FromDS and ToDS bits of the FrameControl field are both 0. The first address field is set to the destination address. The second address field is set to the source address. The third address field is set to broadcast address (0xff:0xff:0xff:0xff:0xff:0xff).

For this project following steps with the corresponding API functions are going to be used.

2.5.2.1 Initialization and De-initialization

esp_now_init() is called to initialize ESP-NOW and esp_now_deinit() to de-initialize ESP-NOW. The ESP-NOW is based on the Wi-Fi therefore before transmitting or receiving any data through this protocol it is recommended to firstly start the Wi-Fi and only after that call esp_now_init(). When esp_now_deinit() is called, all of the information on configuration like paired devices, channel, encryption settings will be deleted.

2.5.2.2 Add Paired Device

esp_now_add_peer() is called to add the device to the paired device list and it must be done before data send to this device, . The maximum number of paired devices is twenty, also if the encryption is enabled the Local Master Keys (LMK) must be set before function call. The ESP-NOW data can be send via the Station and the SoftAP interfaces, so the interface must be set in advance before data transmission. A device with a broadcast MAC address must be added before sending broadcast data. The range of the channel of paired devices is from 0 to 14. If the channel is set to 0, data will be sent on the current channel. Otherwise, the channel must be set as the channel that the local device is on.

2.5.2.3 Send ESP-NOW Data

esp_now_send() is called to send ESP-NOW data and esp_now_register_send_cb() to register sending callback function. In sending callback function return value may be ESP_NOW_SEND_SUCCESS what is representing that the data was successfully received on the MAC layer, or otherwise it will return ESP_NOW_SEND_FAIL and a few reasons may cause a fail in ESP-NOW data send. For example, the destination device does not exist; the channels of the devices are not the same; the action frame is lost when transmitting on the air, etc. It is not guaranteed that the application layer can receive the data. If necessary, send back ack data when receiving ESP-NOW data. If receiving ack data timeouts,

retransmit the ESP-NOW data. A sequence number can also be assigned to ESP-NOW data to drop the duplicate data.

For situations where there is a lot of ESP-NOW data to send, esp_now_send() is called several times to send less than or equal to 250 bytes of data once a time. Note that too short interval between sending two ESP-NOW data may lead to disorder of sending callback function. So, it is recommended that sending the next ESP-NOW data after the sending callback function of the previous sending has returned. The sending callback function runs from a high-priority Wi-Fi task. So, do not do lengthy operations in the callback function. Instead, post the necessary data to a queue and handle it from a lower priority task.

2.5.2.4 Receiving ESP-NOW Data

esp_now_register_recv_cb() is called to register receiving callback function. Call the receiving callback function when receiving ESP-NOW. The receiving callback function also runs from the Wi-Fi task. So, do not do lengthy operations in the callback function. Instead, post the necessary data to a queue and handle it from a lower priority task.

2.6 Security constraints

For now, it is not that clear about the required level of protection needed to not create an overly complicated system. The painlessMesh network itself works on specific SSID, port and password to access it so we can assume this as one level of protection. Also, the CAN data is structured according to the ComAp protocol for intercontroller communication which is not publicly available and when it is sent via mesh it is also base64 encoded.

ESP-NOW is capable of utilizing the CCMP method, which is described in IEEE Std. 802.11-2012, to protect the vendor-specific action frame. The Wi-Fi device maintains a Primary Master Key (PMK) and several LMKs. The lengths of both PMK and LMk are 16 bytes.

- PMK is used to encrypt LMK with the AES-128 algorithm
- LMK of the paired devices is used to encrypt the vendor-specific action frame with CCMP method. The maximum number of different LMKs is six.

Chapter 3

System evaluation

This section will describe all procedures that were made to evaluate system performance and limits.

3.1 Proof of concept

As was mentioned in the beginning as a proof of concept considered 3 CUs that are communicating with each other. For this purpose, the ComAp InteliGen 500 and two InteliGen 200 are used. When the CU is recognizing another CU on the CAN2 bus the bit corresponding to CU address is set to 1 in CAN16/CAN32 register.

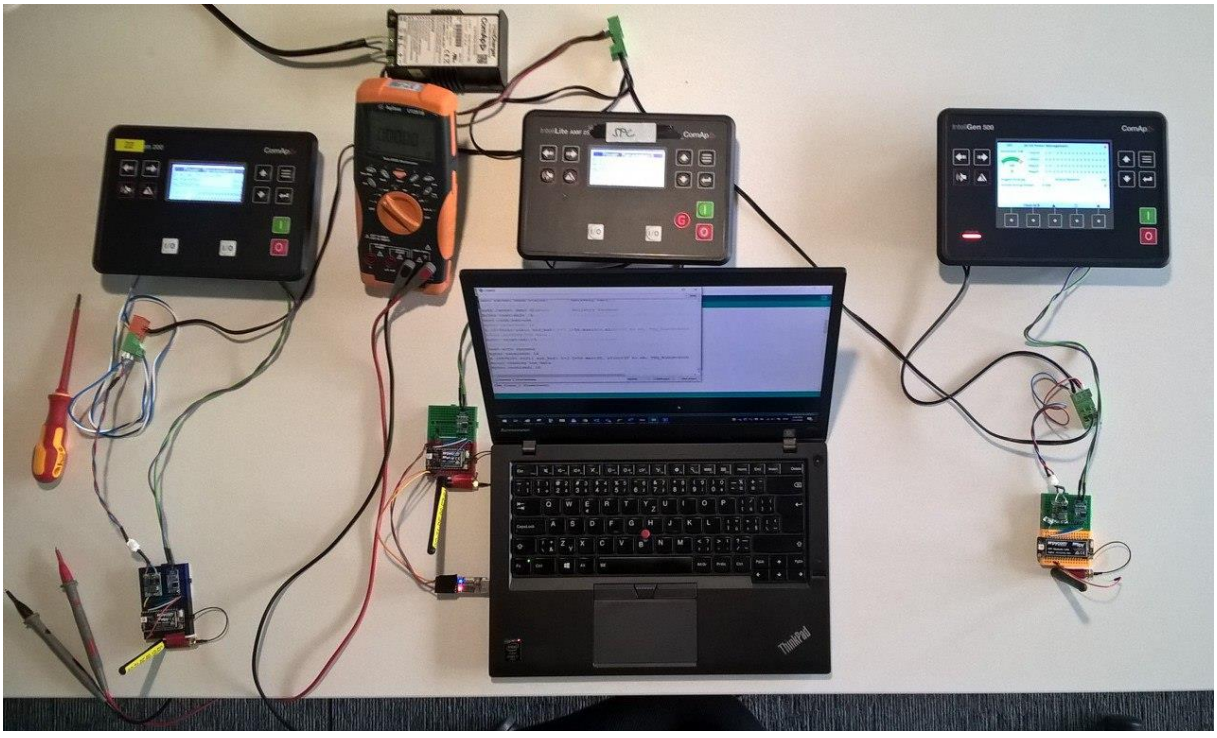


Figure 3-1. Initial test with 3 CUs

Both painlessMesh and ESP-NOW showing that controllers can recognize each other but some visually noticeable communication losses are happening while using the painlessMesh.

Through the observation of changes in CAN16 register on CU screen and monitoring the debug output of painlessMesh some conclusions can be made. For the first few minutes, some sporadic

communication issues are happening with average time around 1 second but after that real problems occur. The gateway starts to cycle: network selfheal with reconnection taking place around 10 seconds, after that proper communication is established but only for some time and it loops. Looking into debug log corresponding messages are seen and probably the reason for problems is seen too. Networking task reports ERROR : addmessage(): message queue full. The library documentation quote about caveats and limitation says:

"Try to be conservative in the number of messages (and especially broadcast messages) you sent per minute. This is to prevent the hardware from overloading. Both esp8266 and esp32 are limited in processing power/memory, making it easy to overload the mesh and destabilise it. And while painlessMesh tries to prevent this from happening, it is not always possible to do so."

After a deeper look into painlessMesh project documentation and forum discussions it can be said that the target application is goes beyond capabilities of this library, therefore overall SW and HW operational principles are working properly but suggested realization of networking model isn't sufficient enough.

On the other hand, when gateways were operating with ESP-NOW no visible problems were seen on the CU screen so further tests to estimate networking performance quality will be done using this protocol.

3.2 System performance

3.2.1 Messaging throughput

To have an estimation about maximum messaging rate that system can handle operating under ESP-NOW following tests were performed.

