

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

User Interface Display with CAN bus communication

João Luís Carvalho Martins



Mestrado em Engenharia Eletrotécnica e Computadores

Supervisor: Mário Jorge Rodrigues de Sousa

March 24, 2022

Abstract

The theme of this dissertation was proposed by the technological company AddVolt. AddVolt has developed the world's first electric Plug-in system, aimed at the transport sector, focused on the refrigerated market. This system allows refrigerated trucks to conserve their products in electric mode, reducing diesel consumption, noise production and CO2 emissions into the environment.

This system features an interface that monitors and informs the driver, at all times, about the status of the Addvolt system. However, it was obvious this interface had some problems and limitations. Since the limitation of variables, to poor visibility and only monitoring of the system outside it, with the truck stopped. This dissertation intends to create a new solution for the Addvolt system interface, which is capable of eliminating the problems identified in the current interface and presenting new functionalities.

We therefore intend to have a display as the new interface, where different pages can be created with different purposes. Thus, the limitation of variables ceases to exist. All the information needed to update the display is available on the CAN bus, where until now, to receive these messages, a physical connection to the bus is required. The integration of wireless communication, Bluetooth, makes this physical connection no longer necessary. In cases where the interface would have to be outside the truck, it can now be in the driver's cabin. Furthermore, in a service context, to analyze data in real time when we want to connect to the system, once again, this connection is made to the CAN bus through an adapter that has a high cost. Bluetooth allows you to save on the costs that it implies.

The new interface to be developed must therefore make available to the driver any type of information in a visible, simple and objective way. It should also be able to communicate via CAN and Bluetooth.

After having raised the main problems and the necessary requirements to obtain a valid solution, the implementation proceeded. All technologies discussed will be tested. After obtaining the first solution, functional tests were carried out in order to fulfill the requirements mentioned above.

In conclusion, the new solution can show any type of information to the user and establish communication with the system physically via the CAN bus or wirelessly via Bluetooth.

Resumo

O tema da presente dissertação foi proposto pela empresa tecnológica AddVolt. A AddVolt desenvolveu o primeiro sistema Plug-in elétrico do mundo, direcionado para o setor de transporte, focado no mercado refrigerado. Este sistema permite, aos caminhões com refrigeração, a conservação dos seus produtos em modo elétrico, reduzindo o consumo de diesel, a produção de ruídos e a emissão de CO₂ para o meio ambiente.

Este sistema apresenta uma interface que monitoriza e informa o condutor, a todo o momento, sobre o estado do sistema Addvolt. No entanto, verificou-se que esta interface apresentava alguns problemas. Desde limitação de variáveis, a fraca visibilidade e apenas monitorização do sistema no exterior deste, com o caminhão parado. Esta dissertação pretende criar uma nova solução para a interface do sistema Addvolt, que seja capaz de eliminar os problemas identificados na interface atual e apresentar novas funcionalidades.

Pretendemos assim que a nova interface passe a ser um display onde diferentes páginas podem ser criadas com diferentes finalidades. Assim, a limitação de variáveis deixa de existir. Toda a informação necessária para a atualização do display está disponível no barramento CAN, onde até agora, para receber estas mensagens uma ligação física ao barramento é necessária. A integração de uma comunicação sem fios, o Bluetooth, esta ligação física deixa de ser necessária. Em casos onde a interface teria de ficar no exterior do caminhão, pode agora estar na cabine do condutor. Para além disso, num contexto de serviço, para analisar dados em tempo real quando nos queremos ligar ao sistema, mais uma vez, esta ligação é feita ao barramento CAN através de um adaptador que apresenta um elevado custo. O Bluetooth permite poupar nos custos que este implica.

A nova interface a ser desenvolvida deve então disponibilizar, ao condutor, toda e qualquer tipo de informação de forma visível, simples e objetiva. Deve também conseguir comunicar através de CAN e Bluetooth.

Após se terem levantado os principais problemas e os requisitos necessários para se obter uma solução válida, passou-se à implementação. Todas as tecnologias abordadas vão ser testadas. Após a obtenção da primeira solução, foram levados a cabo testes funcionais de forma a cumprir os requisitos expostos anteriormente.

Concluindo, a nova solução consegue mostrar qualquer tipo de informação ao utilizador e estabelecer uma comunicação com o sistema fisicamente através do barramento CAN ou sem fios através do Bluetooth.

Acknowledgements

To begin I would like to thank my parents, my brother, Eduardo, my girlfriend, Leticia and all my friends for all the support and strength over the years. without you all my academic journey wasn't possible.

A sincere thank you to Addvolt for allowing me to be a part of their team. Everyone was incredibly kind to me and always offered me help whenever I needed it. Special thanks to Ricardo Gonçalves and José Pedro, for being the ones who watched me up close and for all the guidance. There wasn't a day I haven't learned something new from you.

To professor Mario Jorge a thank you for always being available to listen at all times.

João Luís Carvalho Martins

*“Until I began to learn to draw,
I was never much interested in looking at art.”*

Richard P. Feynman

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	1
1.3	Goals	2
1.4	Methodology	2
1.5	Document Structure	3
2	State of the art	5
2.1	CAN	5
2.1.1	Overview	5
2.1.2	Standard and Extended CAN	7
2.1.3	Message types	9
2.1.4	Data transmission in CAN	9
2.1.5	Termination types	11
2.2	Displays	13
2.2.1	Touchscreen vs Buttons	14
2.3	Bluetooth	15
2.3.1	Pairing	15
2.3.2	Bluetooth versions	16
2.4	Bluetooth Low Energy	18
2.4.1	Protocol stack	18
2.4.2	BLE packet	19
2.4.3	BLE roles	20
2.4.4	BLE data transfer	20
2.5	XMC4300 Relax Kit	21
2.6	Click boards	22
2.7	PCAN-View & PCAN-USB	23
2.8	Communication Protocols	24
3	Requirements analysis	31
3.1	Problem Description	31
3.1.1	Current technology	31
3.1.2	Main Problems	32
3.2	Proposed solution	34
3.3	Requirements	36
3.3.1	Device Communicates via CAN	36
3.3.2	Display shows user system status	37
3.3.3	User can interact with the device	37

3.3.4	Device responds in real-time to user inputs	37
3.3.5	Device can be debugged and updated via CAN	38
3.3.6	Prototype is approved	38
4	Implementation	39
4.1	Displays	40
4.1.1	LCD Click	40
4.1.2	OLED Click	48
4.2	Bluetooth	51
4.2.1	Overview	52
4.2.2	Connection and Communication	53
4.3	User Interface Design	57
4.3.1	LCD Display	58
4.4	User Interface Implementation	59
4.5	Solution	60
4.5.1	Display & CAN	62
4.5.2	Display & CAN & BLE	64
4.5.3	PCB	67
5	Work results	73
5.1	Requirements validation	73
5.2	Comparison with previous solution	76
6	Conclusions and Future Work	79
6.1	Conclusion	79
6.2	Future Work	79
A	Python script developed	81
	References	83

List of Figures

2.1	CAN network	5
2.2	Layered ISO 11898 Standard Architecture, [7]	7
2.3	Standard	8
2.4	Extended	8
2.5	CAN node	10
2.6	CAN voltage [5]	11
2.7	CAN nodes transmitting at the same time	11
2.8	Standard Termination, [10]	12
2.9	Split Termination [10]	12
2.10	Biased Split Termination, [26]	13
2.11	Piconet topologies, [6]	15
2.12	Bluetooth versions comparison, [4]	18
2.13	Bluetooth Low energy protocol stack, [8]	18
2.14	BLE packet structure[17]	19
2.15	BLE packet structure[17]	20
2.16	BLE connection interval and connection event[1]	21
2.17	XMC4300 Relax Kit.[13]	22
2.18	MikroBUS	23
2.19	PCAN-View	23
2.20	Hardware, PCAN-USB and pin assignment	24
2.21	SPI[16]	25
2.22	SPI timings[16]	25
2.23	SPI topologies: Left figure - Standatd; Right figure - Ring or daisy chain[16]	26
2.24	I2C[3]	26
2.25	Data transfer in I2C[3]	27
2.26	UART connection	28
2.27	UART message	28
3.1	Addvolt current solution	31
3.2	Truck with trailer example	33
3.3	Truck example	33
3.4	Solution use case diagram	34
3.5	User interface in trucks with trailer	35
3.6	User interface with normal truck	36
4.1	DAVE Apps	40
4.2	LCD click module & LCD click schematic[20]	41
4.3	Interactions	42