Firstly, the time that needed for a message broadcast itself is measured in a periodically executed task Figure 3-2. The output of serial monitor from two different boards provides almost identical result, value in microseconds, "Send with success" monitor output ensures that the message was transmitted successfully on the MAC layer:

12:19:11.096 → 247
12:19:11.096 → Sent with success
12:19:12.090 → 249
12:19:12.090 → Sent with success
12:19:13.085 → 250
12:19:13.085 → Sent with success
12:19:14.083 → 250
12:19:14.083 → Sent with success
12:19:15.083 → 248
12:19:15.083 → Sent with success

12:19:16.096 → 246
12:19:16.096 → Sent with success
12:19:17.090 → 251
12:19:17.090 → Sent with success
12:19:18.084 → 250
12:19:18.084 → Sent with success
12:19:19.090 → 251
12:19:19.090 → Sent with success
12:19:20.083 → 248
12:19:20.083 → Sent with success

Therefore, it can be count that on average 250 microseconds are needed to broadcast the message and so theoretically maximum messaging rate can be 4000 messages per second.

To see if the message rate has an influence on communication quality or more explicitly packet delivery ratio (PDR) a simple test can be performed. The idea is following, both boards will send to

another one a counter variable that is incremented for each message send. The receiving side in the receive callback will evaluate a continuity of counter value from another message and if there is a loss it will be evaluated with output to the serial monitor.

```
void sendMsg( void * parameter )
{
  while(1){
  prevus = micros();
  esp_err_t result = esp_now_send(0, (uint8_t *) &cnt, sizeof(cnt));
  currentus = micros();
  interval = currentus - prevus;
  Serial.println(interval, DEC);
  if (result == ESP_OK) {
  Serial.println("Sent with success");
  }
  else {
  Serial.println("Error sending the data");
  }
  cnt++;
  vTaskDelay(1000);
  }
}
```

Figure 3-2. sendMsg() task

By changing the value of vTaskDelay() function a different messaging rate is provided. To avoid mess in the serial monitor output only the “Packet loss”, error messages together with counter of losses and total number of messages were shown and then counted for evaluating of PDR.

```
// Callback when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&tmp1, incomingData, sizeof(tmp1));
  if (tmp1 - tmp0 != 1)
  {
    Serial.println("packet loss");
    loss++;
    Serial.println(tmp1, DEC);
    Serial.println(loss, DEC);
  }
  tmp0 = tmp1;
}
```

Figure 3-3. Message receive callback

The FreeRTOS minimal time resolution is 1 millisecond, therefore the test was performed for half of hour with periods of 50, 10, 5 and 1 milliseconds and results shows that PDR for 50 is 100%, for 10 – 99.9999%, for 5 – 99.9991%, for 1 – 99.0457%.

It is worth mentioning that during this test a dependency to Wi-Fi channel overall load was noticed, so the results above were observed when the channel was set to the least occupied one.

Another qualitative test for establishing the limit of networking technology together with application layer of CAN bus processing can be performed. For this purpose, two InteliCompact^{NT} CUs with a special SW, named IC-Contr-SIM, will be equipped with gateway modules. This SW was specially designed to simulate up to 32 CUs on the intercontroller communication line, meaning that it can be used to iteratively add another package of frames that is representing another CU on the bus and in the same time it also shows a presents of other CUs. When each of the unit is generating frames to simulate 8 CUs, first one is responsible for CUs with addresses 1 to 8 and the second one – 9 to16, no noticeable losses are observed on their screens, so in total system is handling up to 16×30=480 messages per second. When the 18th CU is added on the bus noticeable changes are starting to take place – the bits representing presence are jittering.

3.2.2 Application test with the MultiKit

MultiKit is a system that consists of 3 genset CUs and one mains CU with peripherals that can produce real voltages and currents to properly simulate many real-world applications, especially with power management function enabled.

In this test 3 genset (next G1, G2 and G3) CUs and mains CU are equipped with gateway modules, the

Figure 3-4 shows the connection of power lines between gensets, plant and mains as also the CAN2communication bus which is now replaced by the wireless system. The mains CU acts as a “master” meaning that it can turn Power Management functionality on/off (Sys start/stop) via intercontroller communication line.

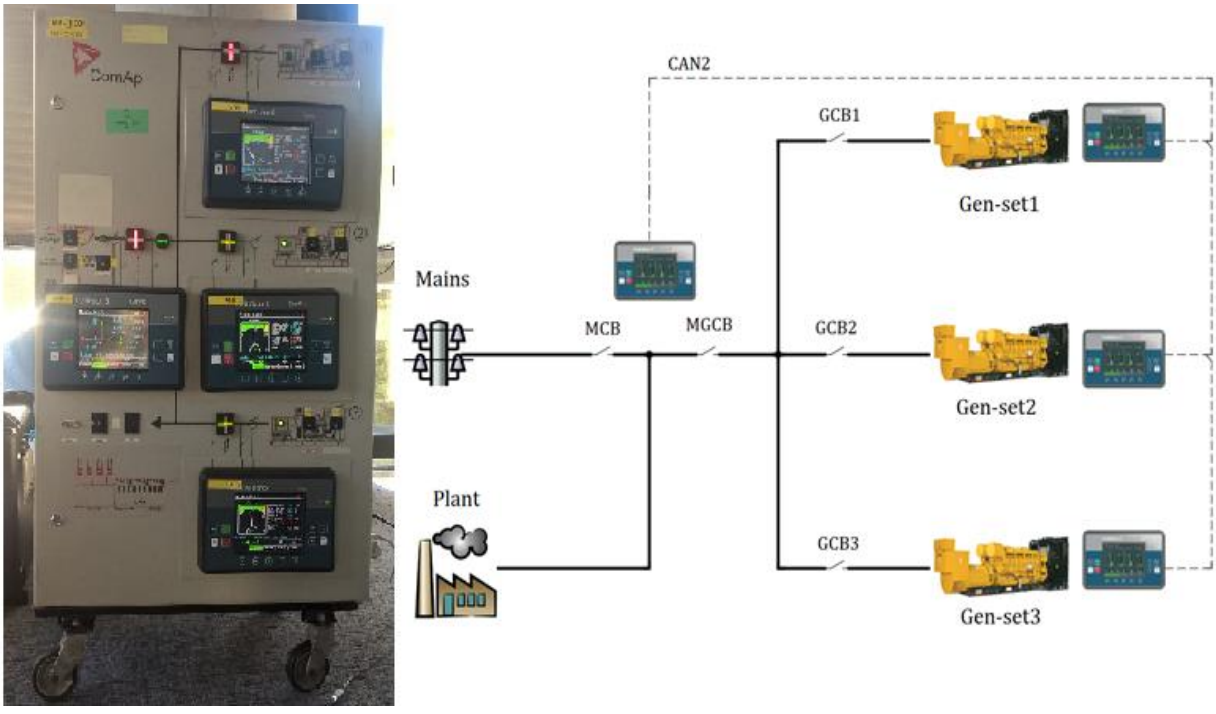


Figure 3-4. The MultiKit and application test scheme

The application test scenario is similar to one that is described in section 1.3 on page 4.

To ensure that system behaves according to requirements and for ease of possible troubleshooting, those values are logged and monitored:

- LBI (logical binary input) Sys S/S
- ActPwrRes
- LoadResStart, LoadResStop
- Engine State (Ready, Running, Loaded, etc.)
- CAN16

The WinScope PC tool - ComAp Controllers' Monitoring Software is used to log this data. On the Figure 3-5 the screenshot of a scope from CU with address 3 is shown.

Initial conditions:

- no load on the bus
- G1 running in idle, GCB1 closed
- G2 and G3 are in Ready state meaning that they are capable to start the engine and supply power to the bus when it will be required

For the engine to start two conditions should be fulfilled: LBI Sys S/S is activated and actual power consumption reducing the ActPwrRes less than #LodResStart(1-3) of the corresponding genset, so at stage 1 in Figure 3-5, the starting sequence of G3 is initiated, after synchronization procedure GCB3 is closed and genset is loaded providing the required amount of power to the plant. In 4 minutes, the reverting process is taking place and G3 returns to Ready state, and back to Loaded after the same amount of time. After half an hour load demand is rising and the starting sequence of G2 is induced (stage 2).