4.4	SPI addressing registers[18]	42
4.5	Write timing diagram[18]	43
4.6	List of characters available[28]	44
4.7	List of instructions available[28]	45
4.8	Initialization process[28]	46
4.9	After initialization	47
4.10	LCD click validation	48
4.11	OLED click module & OLED click schematic [21]	49
4.12	OLED click write timings[29]	50
4.13	RGB[29]	50
4.14	OLED initialization[27]	51
4.15	OLED click validation	51
4.16	BLE click module	52
4.17	BLE click schematic[19]	52
4.18	BLE click connected to PC	53
4.19	Basic AT commands	54
4.20	Powering two BLE clicks	55
4.21	Connection established with Datapump configuration	56
4.22	Binary mode	56
4.23	LCD 20x4	57
4.24	TFT LCD display	58
4.25	LCD mock-ups	58
4.26	How to change page	59
4.27	Page views	60
4.28	Interface system	61
4.29	Prototype developed	61
4.30	Display with CAN	62
4.31	CAN msg AddVolt protocol	62
4.32	Variables structure	63
4.33	message represented in PCAN view	63
4.34	Architecture	64
4.35	CAN and BLE	64
4.36	BLE message format	65
4.37	sub_can format	65
4.38	Bluetooth messages handling	66
4.39	System Protections	67
4.40	Microcontroller connections	68
4.41	Display connections	69
4.42	CAN connections	70
4.43	Bluetooth connections	70
4.44	Board connections	71
4.45	Top layer and Bottom layer	72
4.46	PCB printed	72
5.1	Prototype connected to Addvolt system	74
5.2	First test of BLE messages	75
5.3	Second test of BLE messages	76
A.1	Python script to send BLE messages	81

A.2	Python script to receive BLE messages	82
-----	---	----

Chapter 1

Introduction

1.1 Context

Addvolt has the mission to anticipate solutions for sustainable transport. It focuses in the transport of refrigerated merchandise. A refrigerated truck has a diesel engine dedicated only to the refrigeration, so Addvolt developed a plug-in electric system which reduces fuel usage, reduces noise and CO2 emission of refrigeration. The system developed can be adaptable to every truck allowing this way to electrically feed auxiliary systems with his stored energy. Addvolt technology also allows to recover energy during the braking of a truck in order to supply power to any auxiliary systems of the truck. Through this technology the refrigeration units are now more efficient and sustainable. With all of this comes the need for Addvolt and the truck drivers to be able to extract important information in order to monitor and control the system behavior whenever in use.

1.2 Motivation

Addvolt operates in the technological sector, it's a company that wants to improve and innovate with his solutions, always developing new and better ideas. This dissertation aims to develop and innovate a part of the Addvolt system, the user interface.

Although there is already a solution in use, Addvolt came to the conclusion that it can and it should exist a better interface that displays all information about Addvolt system. The current solution comes with numerous restrictions. It offers a limited amount of information to the user and no quantitative information can be displayed. This solution wasn't developed in Addvolt, it's provided by another company, therefore it doesn't have the flexibility to address other problems already identified by Addvolt but not solved, such as, cases of trucks with trailers. Here, the user interface has to be outside, being susceptible to different climates that can influence the visibility. Another problem is in a context of maintenance. Here, to analyse different variables a specific

adaptor is used in order to connect physically to the system. For the reasons mentioned, comes up the motivation and the challenge of developing a whole new interface, completely from scratch, that has no restrictions, can introduce new technologies, it's fully customizable, is an improvement compared to the old solution and it solves the challenges, problems identified and described before.

1.3 Goals

This dissertation aims to develop a fully functional prototype that is able to interact with Addvolt products and its users. The user interface must be able to:

- Display all the information, relevant to the truck driver, about the Addvolt system;
- Maintain the simplicity of how the information is provided;
- Overcome the issue of using adapters for maintenance or inspections;
- Overcome the problem that a truck with trailer imposes and obligates the interface to be outside the truck.

1.4 Methodology

In order to achieve the final solution and technology, this project was divided in different steps:

- **Requirements:** Given the problem in hands and what we want for a solution, a list o requirements is created in order to validate the solution developed;
- **Solution Proposal:** The proposed solution must obey all requirements proposed, give solution to problems that emerged and were described and at least have the same functionalities as the previous solution;
- **Technology validation:** Each technology is tested individually to ensure that every individual functionalities is achieved.
- **Solution implementation:** Where all technologies come together and become one system, the desire solution;
- **Validation of solution:** The achieved solution must be tested in a way that re-ensures that all problems and new ideas are correctly implemented;
- **Results analysis:** Finally all the work will be into analysis in order to compare and quantify if the solution achieved is indeed a better one.

1.5 Document Structure

This document is divided in several chapters, each having crucial importance to the developed work through out the hole project. Here a description to each one is exhibited.

- **Chapter 1:** The *Introduction* chapter, introduces the theme of this dissertation, as well as its context and goals.
- **Chapter 2:** The *State of Art* chapter, a closer look to all technologies related to the theme were explored and explained;
- **Chapter 3:** The *Requirements Analysis* chapter, the main purpose and problem presented by Addvolt, that has culminated in this dissertation, is explained and detailed. Then, all requirements necessary for the solution are listed and in the end of the chapter, the proposed solution for the current problem is presented and explained;
- **Chapter 4:** The *Implementation* chapter, every step of the process that resulted in the final solution is shown;
- **Chapter 5:** The *Work Results* chapter, all requirements listed before are checked by doing validation tests followed by an analysis of this results ;
- **Chapter 6:** The *Conclusions and Future Work* chapter, a conclusion of all the work done is discussed and ideas and suggestions of future work are specified.

Chapter 2

State of the art

In this chapter every concept that is related and relevant to the project is studied and analysed in depth. Understanding each point that will be covered will give a better overview of the project in hands.

2.1 CAN

2.1.1 Overview

CAN (Controller Area Network), was originally created for vehicles. Before, electronic devices inside vehicles were connected using point to point wiring systems, and over time electronic devices were used more and more in vehicles which resulted into having a high volume of wires that were heavy and expensive. CAN protocol came has a solution. We can imagine CAN as the nervous system that enables communication, and all the nodes, electronic devices, are interconnected via CAN. Information can be shared to all nodes, that's the beauty of the CAN protocol. Since it's introduction, CAN has proved to be a simple and versatile technology and can now be found in all areas which imply electronic devices to communicate with each other, not just in the automobile industry.

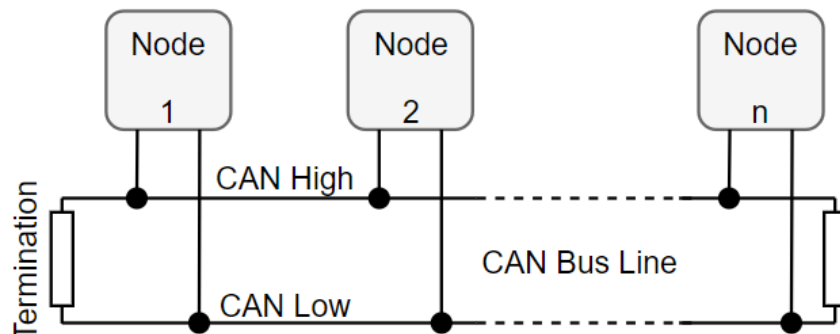


Figure 2.1: CAN network

What are the advantages of using the CAN bus protocol:

- **Simple and Low Cost** - A vehicle, has sensors to capture data and actuators to send data, it requires a dedicated wire to each individual sensor and actuator. One wire to communicate with the airbag system, one wire to communicate with the ABS braking system, one wire to the engine control, etc. With CAN protocol, a single wire connects all electronic systems and sensors, in the vehicle, into one circuit that facilitates high-speed data transmission between all components, reduces errors, weight and costs; [32]
- **Built-in Error Detection** - If a node detects an error it transmits a special error flag, even the node that is generating the error will detect it and eventually it will lead up to stop participating in the network traffic. This way allows CAN nodes to not only detect errors, but also prevent faulty devices from creating useless bus traffic;[32]
- **Robustness** - The protocol's error detection capabilities are advantageous, as they enable systems to identify and recover from errors on their own without intervention from an outside actor;[32]
- **Speed:**
 - **High-Speed** - Depending on the length of the cable, High-Speed CAN offers transfer rates of between 40 kbps and 1 Mbps;
 - **Low-Speed** - It offers lower transfer rates that may start at 40 kbps but go up to 125 kbps. The lower rates allow communication to continue, even when a wiring failure exists. While high-speed CAN networks terminate at either end of the bus line with a 120-ohm resistor, each device in a low-speed CAN network has its own termination. Low Speed CAN network presents greater fault tolerance and are more resistant failure modes.[32]
- **Flexibility** - In CAN bus protocol, nodes can easily be added or removed without performing any software or hardware updates on the system. This feature makes it easier to integrate new electronic devices into the CAN bus network without significant programming overhead;[32]
- **Priority** - If two nodes try to send a message simultaneously, because every message has a priority, the one with the highest priority is transmitted first;
- **Broadcast Communication** - Because each node is intelligent, and every message is received in every node of the CAN bus, each node has the freedom to decide if he wants to ignore the message or to collect the message.