After the GCB2 is closed the amount of power required from gensets is divided between them according to settings. In this case, all gensets are configured as equal therefore the load is also split in the same way. In the last 20 minutes of recording a few Start/Stop sequences are performed and then Power Management system is shut down via shared Sys S/S signal. Actual scopes from all CUs are provided in the attached Appendix B - Contents of attached CD on page 49, screenshots of those for CU1 and CU2 are seen in Appendix A - Application test logs on page 47.

Using a PCAN-View software and USB-CAN interface from PEAK-System [21] the load on network can be evaluated. The monitoring tools shows that following frames are present on a bus Figure 3-6. Besides the frames that were described in the section 1.5 on page 7 and can be easily recognized from the period, there are also present extra frames that a responsible for modules of shared binary inputs and outputs – such module provides information to all CUs even when the source of those signal is one particular controller. Summing up, the bus is loaded with approximately 130 messages per second.

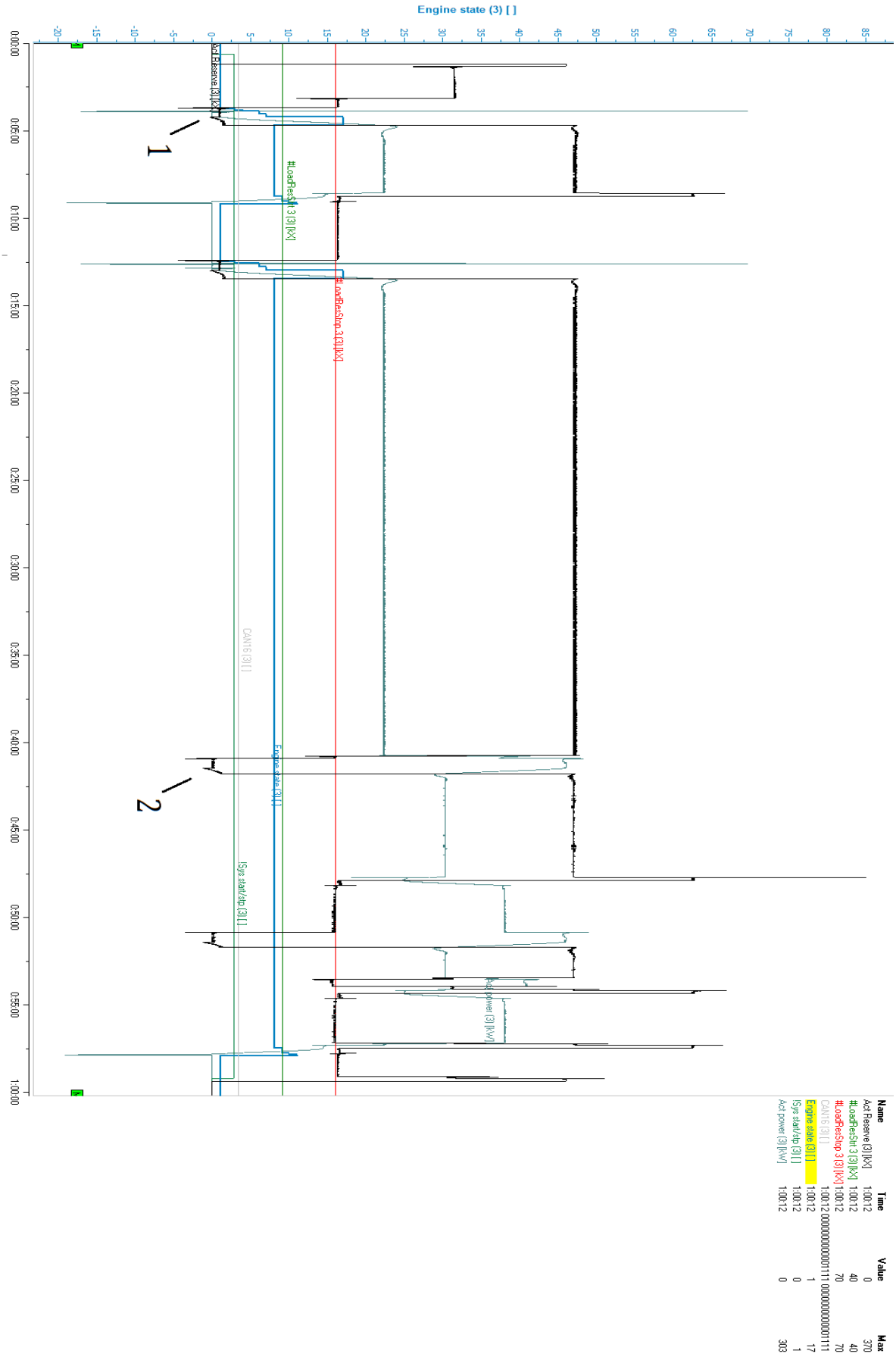
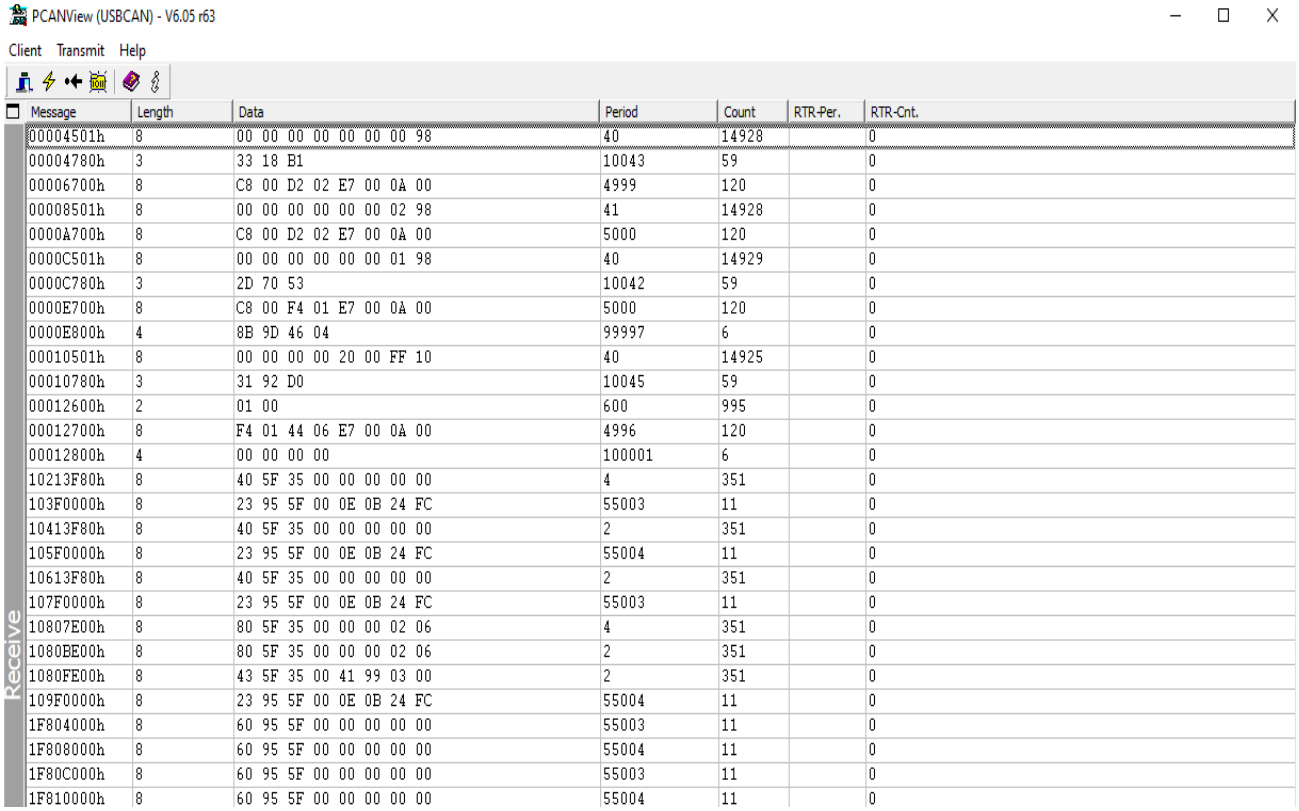


Figure 3-5. Application test values graphs



Message	Length	Data	Period	Count	RTR-Per.	RTR-Cnt.
00004501h	8	00 00 00 00 00 00 00 98	40	14928		0
00004780h	3	33 18 B1	10043	59		0
00006700h	8	C8 00 D2 02 E7 00 0A 00	4999	120		0
00008501h	8	00 00 00 00 00 00 02 98	41	14928		0
0000A700h	8	C8 00 D2 02 E7 00 0A 00	5000	120		0
0000C501h	8	00 00 00 00 00 00 01 98	40	14929		0
0000C780h	3	2D 70 53	10042	59		0
0000E700h	8	C8 00 F4 01 E7 00 0A 00	5000	120		0
0000E800h	4	8B 9D 46 04	99997	6		0
00010501h	8	00 00 00 00 20 00 FF 10	40	14925		0
00010780h	3	31 92 D0	10045	59		0
00012600h	2	01 00	600	995		0
00012700h	8	F4 01 44 06 E7 00 0A 00	4996	120		0
00012800h	4	00 00 00 00	100001	6		0
10213F80h	8	40 5F 35 00 00 00 00 00	4	351		0
103F0000h	8	23 95 5F 00 0E 0B 24 FC	55003	11		0
10413F80h	8	40 5F 35 00 00 00 00 00	2	351		0
105F0000h	8	23 95 5F 00 0E 0B 24 FC	55004	11		0
10613F80h	8	40 5F 35 00 00 00 00 00	2	351		0
107F0000h	8	23 95 5F 00 0E 0B 24 FC	55003	11		0
10807E00h	8	80 5F 35 00 00 00 02 06	4	351		0
1080BE00h	8	80 5F 35 00 00 00 02 06	2	351		0
1080FE00h	8	43 5F 35 00 41 99 03 00	2	351		0
109F0000h	8	23 95 5F 00 0E 0B 24 FC	55004	11		0
1F804000h	8	60 95 5F 00 00 00 00 00	55003	11		0
1F808000h	8	60 95 5F 00 00 00 00 00	55004	11		0
1F80C000h	8	60 95 5F 00 00 00 00 00	55003	11		0
1F810000h	8	60 95 5F 00 00 00 00 00	55004	11		0

Figure 3-6. Intercontroller communication traffic of application test

During this test, no communication losses are observed on CAN16 register and no shut downs were initiated so overall system performance looking promising due to absence of noticeable changes in operation routine in comparison to wired CAN.

3.2.3 Distance

As a baseline for estimation of network communication quality over distance, firstly a measurement between two gateway modules is taking place.

Both boards are running ESP-NOW and each one is the only peer of another. The first board placed stationary on the desk and the second one is connected to laptop so it can be moved around and provide data about message delivery. First test was done on the boards with internal ceramic antenna only and so the range of was very limited - less than 10 meters. Then, boards were equipped with a simple 2.4 GHz rubber duck antennas and the range exceeded quite noticeably.

While walking around the office floor few points were observed. On the Figure 3-7 those points are marked in violet colour and the first gateway is shown in the red circle. Points with number 1 are representing places where only some packet losses are starting but over 99.99% are still delivered. Between points 1 and 2 more losses are happening and the success percentage can drop down to

99%. After passing point 2 a region of uncertain communication quality starts. Therefore, for the next test, we will put gateways in a circle of 30 m.

Figure 3-7. Distance evaluation

3.2.4 Overnight test

The test setup is the following: three CUs equipped with gateway modules are placed according to Figure 3-8 (red circles). CU2 is connected to the corresponding laptop whereby means of WinScope tool the current state of CAN16 register and CU heartbeat (square wave signal with a period of 0,5s) are continuously logged.

Looking into the recorded log for the CU2 Figure 3-9 a further analysis can be made. On the x-axis hours are shown, value 7 of CAN16 representing all 3 CU are seen on the bus - 0b0111. Besides the first 20 minutes of scoping - configuring of the test setup, and last 10 minutes - shutting down the system, results are following.

A few spikes on the CAN16 graph are showing that there were some losses of connection on the second, fifth and eighteenth hours of observations. Taking a closer look into the first two spikes Figure 3-10 we can see quite similar behaviour. The overall time where the CAN16 register value is unstable is no more than 3 seconds with maximum continuous communication loss of about 0.5 seconds. Definitely, such behaviour isn't good by any means and depending on the CU's configuration it may cause a problem when the protection is set to shut down or provide only a warning without

interrupting overall behaviour - power generation systems have some inertia and therefor typical delays for next engine start/stop are 5 seconds so even if start/stop command is initiated at the beginning of shown connection loss, it will be declined immediately when communication is returned to normal.

Figure 3-8. Overnight test setup

On the other hand, the third spike looks worse inside. The overall time of CAN16 instability is 17 seconds but the longest duration of one spike is around 0.3 s. In this case it is also seen that CAN16 sometimes goes down completely meaning that connection is lost with all other CUs not like in the previous cases where the connection was lost only with one of them.