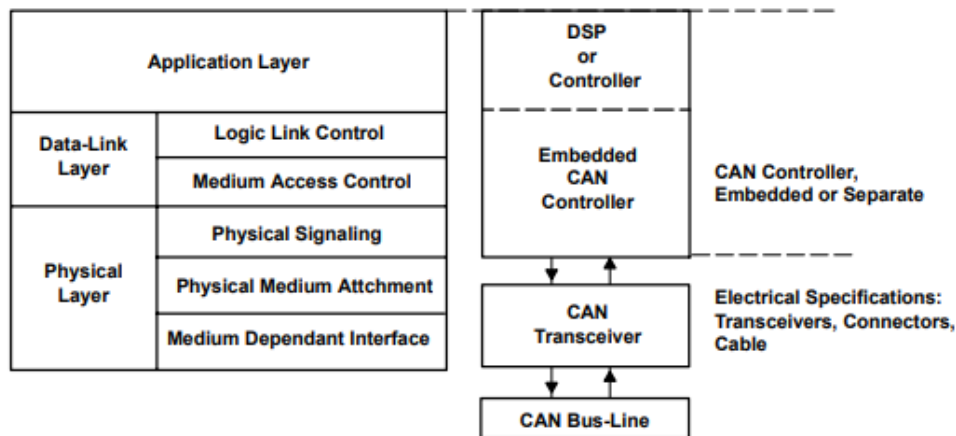


Figure 2.2: Layered ISO 11898 Standard Architecture, [7]

Figure 2.2 illustrates the layers defined in ISO-11898. Here, how data is transmitted between devices on a network is defined in layers. [31][7]

- **Data-Link Layer:** It's the layer responsible for the transfer of data between nodes, as well as to detect and correct possible errors that may occur in the physical layer. It's divided into two sub-layers:
 - MAC: Media Access Control, it specifies how the nodes in the network gain access to the medium. Provides error detection, the encapsulation and decapsulation of data and signaling.
 - LLC: Logical Link Control and is responsible for frame acceptance, overload notification, filtering and recovery management.
- **Physical Layer:** It's the layer responsible for the transmission of raw data. It defines the specifications for the parameters of transmission such as voltage level, timing and data rates.

2.1.2 Standard and Extended CAN

There are two variants of CAN frames length, Standard CAN (2.0A) and Extended CAN (2.0B), the main difference is that the first has a 11-bit identifier and the second a 29-bit identifier as well as the 11-bit identifier.[31][7]

2.1.2.1 Standard CAN:

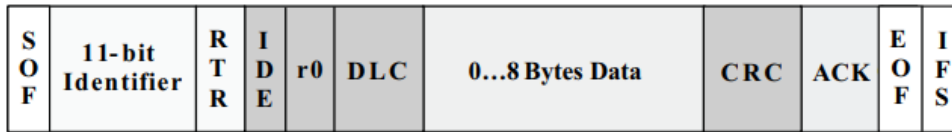


Figure 2.3: Standard

- **SOF (1-bit)** - The single dominant start of frame, marks the start of a message, and is used to synchronize the nodes on a bus after being idle (Must be dominant)
- **Identifier** - 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.
- **RTR (1-bit)** - Remote transmission request. This bit is dominant when information is requested by another node.
- **IDE (1-bit)** -
- **r0 (1-bit)** - Reserved for future use.
- **DLC (4-bit)** - Data Length Code. Number of bytes of data being transmitted.
- **DATA (0 to 64 -bit)** - The actual data being transmitted.
- **CRC (16-bit)** - Cyclic redundancy check
- **ACK (2-bit)** - One bit is the acknowledgment bit and the second is a delimiter. When a node successfully receives a message, is acknowledgment bit is a dominant bit. the delimiter bit is a recessive bit.
- **EOF (7-bit)** - End Of Frame is a 7-bit field marks the end of a CAN frame.
- **IFS (7-bit)** - Inter Frame Space is the time required by the controller to move a correctly received frame to its proper position in a message buffer area.

2.1.2.2 Extended CAN:

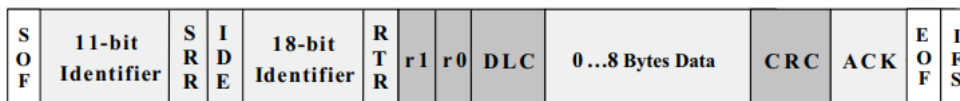


Figure 2.4: Extended

Is the same as Standard CAN with the addition of:

- **SRR (1-bit)** - Substitute remote request bit replaces the RTR bit in the standard message location as a placeholder in the extended format.

- **IDE (1-bit)** - A recessive bit in the identifier extension indicates that more identifier bits follow. The 18-bit extension follows IDE
- **r1 (1-bit)** - An additional reserve bit has been included ahead of the DLC bit.

2.1.3 Message types

There are four different message or frame types that can be transmitted over the CAN bus:

- **Data Frame** - This frame is the most common message type. Is composed by the arbitration field, the data field, the CRC field, and the acknowledgement field. The part that determines the priority of a message when two or more nodes are contending for the bus is the arbitration field. The arbitration field contains an 11-bit identifier for standard CAN messages and the RTR bit, which is dominant for data frames. For extended messages it contains the 29-bit identifier and the RTR bit. Next is the data field, here it contains up to eight bytes of data, The CRC field has a 16-bit checksum used for error detection. In the end there is an acknowledgement field. A CAN node that receives a correct CAN message sends a dominant ACK bit that overwrites the transmitted recessive bit. Then transmitter checks for the presence of the dominant ACK bit and retransmits the message if no acknowledge is detected;[\[31\]](#)[\[7\]](#)
- **Remote Frame** - This frame solicits the transmission of data of a specific node. The differences between this frame and the data frame are:
 - It's marked as a remote frame, RTR bit is recessive;
 - It doesn't have a data field.[\[31\]](#)[\[7\]](#)
- **Error Frame** - It's a special message because it violates the formatting rules of a CAN message. It is only sent when a node detects an error, and makes every node in the network to detect this same error and send the error frame;[\[31\]](#)[\[7\]](#)
- **Overload Frame** - This frame is similar to the error frame in terms of format. The overload frame is transmitted by a node that is too busy. It's used to have a delay between messages[\[31\]](#)[\[7\]](#)

2.1.4 Data transmission in CAN

A CAN network is composed of several CAN nodes, as we see in figure 2.1, and each node it's composed by three components:

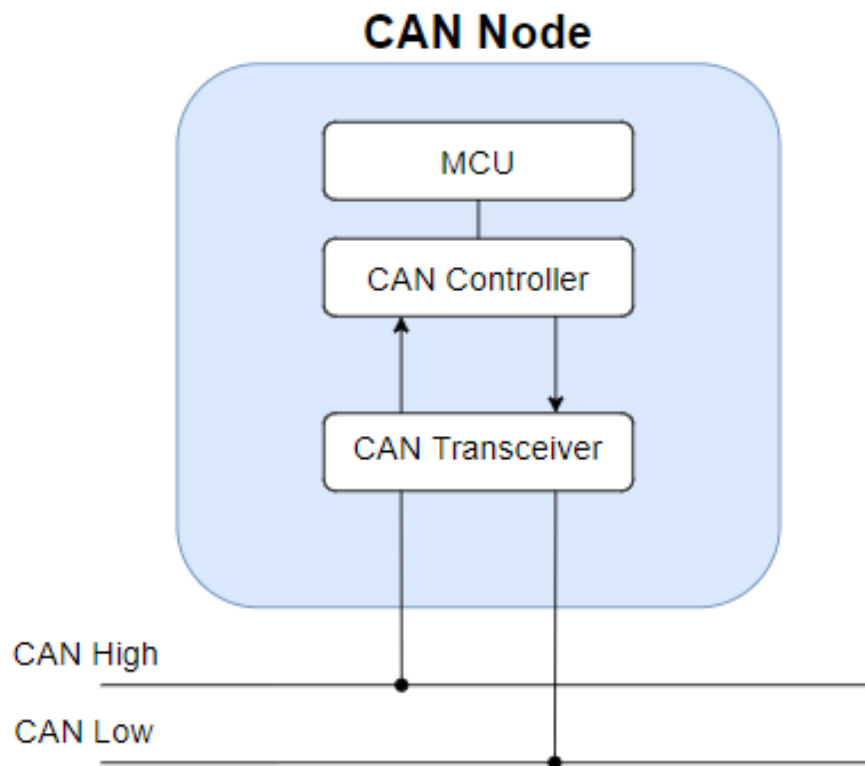


Figure 2.5: CAN node

- **MCU:** The microcontroller which is running the application to do something specific. It's the one who decides what to do with the received messages and what messages to send.
- **CAN Controller:** The CAN controller deals with the communication functions described by the CAN protocol, it activates the transmission and the reception of a CAN message.
- **CAN Transceiver:** The CAN transceiver is the interface between the CAN controller and the CAN bus. It converts or translates the logic level messages from the controller to the CAN bus, CAN high and CAN low, and vice-versa.