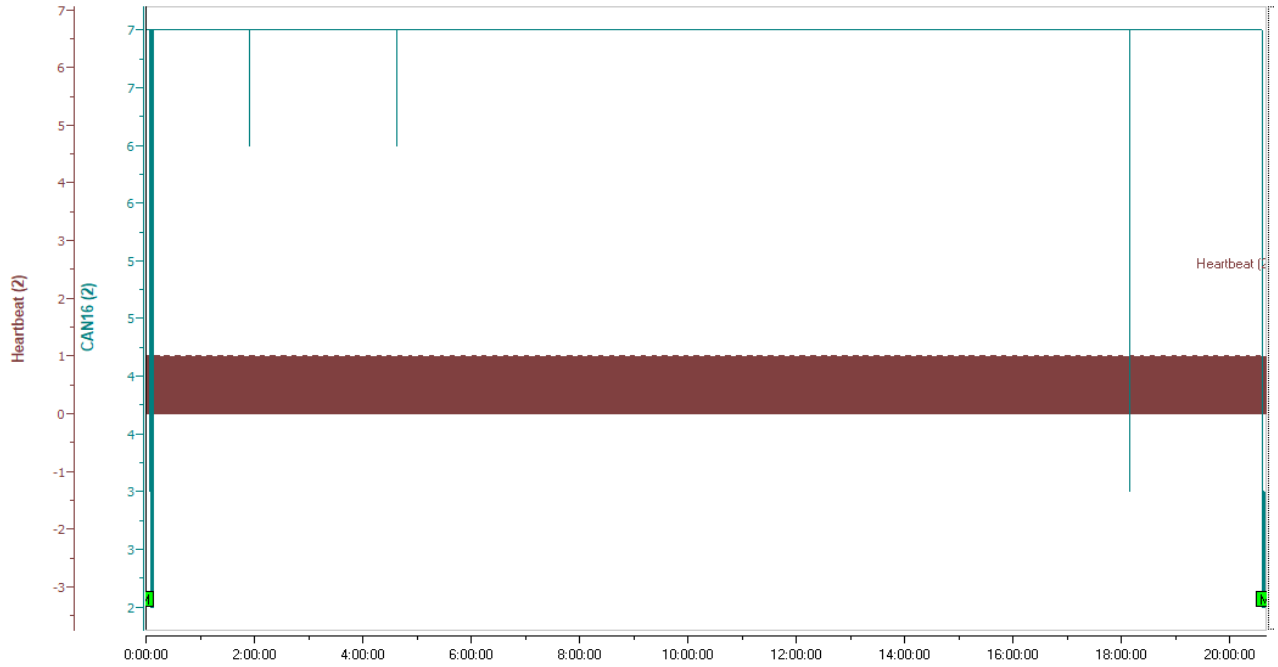


Figure 3-9. Overnight test CAN16 register log

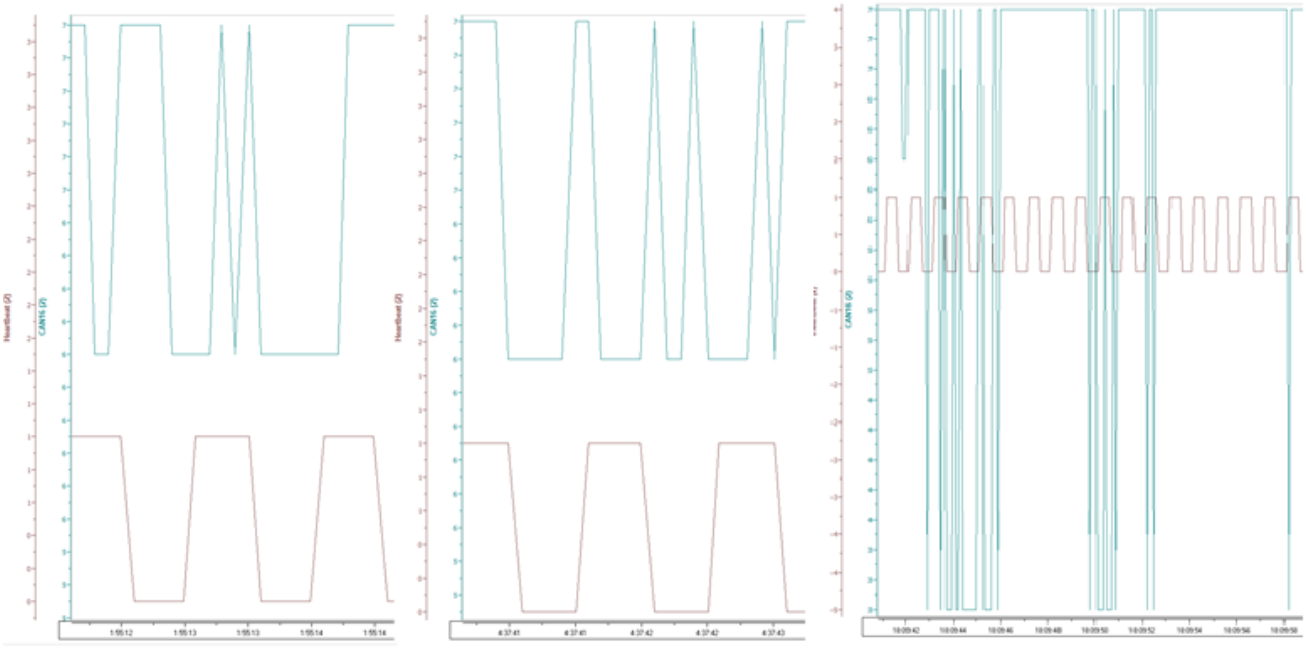


Figure 3-10. Detailed scopes of connection losses

3.2.5 Long-term test with 4 CUs

Scoping of CAN16 register providing a lower level access to CU state but to be sure that those changes are also propagated to the application layer another test is performed. Also, to see the overall trends of system behaviour the test duration increased. For this purpose, two InteliGen 200 and InteliGen 500 CUs are used due to their advanced PLC capabilities and history logging options. PLC work on the application layer and therefore all the changes there will be directly related to proper functioning. Each bit of CAN16 register can be taken as a binary value for PLC processing therefore a following PLC sheet is composed and written to the CU.

As an input values presents of bits 1 to 4 of CAN16 register are taken. Output of all bits, except the bit which represents host CU where PLC is running, are connected to block which will make a history record “CU № present” each time the host CU detects one of another 3 CUs on the bus. To detect the loss of connection 3 AND PLC blocks are utilized in the following way: one of the input of each is connected to the host CAN16 bit providing a constant logical 1 on that input, and another is connected to one of 3 others bits with inversion, meaning that the moment the loss will be detected AND block will initiate another history record “CU № lost”. PLC configuration shown in Figure 3-11.

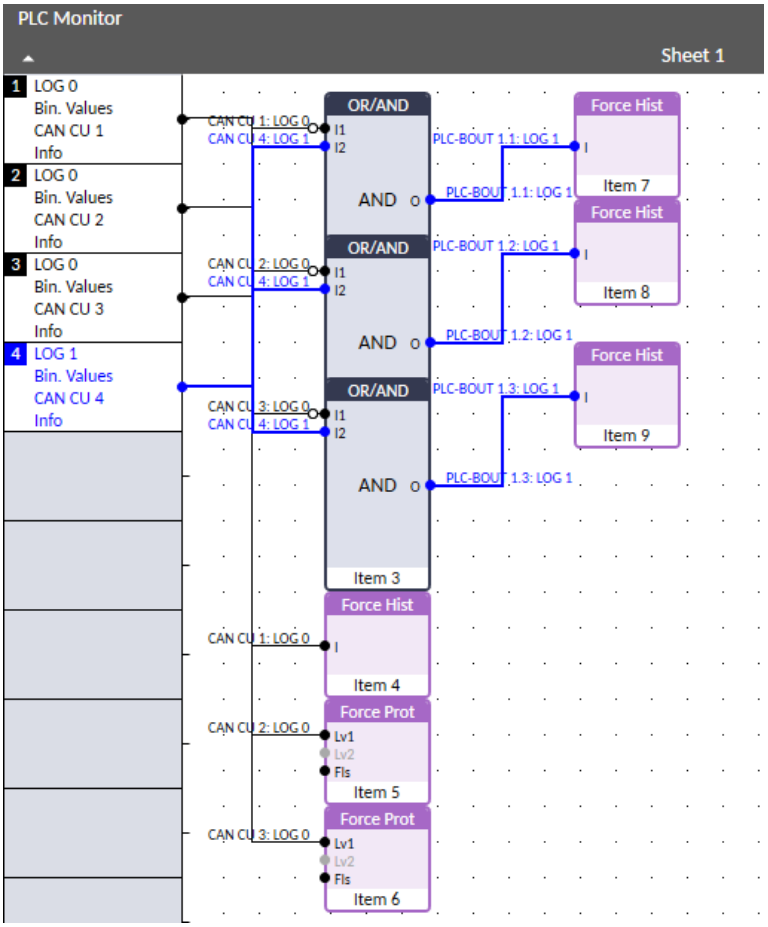


Figure 3-11. PLC configuration for long-term test, CU4

With such a configuration 4 CUs were left over the weekend. The history records that were taken in the end of test provides following data for further analysis, for now let's focus on history record of CU4 , test was started 8/7/2020 at 5:01:15 PM after record №-125 "SetpointChange" and the last record that is related to the test is on 8/8/2020 5:02:28 PM №-13 "CU1 present". Actual history logs from all CUs provided in the attached CD Appendix B - Contents of attached CD on page 49.