The CAN bus is where a CAN message goes through to go from one node to another, and it consists of two lines, CAN high and CAN low. The transmission of a message happens due to the differential voltage applied in this two lines. CAN receivers measure this differential voltage to determine the bus level. Figure 2.6 shows how a 0 and a 1 is transmitted through the CAN bus. A crucial feature of CAN is, the bus isn't actively driven during logic-high, referred as recessive. Both lines are at the same voltage, approximately $VCC/2$, so the differential voltage of both lines is zero volt, this zero volt will be read by the transceiver as recessive or logic 1. CAN bus is only driven during logic-low, referred as dominant, second third of figure 2.6. In this period the lines are driven so that $(CANH - CANL) \geq 1.5V$, this allows to have a differential voltage different from zero. It will be read by the transceiver as a dominant bit, logic 0.[5]

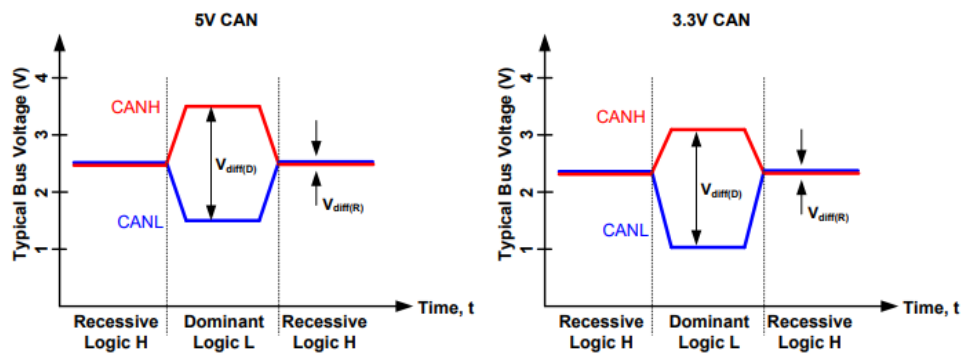


Figure 2.6: CAN voltage [5]

This allows a node transmitting a ‘High’ to detect if another node is trying to send a ‘Low’ at the same time. This is used for non-destructive arbitration, where nodes start each message with an address (priority code) to determine which node will get to use the bus. The node with the lowest binary address wins arbitration and continues with its message. There is no need to back-off and re-transmit like other protocols. As seen before the differential signal on a CAN bus can be one of two states, dominant (Logic 0) or recessive (Logic 1). If two nodes attempt to transmit at the same time, a dominant bit transmission will overwrite a concurrent recessive bit transmission, so all nodes must monitor the bus state while transmitting and cease transmission if overwriting occurs when they transmit a recessive bit.

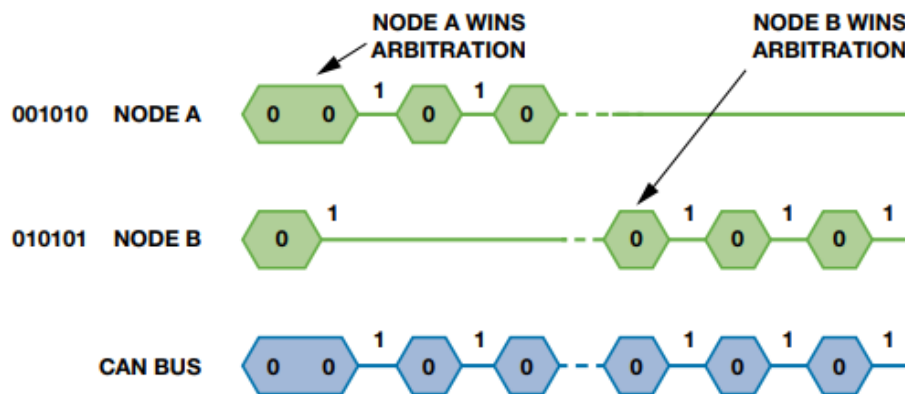


Figure 2.7: CAN nodes transmitting at the same time

2.1.5 Termination types

CAN bus must be terminated with resistors because a CAN network is a two-way communication. The termination makes sure the CAN signal is not reflected. This reflections can cause interference and damaged signals. ISO-11898 CAN standard defines that both ends of a CAN bus must be terminated with resistors that match the nominal impedance of the cable.[10]

2.1.5.1 Standard Termination

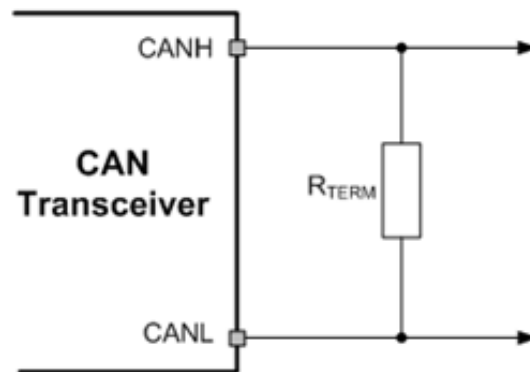


Figure 2.8: Standard Termination, [10]

This termination is composed by a single resistor between CANH and CANL, figure 2.8. This resistor matches the cable's impedance, which is usually 120Ω , between the CANH and CANL bus lines for each termination.[10]

2.1.5.2 Split Termination

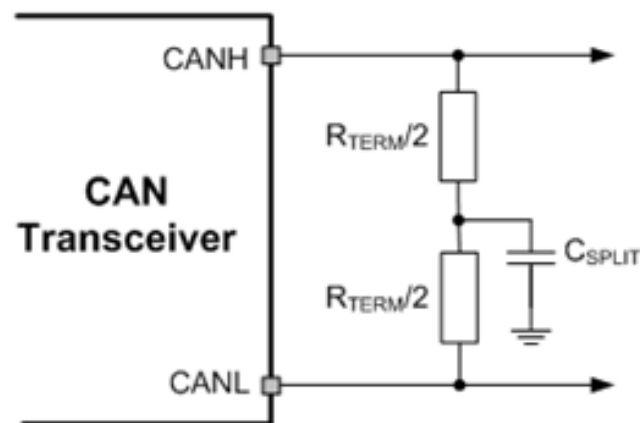


Figure 2.9: Split Termination [10]

This termination is composed by two resistors that are equal to one half the cable's impedance, usually 60Ω each, with a capacitor placed between, typically between 1-100nF, figure 2.9. The split termination uses more components, but it has the advantage of creating a low-pass filter which improves electromagnetic emissions.[10][26]

2.1.5.3 Biased Split termination

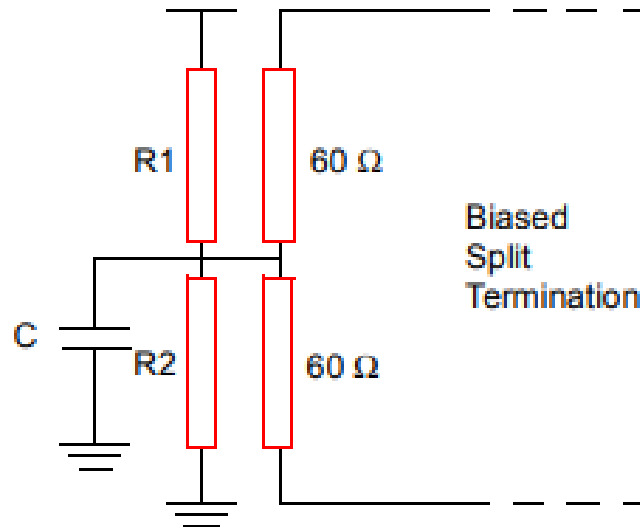


Figure 2.10: Biased Split Termination, [26]

This termination method is used to maintain the common mode recessive voltage at a constant value, thereby increasing EMC performance. This circuit is the same as the split termination with the addition of a voltage divider circuit to achieve a voltage of $V_{DD}/2$ between the two 60Ω resistors.[10][26]

2.2 Displays

In order to give the best possible way, for the user, to use and see the information that is being displayed different aspects have to be accounted in the display, such as, visibility, being water-proof, market availability, power consumption, refresh rate, etc. All of those are taken into account and compared in the table below.[30] [11] [12]

Display Types	Advantages	Disadvantages
LCD	-Consume little electricity and produce little heat -Screens are perfectly flat. -Available in many sizes. -Ability to last for a very long time.	-Viewing angle. -Poor reproduction of blacks -Limited brightness: LCD displays have lower contrast and are harder to view in a brightly lit room
TFT	-Minimal energy consumption. -Fast and accurate response time -Minimizes eye strain. -Good visibility and color.	-Relies on backlighting to provide brightness rather than producing its own light. -Disproportionate viewing angles - More energy consumption, compared to OLED and AMOLED.
OLED	-It provides remarkable color fidelity, high efficiency and operational stability. -OLED consumes less power and are suitable for devices requiring less power consumption	-It is expensive. -It can be easily damaged by water. -OLED screens are even worse compare to LCD when subjected to direct sunlight. -Limited market availability.
E-LINK	-It holds static text and images for indefinite time without any usage of electricity. -The display contrast is very good and can be viewed from wide viewing angles. -It is possible to read in direct sunlight as there is no glare in bright light condition.	-It has very low refresh rate compare to LCD display. -It can not be used in the dark
AMOLED	-It offers wider viewing angles which is about 170 degrees. -It offers super flexibility. -These screens are brighter than LED screens. -It resists instant pressure.	-It is higher in cost due to expensive manufacturing process. -It offers short lifespan than LED-LCD screens. -It gets easily damaged by water.

2.2.1 Touchscreen vs Buttons

There are 2 types of touchscreens, resistive and capacitive. The difference between them is that capacitive is much more sensitive, accurate, and has higher visibility but is more expensive.