CU1 present	8/9/2020	5:02:28 PM	CU 3 lost	8/8/2020	10:39:32 PM	Hst CU2 present	8/8/2020	12:46:14 AM
CU 1 lost	8/9/2020	5:02:28 PM	Hst CU3 present	8/8/2020	5:50:02 PM	CU 2 lost	8/8/2020	12:46:14 AM
Hst CU2 present	8/9/2020	2:19:16 PM	CU 3 lost	8/8/2020	5:50:02 PM	Hst CU2 present	8/8/2020	12:46:14 AM
CU 2 lost	8/9/2020	2:19:16 PM	Hst CU2 present	8/8/2020	12:23:55 PM	CU 2 lost	8/8/2020	12:46:13 AM
Hst CU2 present	8/9/2020	9:08:00 AM	CU 2 lost	8/8/2020	12:23:55 PM	Hst CU2 present	8/8/2020	12:46:05 AM
CU 2 lost	8/9/2020	9:08:00 AM	Hst CU2 present	8/8/2020	12:23:47 PM	CU 2 lost	8/8/2020	12:46:05 AM
Hst CU2 present	8/9/2020	9:07:56 AM	CU 2 lost	8/8/2020	12:23:47 PM	Hst CU3 present	8/8/2020	12:45:52 AM
CU 2 lost	8/9/2020	9:07:56 AM	Hst CU2 present	8/8/2020	12:23:43 PM	CU 3 lost	8/8/2020	12:45:52 AM
Hst CU2 present	8/9/2020	9:07:52 AM	CU 2 lost	8/8/2020	12:23:43 PM	Hst CU3 present	8/8/2020	12:45:48 AM
CU 2 lost	8/9/2020	9:07:52 AM	Hst CU2 present	8/8/2020	6:45:44 AM	CU 3 lost	8/8/2020	12:45:48 AM
Hst CU2 present	8/9/2020	9:07:47 AM	CU 2 lost	8/8/2020	6:45:44 AM	Hst CU3 present	8/8/2020	12:45:44 AM
CU 2 lost	8/9/2020	9:07:47 AM	Hst CU3 present	8/8/2020	6:34:35 AM	CU 3 lost	8/8/2020	12:45:44 AM
Hst CU2 present	8/9/2020	3:52:43 AM	CU 3 lost	8/8/2020	6:34:35 AM	Hst CU2 present	8/8/2020	12:42:02 AM
CU 2 lost	8/9/2020	3:52:43 AM	Hst CU3 present	8/8/2020	6:34:31 AM	CU 2 lost	8/8/2020	12:42:02 AM
Hst CU3 present	8/9/2020	3:52:35 AM	CU 3 lost	8/8/2020	6:34:31 AM	Hst CU2 present	8/8/2020	12:41:54 AM
CU 3 lost	8/9/2020	3:52:35 AM	Hst CU3 present	8/8/2020	6:34:27 AM	CU 2 lost	8/8/2020	12:41:54 AM
Hst CU3 present	8/9/2020	3:52:31 AM	CU 3 lost	8/8/2020	6:34:27 AM	CU1 present	8/8/2020	12:41:51 AM
CU 3 lost	8/9/2020	3:52:31 AM	Hst CU3 present	8/8/2020	6:34:27 AM	CU 1 lost	8/8/2020	12:41:51 AM
Hst CU3 present	8/9/2020	3:52:27 AM	CU 3 lost	8/8/2020	6:34:27 AM	CU1 present	8/8/2020	12:41:51 AM
CU 3 lost	8/9/2020	3:52:27 AM	Hst CU3 present	8/8/2020	6:34:23 AM	CU 1 lost	8/8/2020	12:41:51 AM
CU1 present	8/9/2020	2:50:11 AM	CU 3 lost	8/8/2020	6:34:23 AM	CU1 present	8/8/2020	12:41:47 AM
CU 1 lost	8/9/2020	2:50:11 AM	CU1 present	8/8/2020	6:32:12 AM	CU 1 lost	8/8/2020	12:41:47 AM
CU1 present	8/8/2020	11:54:54 PM	CU 1 lost	8/8/2020	6:32:12 AM	CU1 present	8/8/2020	12:41:39 AM
CU 1 lost	8/8/2020	11:54:54 PM	CU1 present	8/8/2020	6:32:08 AM	CU 1 lost	8/8/2020	12:41:39 AM
Hst CU3 present	8/8/2020	10:39:52 PM	CU 1 lost	8/8/2020	6:32:08 AM	CU1 present	8/8/2020	12:41:39 AM
CU 3 lost	8/8/2020	10:39:52 PM	CU1 present	8/8/2020	6:32:04 AM	CU 1 lost	8/8/2020	12:41:39 AM
Hst CU3 present	8/8/2020	10:39:52 PM	CU 1 lost	8/8/2020	6:32:04 AM	CU1 present	8/8/2020	12:41:35 AM
CU 3 lost	8/8/2020	10:39:52 PM	CU1 present	8/8/2020	6:32:00 AM	CU 1 lost	8/8/2020	12:41:35 AM
Hst CU3 present	8/8/2020	10:39:44 PM	CU 1 lost	8/8/2020	6:32:00 AM	Hst CU3 present	8/7/2020	8:00:20 PM
CU 3 lost	8/8/2020	10:39:44 PM	CU1 present	8/8/2020	6:31:56 AM	CU 3 lost	8/7/2020	8:00:20 PM
Hst CU3 present	8/8/2020	10:39:40 PM	CU 1 lost	8/8/2020	6:31:56 AM	CU1 present	8/7/2020	8:00:07 PM
CU 3 lost	8/8/2020	10:39:40 PM	CU1 present	8/8/2020	1:35:44 AM	CU 1 lost	8/7/2020	8:00:07 PM
Hst CU3 present	8/8/2020	10:39:36 PM	CU 1 lost	8/8/2020	1:35:44 AM	CU1 present	8/7/2020	8:00:07 PM
CU 3 lost	8/8/2020	10:39:36 PM	CU1 present	8/8/2020	1:35:44 AM	CU 1 lost	8/7/2020	8:00:07 PM
Hst CU3 present	8/8/2020	10:39:32 PM	CU 1 lost	8/8/2020	1:35:44 AM	Hst CU3 present	8/7/2020	5:06:22 PM
CU 3 lost	8/8/2020	10:39:32 PM	Hst CU3 present	8/8/2020	1:35:41 AM	CU 3 lost	8/7/2020	5:06:22 PM
Hst CU3 present	8/8/2020	10:39:32 PM	CU 3 lost	8/8/2020	1:35:41 AM	Hst CU3 present	8/7/2020	5:06:18 PM
						CU 3 lost	8/7/2020	5:06:18 PM

Figure 3-12. Long-term test history log CU4

It is seen that each connection loss is going in pair with a returning back to presents of communication with corresponding CU. Those events are recorded in the same second and one second is the resolution of time stamp so most likely that disconnection is active for less than that, similar to the Figure 3-10 on page 34. A few disconnections are happening in a row and usually within one minute or less. The durations of periods with stable communication without any history records are from 1 to 6 hours.

This result may explain while during the application test in 3.2.2 no communication losses were observed – the test was performed in one hour and even after that the system was left for few hours in a still state without any warnings that will represent a communication loss.

Comparing the results from 4 CUs it is seen that most of the time losses were happening not synchronously in the all nodes – some of them were facing communication losses more often than another. Moreover, disconnections were happening inconsistently in terms of pairing, meaning that when for example CU1 was detecting a lost with CU2 with corresponding record “CU2 lost” the CU2 didn’t faced any problem with detection of the CU1 in particular and to other nodes in general. Counting losses that were happening within one minute as on incident the total amount of such incidents in each node are following: CU1 – 19, CU2 – 25, CU3 – 21, CU4 – 15.

Evaluation of PDR gives a result of 99.9995%, considering the system is producing around 120 messages per second, in other words the period is 8 milliseconds, the result is aligning with observation in section 3.2.1 on page 26.

From all the tests mentioned above among with the tests that were performed during the development without explicit logging and scoping it can be said that such behaviour of system will be propagated not only for a few days but for whole time of system work.

On a positive side of system performance is the fact that no fatal failures occurred – each loss of connectivity with a particular CU was followed by immediate revive of communication. The first reason that comes to mind is a simple packet loss in the air medium during transmission, there is still no wireless technology that is completely free from such occasions. Another possible cause of loss is may happening inside the gateway itself between Wi-Fi rx/tx buffers to the CAN tx/rx due to arbitration, but changes in queues size did not produced a noticeable result.

Chapter 4

Future work

While this work provides a solution that can work to some extend and shows that the wire in CAN bus communication can be replaced by some wireless network the overall question about replacing wired CAN bus by wireless solution where peer-to-peer communication between nodes provided is still open for future work in several directions and will be continued.

4.1 Publicly available networking technologies

Even that painlessMesh did not provide a sufficient result by any means, the overall mesh networking may be suitable for the target application. Both ESP-Wi-Fi-MESH and ESP-BLE-MESH may be promising, and it is planned as a next step to try them out. Unfortunately, due to the current level of programming skills and yet still not a very user-friendly SDK (ESP-BLE-MESH is in beta) for both networks their performance was not evaluated in the scope of this work.

Another option for the networking solutions that are publicly available and may be evaluated without huge investments provided by Nordic Semiconductor. Those are [22]:

- Bluetooth mesh as a state-of-the-art standard that extends the capabilities of Bluetooth Low Energy to include the mesh networking. It enables powerful concurrent multicast (many-to-many) communication in networks with thousands of devices. The functionality is a vital update for new applications in lighting, sensor networking, predictive maintenance, asset tracking and positioning. Bluetooth mesh is a managed flooding mesh, which is a simple and reliable approach to distribute messages in a larger networks. Reliability is ensured with multiple paths from source to destination and there is no single point of failure. Security permeates the technology, network and application security are completely separated, and protection against attacks is built in.
- 2.4 GHz proprietary RF. There are occasions when complete control of the wireless link is required for reasons such as low latency, reduced packet size or particular unique protocol behavior. The nRF52 and nRF51 Series wireless SoCs all support 2.4 GHz proprietary development. As multiprotocol wireless SoCs they offer simultaneous Bluetooth Low Energy operation, or another supported protocol, if the application demands it. Whilst 2.4 GHz proprietary development does not offer the interoperability that comes with standards like Bluetooth, it can offer special abilities to tailor both ends of a communication link for maximum efficiencies.

4.2 LumenRadio AB

At the time of writing this thesis we reached out to LumenRadio business manager and application engineer to discuss the suitability of their solution to our application of interest. After explaining the use case scenario and providing the requirements they claimed that their solution will be sufficient in terms of numbers of nodes, PDR and overall system data throughput and range. Also, looking through their case studies [23] it can be seen that their solutions are capable to work in harsh industrial environments.