The user interface is supposed to be the simplest possible, to give the driver has little instructions to use this interface. A touchscreen implies looking to the screen in order to touch in the

right spot, and if the truck driver is on the road this isn't advisable. With buttons, the driver can press the desired button even without looking and when possible visualize what he desires.

2.3 Bluetooth

Bluetooth is a standardized protocol for sending and receiving data via a 2.4 GHz wireless link. It's a secure protocol, and it's perfect for short-range, low-power, low-cost, wireless transmissions between electronic devices.[24][6][4]

2.3.1 Pairing

There are different ways in which Bluetooth networks can be set up. Bluetooth networks adopt what is called a piconet topology. In this type of network, one device acts as the master and it is able to talk to a maximum of seven slave nodes or devices.

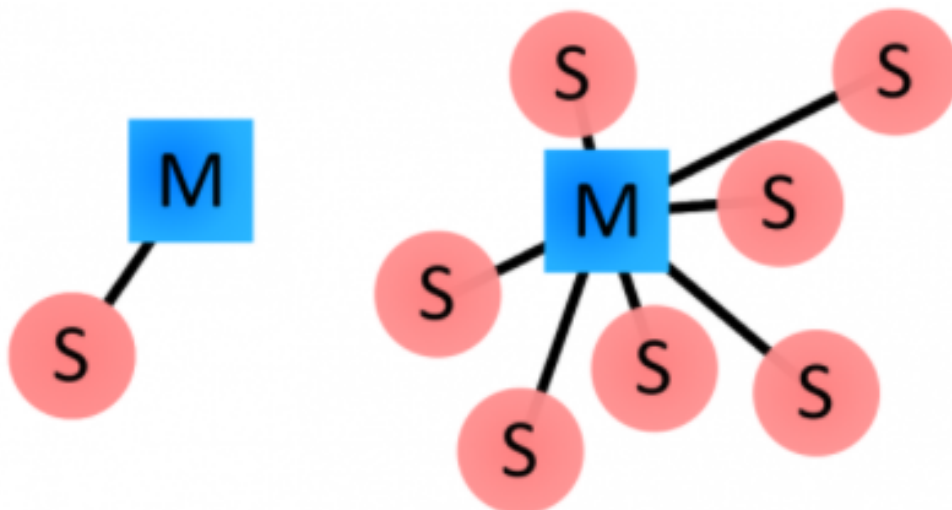


Figure 2.11: Piconet topologies, [6]

First I will explain different definitions in order to be more easily comprehended.

A paired device, a connected device, and an actively communicating device are different things. A device can be paired, but not connected. Similarly, it can be connected but not transmitting.

[24][6][4] There are the host devices (Master) and the client devices (Slave). We can think of PCs, mobile phones or tablets as the hosts and headsets, keyboards, controllers, mice as the clients. Like said before host devices support up to 7 simultaneously connected devices and an unlimited number of paired. The limit of 7 comes from the three bit address that is used. Client devices normally support a limited number of pairings, and only one single connection.[24][6][4]

Every Bluetooth device has a unique 48-bit address, the `BD_ADDR`. This will usually be presented in the form of a 12-digit hexadecimal value. The most-significant half, 24 bits, of the

address is an Organization Unique Identifier (OUI), which identifies the manufacturer. The other 24-bits, least-significant, are the more unique part of the address. Bluetooth devices can also have user-friendly name given to them. They are used instead of the address in order to be able to identify which device it is.[24][6][4]

In order to create a Bluetooth connection between two devices multiple steps have to be followed. These connections are maintained until they are broken, either by disconnecting the two, or because the devices go out of range of each other.

- **1-Inquiry:** If two Bluetooth devices don't know each other, one must run an inquiry to try and discover the other. One device sends out the inquiry request, and any device listening for such a request will respond with its address, its name and other information that may be relevant.[24][6][4]
- **2-Paging(connecting):** Paging is the process of forming a connection between two Bluetooth devices. Before this connection can be initiated, each device needs to know the address of the other, previous process.[24][6][4]
- **3-Connection:** After the paging process is complete, the device, enters the connection state. While connected, a device can either be actively participating or it can be put into a low power sleep mode.[24][6][4]
 - **Active Mode:** The device is actively transmitting or receiving data.[24][6][4]
 - **Sniff Mode:** This is a power-saving mode, where the device is less active. It'll "sleep" and only listen for transmissions at a set interval, example, at every 100ms.[24][6][4]
 - **Hold Mode:** The device sleeps for a defined period and then returns back to active mode when that interval has passed. The master can command a slave device to hold. Is a power-saving mode.[24][6][4]
 - **Park Mode:** The master can command a slave to "park", and that slave will become inactive until the master tells it to wake up.[24][6][4]

The Bluetooth pairing process is normally triggered automatically the first time a device receives a connection request from a device with which it is not yet been paired to. In order for that Bluetooth pairing occur, it requires an authentication process. It depends on the capabilities of the devices, it can be a password that has to be exchanged between the two devices, this password is a code shared by both Bluetooth devices, or it can be as simple as a click of a button to pair both devices. This is used to ensure that both users have agreed to pair with each other. Once the Bluetooth pairing has been established it is remembered by the devices, which can then connect to each without user intervention.[24][6][4]

2.3.2 Bluetooth versions

Bluetooth technology has evolved over the years. New versions have been developed over time and each new version supports new features. The latest versions of Bluetooth are all backward

compatible with older versions. This means a Bluetooth device using a recent version is able to connect to another Bluetooth device that supports an older version, but it will be limited to the features supported by the older version.

- **Bluetooth 1.0:** First one developed, although it was a groundbreaking discovery the technology suffered from many problems. Some of these problems were solved with the version 1.2 and features were improved like support for adaptive frequency-hopping spread spectrum (AFH) which reduced interference, faster transmission speeds of up to 0.7 Mbps, faster connection, Host Controller Interface (HCI), and Extended Synchronous Connections (ESCO);[24][6][4]
- **Bluetooth 2.0:** The highlight of this version was the support for EDR, Enhanced Data Rate which provides better data transfer speeds. A later version was released, 2.1, that supported Secure Simple Pairing (SSP) which improved the security as well as the pairing. This version was widely implemented and popular because of its simplicity, more range, now 33m instead of 10m, and data transfer speeds up to 3Mbps;[24][6][4]
- **Bluetooth 3.0:** This version introduced the High Speed mode (HS), data transfer speeds now can go up to 24 Mbps. It also established new features such as, Enhanced Power Control, Ultra-wideband, Unicast Connectionless Data and many more, but had a major setback, high power consumption;[24][6][4]
- **Bluetooth 4.0:** When released, this version brought the support for Bluetooth Low energy, BLE. It supported all the previous features of all other versions but the big change was in power consumption. The 4.1 version improved the user experience even more by allowing devices to support multiple roles simultaneously, and facilitating data transfer rates. Later it was released the 4.2 version that made the Internet of Things (IoT) possible;[24][6][4]
- **Bluetooth 5.0:** This is the latest version and it focuses on improving the connectivity and experience of the Internet of Things by offering a consistent data flow. In this version we are capable of data transfer at double the previous speed, 2 Mbps. The range over which signals can be transmitted between devices expanded from 60m, in the previous version, to now up to 240m. The most significant change in the 5.1 version is the use of different direction finding methods, Angle of Arrival and Angle of Departure. This allows the receiver and transmitter to know exactly where they are situated in relation to one another. Previously, the technology would have to essentially guess the distance between devices. The latest Bluetooth version to this date is 5.2, now, larger amounts of information can finally be transmitted over lower bandwidth and at lower power. Another feature that comes with it is the Enhanced Attribute protocol, which allows parallel transactions between two Bluetooth devices to occur simultaneously.[14]

Feature	Bluetooth 1.0	Bluetooth 2.0	Bluetooth 3.0	Bluetooth 4.0	Bluetooth 5.0
Range	Up to 10m(33ft)	Up to 30m(100 ft)	Up to 30m (100ft)	Up to 60m(200ft)	Up to 240m(800 ft)
Basic Rate (BR)	Yes	Yes	Yes	Yes	Yes
Enhanced Data Rate (EDR)	No	Yes	Yes	Yes	Yes
High Speed (HS)	No	No	Yes	Yes	Yes
Low Energy (LE)	No	No	No	Yes	Yes
Slot Availability Masking (SAM)	No	No	No	No	Yes

Figure 2.12: Bluetooth versions comparison, [4]

2.4 Bluetooth Low Energy

BLE, or Bluetooth Low Energy, was first introduced as part of Bluetooth 4.0 core specification. BLE was created to have some of the features that Bluetooth has, but focusing in low power. When we look to solutions with energy efficient radio modules and can still transfer data at a cheap cost of energy consumption, BLE stands out.

2.4.1 Protocol stack

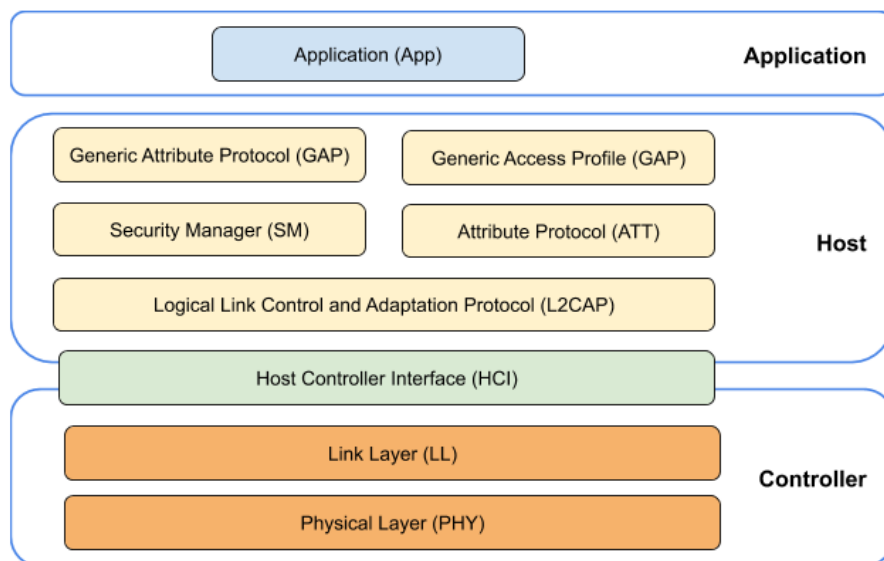


Figure 2.13: Bluetooth Low energy protocol stack, [8]

Bluetooth has several protocol layers and each layer has a different job. BLE layers are separated in three parts, figure 2.13. [8]

- **GATT:** This layer is responsible for defining the protocol of data transfer between two BLE devices. Uses the Attribute Protocol and defines types of attributes. Additionally, a group of attributes forms characteristics and a group of characteristics forms a service. This adds new functionalities;
- **GAP:** This layer is responsible for the connections and the advertising in Bluetooth. Like device discovery and connection establishment;
- **SM:** This layer is responsible for the methods and protocols of pairing and key distribution in low level security;
- **ATT:** This layer defines attributes, how they can be accessed and interacted with. In the top layers, data is defined by these attributes that include different functionalities;
- **L2CAP:** This layer is in charge of quality of service and routing of packets for higher layer protocols like ATT and SMP;
- **HCI:** This layer interfaces the host with the controller;
- **LL:** This layer interfaces with the PHY layer. It specifies how the physical channels are used as well as the packet format. It also handles the transmission of L2CAP packets;
- **PHY:** Is the bottom layer and specifies features such as, the number of RF channels, modulation and frequency. Is in charge of transmitting and receiving data over the air.

2.4.2 BLE packet

The BLE packet format is detailed in the figures below. This format is defined by the Link Layer. The data field can be different given the Bluetooth version:

- In Bluetooth versions 4.0 and 4.1 the maximum size of the data field was 27 bytes

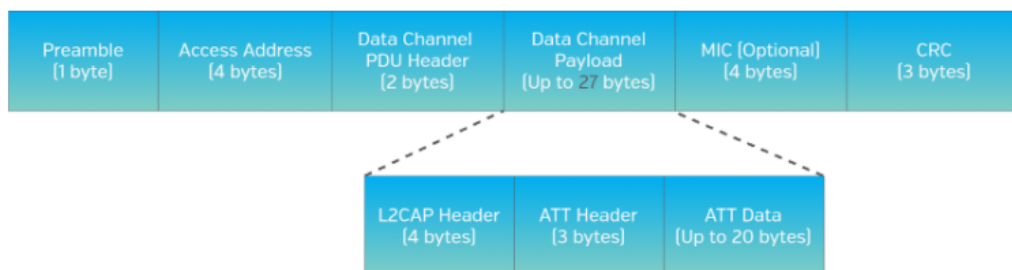


Figure 2.14: BLE packet structure[17]

- From Bluetooth version 4.2 and beyond, a new feature was implemented which increases the data field section

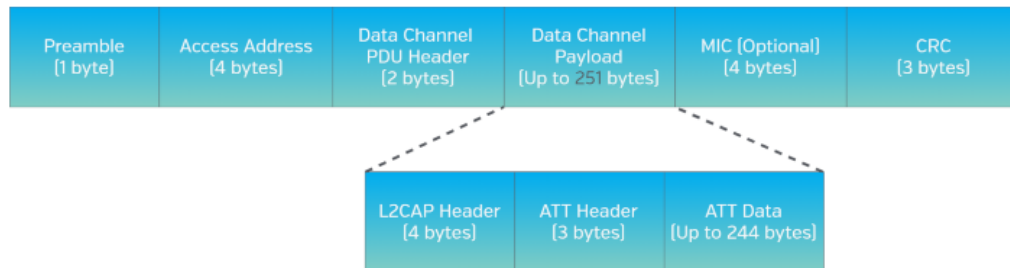


Figure 2.15: BLE packet structure[17]

2.4.3 BLE roles

A BLE device can have different roles[9]:

- **Central:** This role is responsible for scanning devices nearby. When a device is in central mode it is also responsible for initializing the connection;
- **Peripheral:** This role, when it isn't in already connected, sends advertisement packets in order to be visible to devices in central mode;
- **Broadcaster:** This role sends data by transmitting advertisement packets without the need of a connection to send data;
- **Observer:** This role only receives packets from a broadcaster device.

2.4.4 BLE data transfer

In BLE, data can be transferred when two devices establish a connection, a central device and a peripheral device. These two devices can send and receive data to each other on a specific time and channel. A connection event is when this two devices meet to send and receive data. The connection interval is the period of time between connection events. A connection event can continue until a packet fails to be received or the end of this interval has been reached. This way of data transfer is what gives this devices a low power consumption.[1]

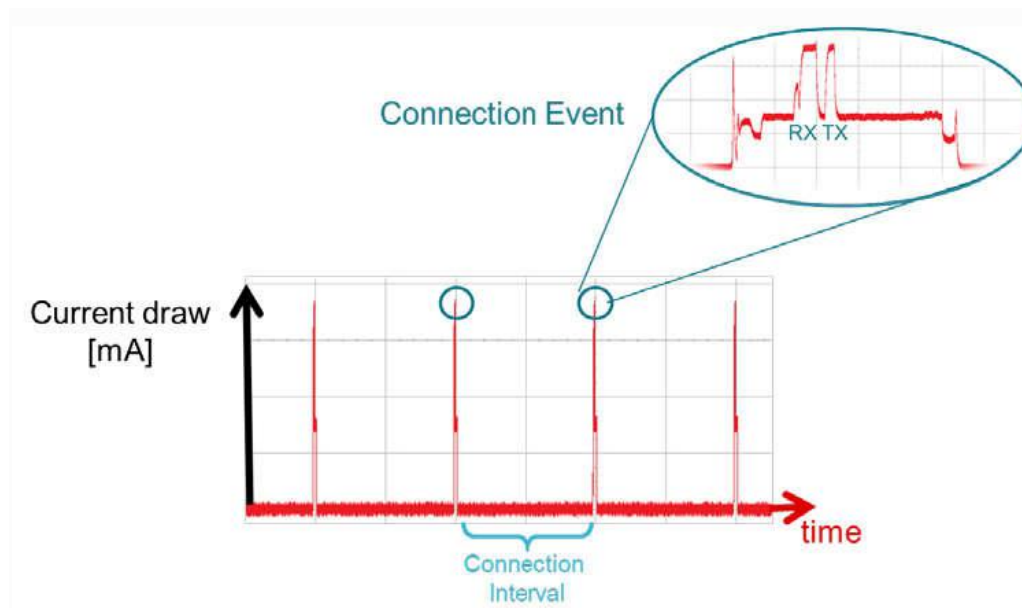


Figure 2.16: BLE connection interval and connection event[1]

2.5 XMC4300 Relax Kit

The XMC4300 Relax Kit is budget-priced evaluation board for the XMC4000 microcontroller family. It was created by Infineon, and has built on an ARM Cortex-M4. Allows the usage of IDE DAVE, the one used in this project. The main features of this board are the following:

- XMC4300 microcontroller (ARM® Cortex®-M4 based), 144 MHz, 256 kB Flash, 128 kB SRAM, LQFP100 package;
- On-board debug probe with USB interface supporting SWD + SWO;
- Virtual COM port via on-board debug probe;
- mikroBUS™ socket for plug-and-play click boards™;
- CAN transceiver and CAN connector (D-Sub DE-9);
- USB interface (Micro-AB USB plug);
- Voltage regulator 5 V -> 3.3 V;
- 2x user button and 2x user LED;
- Reset push-button;
- Arduino hardware compatible 3.3V and 5V pinout.