The only concern that is raised is the possible delays and latencies working with message rates for our application with the number of nodes more than 10 because while it is claimed that all data will be delivered and the communication is reliable, an average message rate of case study applications (HVAC, asset tracking, monitoring) aren't that high. Therefore, we are continuing our discussion and looking towards to try out evaluation kit or have a business trip to their facility with a prepared test setup for a workshop and hands-on experience.

4.3 Wireless systems commissioning and monitoring

Even today, the issue of reliable real-time wireless communication is still a big topic for a research. and it is not an obvious question about reaching same reliability as wired systems so most likely in future we will face more hybrid systems of wired and wireless solution or a few wireless technologies will be combined together to provide required level of communication quality.

On the other hand, knowing the caveats and sources of possible problems in existing wireless networks some measures can be taken to prevent those issues and provide better result. Those problems are related to decrease of PDR in relation to load of air medium or specific channel in wireless domain, and on the stage of arbitration when the data is moving from air to wired bus like in the application that is investigated in this thesis.

Before introducing any wireless system, that may handle critical data, into the existing industrial site or any facility in general a good first measure will be to investigate and scope what is the current load of each channel in particular spectrum waveband range and what sort of EMC noise can be generated during the site operation like ignition of engines or switching contacts. This will provide a background on the choice of technology to avoid some waveband overload or at least will be a warning for a possible issue with technology if no other one that occupies different spectrum wave band can be applied.

As a next step a system for monitoring and control of wireless network can be proposed. Such system monitors current state of wireless network parameters such as PDR, RSSI among with environment variables like current channels occupancy and knows in advance about possible events that can cause EMC disturbance like engine start/stop. This data can act as an input for a control process that is responsible for automatic channel switching as a consequence of PDR decrease or dismiss all possible alarms or shutdowns that can be initiated due to the connectivity losses in the wireless system.

4.4 Rethinking application layer

Usually, the CAN bus provides a foundation for the actual application layer of OSI model in industrial monitoring and automation systems and those layers were designed counting the reliability and robustness of it. But when we move to the wireless domains the situation is changing a bit – you can never guarantee a 100% of all packets delivering, especially operating in the ISM band.

After performing a test run in the application of interest with particular wireless system common rates of PDR can be established and this level may be adopted as a foundation for application layer configuration, meaning that only after some particular loss rate safety measures will start to take place. Similar way of processing can be already implemented in intercontroller communication by changing the type of action that is taken when communication is interrupted for less than some predefined period of time. This way, even the system that were developed in this work may be applicable and in combination with a monitoring system proposed in section above the required reliability of system can be reached.

Chapter 5

Conclusion

The purpose of this thesis was to create a working prototype of the networking system that is suitable to replace wires in CAN based communication. The initial requirements mentioned in 1.6 on page 8 were mostly fulfilled with the exception of reaching the 100 meters working distance due to main focus on reaching better stability of connectivity. A few causes that are responsible for the decrease of PDR and overall system performance were stated and ideas for their elimination proposed together with steps for further research.

- a few wireless technologies were identified as capable to replace CAN communication in the cluster of interconnected genset controllers
- CAN to wireless gateway module was designed and tested in the target application, long-term performance and distance
- the functionality was demonstrated with 4 CU nodes

Test results are showing that the developed system and particularly used wireless networking technology would not be sufficient for the real aim of the future-proof product development – up to 64 nodes in the network with full support of CAN FD, but on the other hand it may be suitable for other, less critical applications. For example, as a replacement of wired CAN bus in communication between CU and extension modules – shorter distances and orders lower messaging rates, as a cheap bridge between two or more CAN buses where achieved PDR is already acceptable or it even can be transformed into wireless monitoring/logging tool for SCADA or PC, Android and iOS devices.

Using this work, further development of systems that are capable to replace wires in machine-to-machine communication lines will be continued.

Bibliography

[1] “Diesel generator market size, share and industry analysis,” [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/diesel-generator-market-100587. [Accessed 01 08 2020].

[2] “History of CAN technology,” [Online]. Available: https://www.can-cia.org/en/can-knowledge/can/can-history/[Accessed 01 08 2020].

[3] “ComAp a.s. InteliGen 500 Global guide,” [Online]. Available: https://www.comap-control.com/support/download-center/documentation/man/inteligen-500-global-guide-1-2-0?lang=en-GB [Accessed 01 08 2020].

[4] “GCAN-211 Wi-Fi to CAN Shenyang Guangcheng Technology Co., Ltd,” [Online]. Available: http://www1.gcanbox.com/fsd/canzxwg/GCAN-211.html [Accessed 01 08 2020].

[5] “HD67644-Wi-Fi-B2 from ADFweb.com S.r.l,” [Online]. Available: https://www.adfweb.com/Home/products/CAN_Wi-Fi.asp?frompg=nav1_26 [Accessed 01 08 2020].

[6] “CAN-Wi-Fi WIRELESS CAN INTERFACE Grid Connect,Inc.,” [Online]. Available: https://www.gridconnect.com/products/can-Wi-Fi-wireless-wi-fi-can-diagnostic-monitoring-development-tool [Accessed 01 08 2020].

[7] “PCAN-Wireless Gateway PEAK-System Technik GmbH,” [Online]. Available: https://www.peak-system.com/PCAN-Wireless-Gateway.331.0.html?&L=1 [Accessed 01 08 2020].

[8] “Kvaser Air Bridge Light HS,” [Online]. Available: https://www.kvaser.com/product/kvaser-air-bridge-light-hs/ [Accessed 01 08 2020].

[9] “How to connect multiple Kvaser BlackBirds,” [Online]. Available: https://www.kvaser.com/developer-blog/how-to-connect-multiple-blackbirds/#/section_four [Accessed 01 08 2020].

[10] “LumenRadio Company,” [Online]. Available: https://lumenradio.com/company/ [Accessed 01 08 2020].

[11] S.Saranya, “A Comparison of Wired Technology & Wireless Technology for effective Communication,” *International Journal of Innovative Research in Computer and Communication Engineering,* vol. 5, no. 3, 2017.

[12] S. S. a. H. C. Mahmoud Elkhodr, “EMERGING WIRELESS TECHNOLOGIES IN THE INTERNET OF THINGS: A COMPARATIVE STUDY,” *International Journal of Wireless & Mobile Networks (IJWMN),* vol. 8, no. 5, 2016.

[13] J. Salazar, Wireless networks, Czech Technical University of Prague Faculty of electrical engineering, 2017.

[14] "ESP32 Series SoCs," [Online]. Available: https://www.espressif.com/en/products/socs [Accessed 01 08 2020].

[15] "Techfun s.r.o e-shop," [Online]. Available: https://techfun.sk/ [Accessed 01 08 2020].

[16] "FreeRTOS™," [Online]. Available: https://www.freertos.org/index.html [Accessed 01 08 2020].

[17] T. Barth, "A CAN driver for the ESP32," [Online]. Available: http://www.barth-dev.de/can-driver-esp32/ [Accessed 01 08 2020].

[18] "SJA1000 Datasheet," Philips Semiconductors, 4 January 2000. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/SJA1000.pdf. [Accessed 01 08 2020].

[19] "PainlessMesh," [Online]. Available: https://gitlab.com/painlessMesh/painlessMesh [Accessed 01 08 2020].

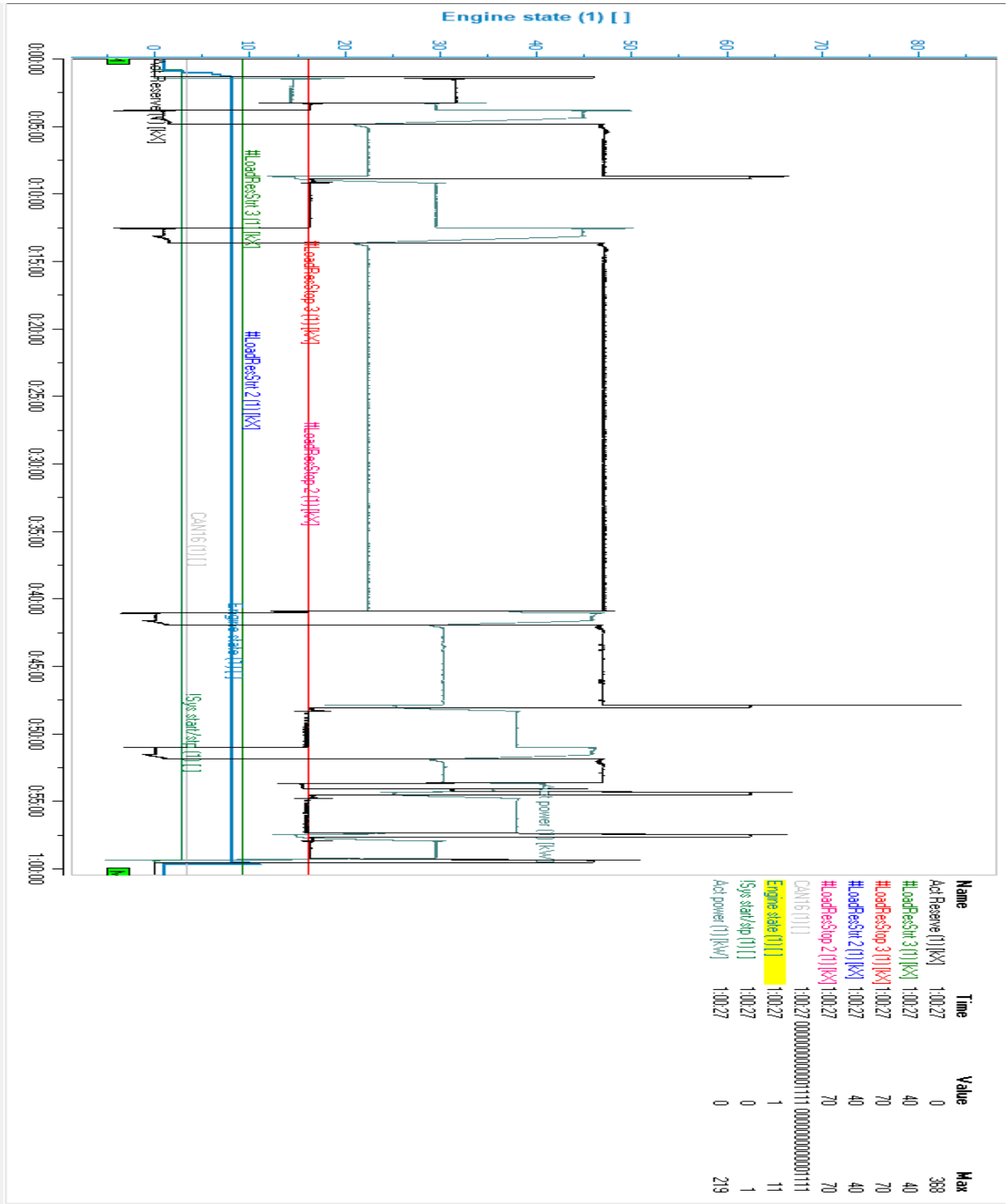
[20] "ESP-NOW," Espressif Systems,, [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html [Accessed 01 08 2020].

[21] "PCAN-View," PEAK Systems, [Online]. Available: https://www.peak-system.com/PCAN-View.242.0.html?&L=1 [Accessed 01 08 2020].

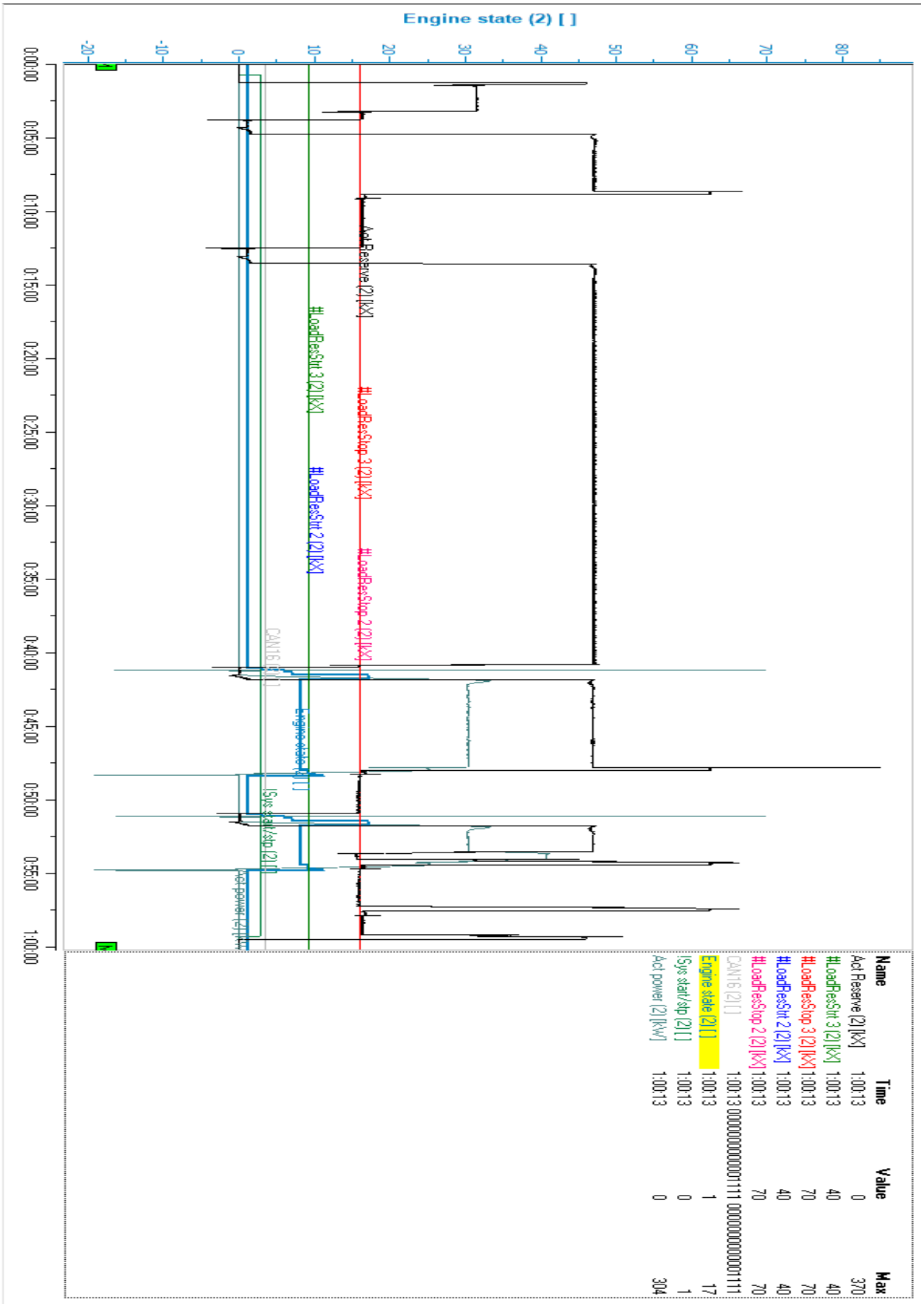
[22] "Nordic products," Nordic Semiconductor, [Online]. Available: https://www.nordicsemi.com/Products [Accessed 01 08 2020].

[23] "Case studies," LumenRadio, [Online]. Available: https://lumenradio.com/customers/#mira---wireless-mesh [Accessed 01 08 2020].

Appendix A - Application test logs



Application test scope CU1



Application test scope CU2

Appendix B - Contents of attached CD

```
CD attachment:
│   thesis_chereda.pdf
│
├───Arduino
│   └───libraries
│       ├───esp32-can-protocol-demo-master
│       │   │   keywords.txt
│       │   │   library.properties
│       │   │   README.md
│       │   │
│       │   ├───examples
│       │   │   ├───esp32can
│       │   │   │       esp32can.ino
│       │   │   │
│       │   │   └───esp32cansend
│       │   │           esp32cansend.ino
│       │   │
│       │   └───src
│       │           CAN.c
│       │           CAN.h
│       │           CAN_config.h
│       │           can_regdef.h
│       │           ESP32CAN.cpp
│       │           ESP32CAN.h
│       │
│       ├───CAN-to-ESP-NOW
│       │       CAN-to-ESP-NOW.ino
│       │
│       └───ESPNOW-test
│               NOWtest.ino
│
├───Long-tern-history-logs
│       History-CU2.xls
│       History-CU1.xls
│       History-CU4.xls
│       History-CU3.xls
│
└───WinScope-scopes
        MK_test_CU3.SDT
        MK_test_CU1.SDT
        MK_test_CU2.SDT
```