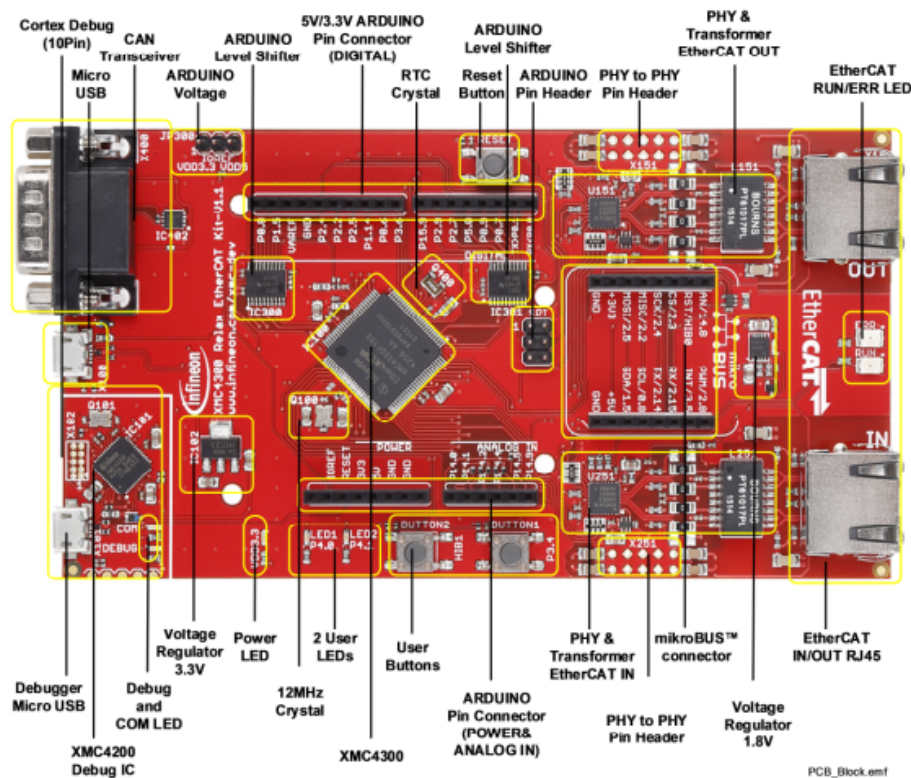


Figure 2.17: XMC4300 Relax Kit.[13]

2.6 Click boards

Click boards were created by MikroElektronika to give to anyone who wants to use any of these boards, the possibility to test any idea or concept, in their project, without the need of creating a whole new board and worrying about a new setup. To facilitate the integration of these boards, MikroElektronika uses a standard socket, mikroBUS, which allows any click to be connected to the microcontroller, figure 2.18.

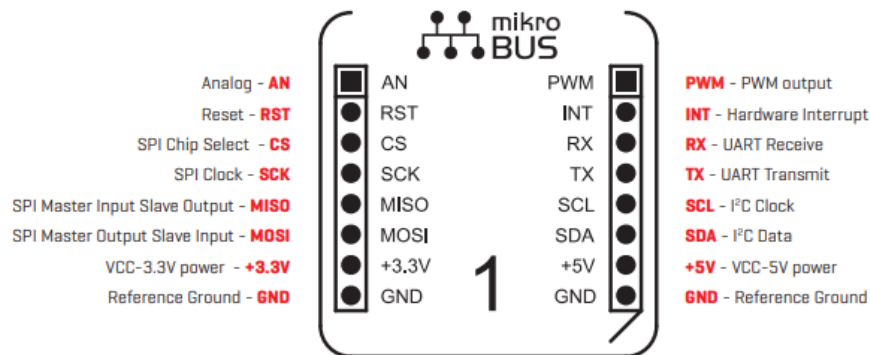


Figure 2.18: MikroBUS

A mikroBus socket is made up of a pair of 1x8 headers. As shown in the figure above, the pinout has three different types of communications, SPI, UART and I2C, six additional pins, PWM, interrupt, analog input, reset and chip select, and two power groups, 3.3v and 5V as well as the ground for each one.

2.7 PCAN-View & PCAN-USB

In order to test and monitor the data flow on a Controller Area Network (CAN), PCAN-View and PCAN-USB were used. PCAN-View it's an easy to use program that allows to send , and monitor any CAN messages. It is developed by PEAKSystem Technik GmbH and has the following features:

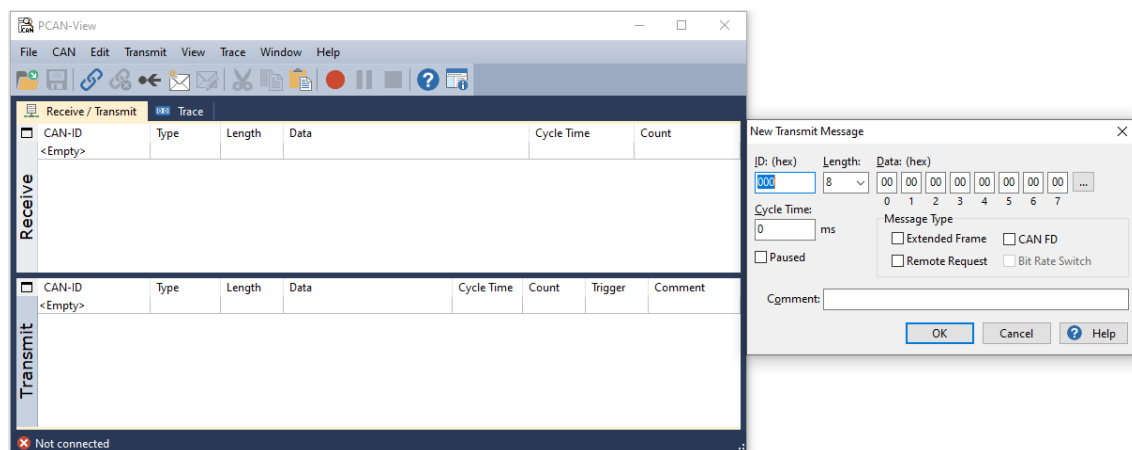


Figure 2.19: PCAN-View

- Can transmit and receive CAN messages, including Remote request and Error frames;
- Any number of frames can be emitted, either manually or automatically in fixed time.

- Displays each CAN message with its ID, data length and data bytes;
- Supports both CAN specifications 2.0 A/B;
- Supports bit rates up to 1Mbit/s.

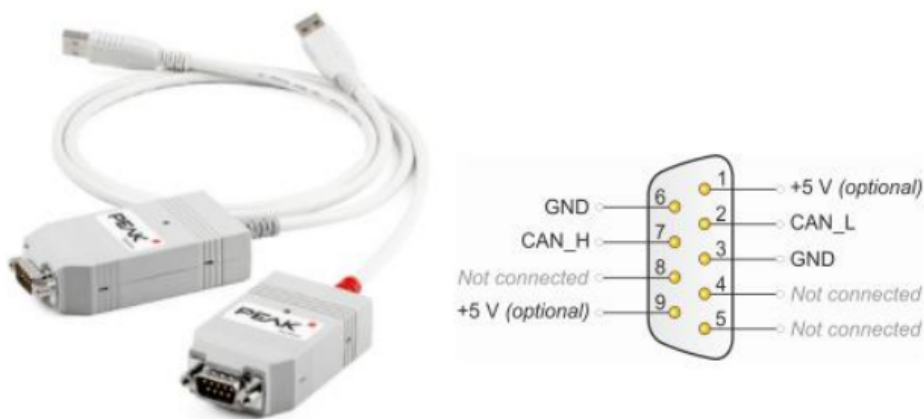


Figure 2.20: Hardware, PCAN-USB and pin assignment

- Has an adapter for the USB connection;
- Establishes CAN connection according to ISO 11898-2;
- Bit rates up to 1 Mbit/s;
- Includes a NXP SJA1000 CAN controller and a NXP PCA82C251 CAN transceiver;
- CAN bus connection via D-Sub, 9-pin. Figure above shows the pin assignment.

2.8 Communication Protocols

- **SPI** - Serial Peripheral Interface (SPI) is a synchronous serial communication interface best-suited for short distances. Because there is no protocol overhead, the transmission runs at relatively high speeds. SPI runs on one master, with one or more slaves, usually the devices outside the central processor. The master clock signal shifts data out of the master and out of the slave at the same time. SPI is a great option for small, low-resolution displays and small LCDs.[16]

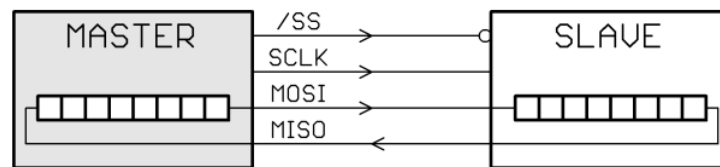


Figure 2.21: SPI[16]

Minimum with 4 signals:

- **SCLK**: Serial Clock;
- **MOSI**: Master Output, Slave Input;
- **MISO**: Master Input Slave Output;
- **SS**: Slave Select;

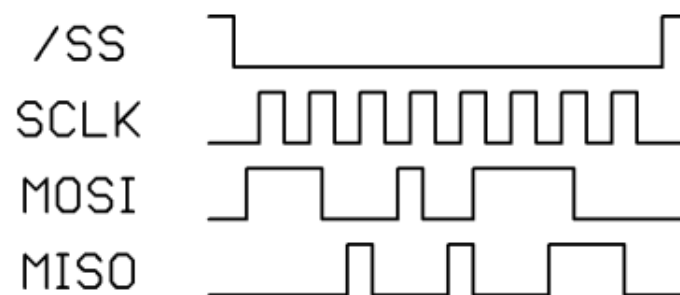


Figure 2.22: SPI timings[16]

Data is shifted out of the master and of the slave at the same time, this can only happen when the SS signal is pulled down, as shown in figure 2.18. During this time after n clock pulses, master and slave swap the contents of their inter n -bit registers. Depending on how it's configured, SCLK can start at one or at zero, data can be sampled on the rising edge or the falling edge or SCLK and the bits being transmitted can start by the most significant bit or the least significant bit.[16]

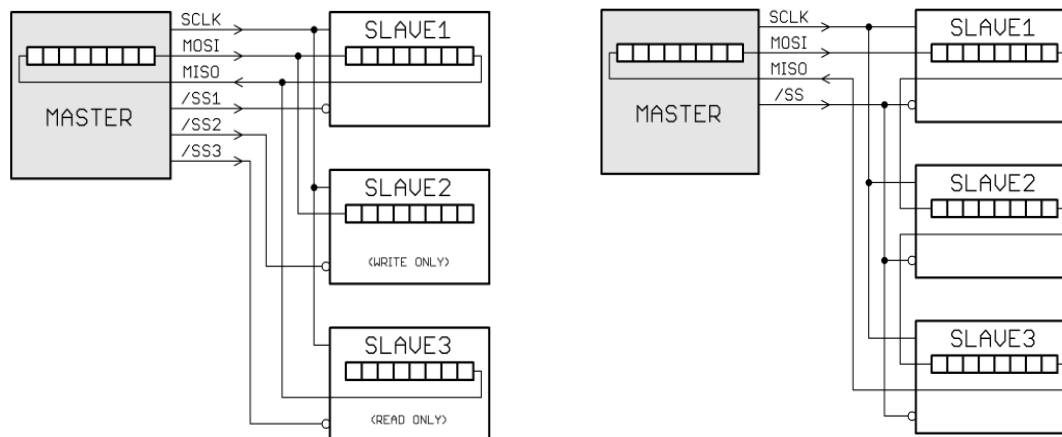


Figure 2.23: SPI topologies: Left figure - Standard; Right figure - Ring or daisy chain[16]

There are two types of SPI topologies, figure 2.19:

- **Standard:** Works with any type of slave, needs on SS signal for each slave connected. This topology simplifies the software but needs more signals;
- **Ring or daisy chain** One common SS signal, all slaves must have MOSI and MISO signals, uses only four signals regarding the number of slaves. This topology needs a more complex software.
- **I2C** - Inter-integrated Circuit (I2C). It utilizes a multi-master, multi-slave, single-ended, serial computer bus system. Engineers developed I2C for simple peripherals on PCs, like keyboards and mice to then later apply it to displays. Like SPI, it only works for short distances within a device and uses an asynchronous serial port. What sets I2C apart from SPI is that it only requires two wires, serial clock (SCL), and serial data (SDA). Like SPI, I2C also works well with smaller LCDs.[2]

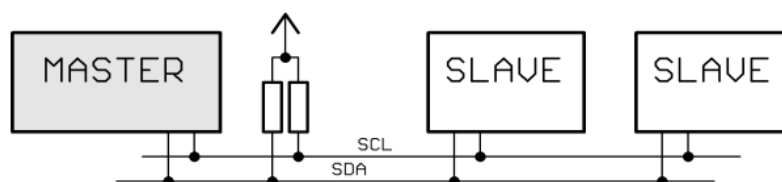


Figure 2.24: I2C[3]

Any device has its unique address and can be, a transmitter, a receiver, or both. The master always initiates the data transfer and every other device is considered as a slave.

There can be more than one master on the bus. Figure 2.20 shows how the data transfer in I2C is conducted[2]

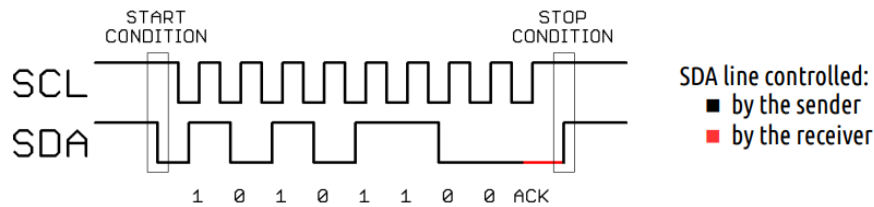


Figure 2.25: Data transfer in I2C[3]

- First, the start condition, to happen a falling edge of SDA with SCL=1 must happen;
- One byte is transferred, data, by the the master, and after recognition the slave will send a ACK to the SDA line. SDA never changes while SCL=1;
- Last, the stop condition, to happen a rising edge of SDA with SCL=1 must happen.

Technology	# pins	Applications	Speed	Benefits
SPI	4 or more	- Small displays on MCU's.	- Up to 60 Mbps (short distances).	- Ease of set up; - Faster than I2C.
I2C	4	- Small displays on PC's and PC peripherals.	- Standard speed: 100 kbit/s; - Full speed: 400 kbit/s; - Fast mode: 1 Mbit/s; - High Speed: 3.2 Mbit/s.	- Low energy consumption; - Resistance to noise; - Handles a broad range of operating temperatures; - Ease of use and troubleshooting.

- **UART** - Stands for Universal Asynchronous Receiver Transmitter. It is a form of device-to-device digital communication. UART supports bi-directional, asynchronous, no clock, and serial data transmission. It has two data lines, one to transmit data, TX and another to receive data RX. Once connected data flows from TX of the transmitting UART to RX of the receiving UART[25]. It can operate in three different ways:

- **Simplex:** data transmission in one direction;
- **Half-duplex:** data transmission in either direction but not simultaneously;
- **Full-duplex:** data transmission in both directions simultaneously.

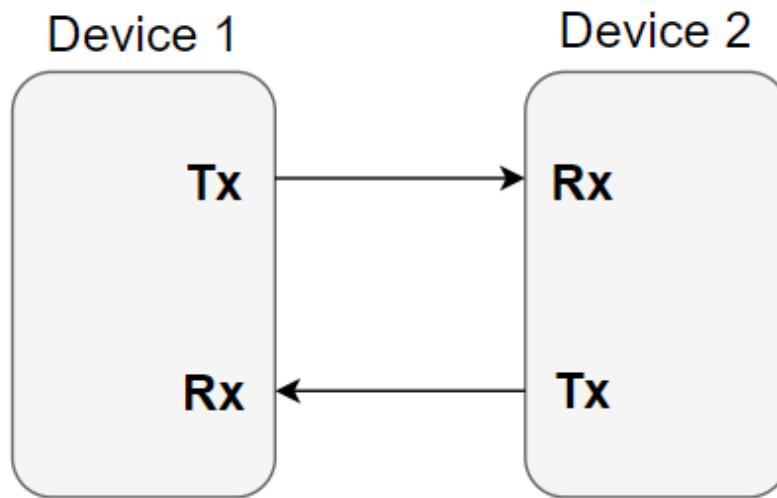


Figure 2.26: UART connection

UART data transmission speed is referred to as baudrate. When the receiving UART detects a start bit, it will read the bits at the defined baudrate. After the starting bit follow five to nine data bits known as the packet's data frame, then an optional parity bit to verify proper data transmission. Finally, one or two stop bits are transmitted, this ends a packet.

Start bit	Data frame	Parity bits	Stop bits
1 bit	5 to 9 Data bits	0 to 1 bits	1 to 2 bits

Figure 2.27: UART message

UART advantages:[33]

- Simple to operate and easy to set up;
- No clock needed;
- Has a parity bit to allow for error checking;
- Well documented and widely used method;

UART disadvantages:[15]

- The size of the data frame is limited to a maximum of 9 bits;
- Doesn't support multiple slave or multiple master systems;
- Is not suitable for heavy lifting serial communication during high energy consumption;
- Baud rates of each UART must be within 10% of each other to prevent data loss.

Chapter 3

Requirements analysis

In this chapter, the current solution will be evaluated by doing an overview of the technology and highlighting the key parts. It will also gather the requirements that will be decisive for the future solution.

3.1 Problem Description

3.1.1 Current technology



Figure 3.1: Addvolt current solution

In the picture above we can see the present solution implemented in the trucks that support Addvolt technology. This interface is connected with the CAN bus, this way it can receive all the information from the Addvolt system. The user interface is done through buttons. Each button

represents a variable and this variable is identified by a figure placed, and glued, to the button. Every button can use colours to light up its figure. The current solution can:

- Describe the current state of charge of the Addvolt battery;
- Indicate if the system is ON or OFF ;
- Indicate if load is detected;
- Indicate if grid is detected;
- This solution also as the capability of detecting; failures that may occur. They are displayed by using different colours in the specific button.

3.1.2 Main Problems

3.1.2.1 Information

One of the main purposes to have this technology is to be able to know the current state of charge of Addvolt battery. As we have seen in the current user interface, this feature can only be achieved by the use of figures, where only one will light up suggesting the battery percentage. The actual percentage can't be read. Just to give an idea of the charge the battery is currently in, we have to use four buttons. The other three buttons are being wasted.

Although the use of figures can be more intuitive, there is a limit to which the displayed information goes. It can't show quantitative information and because there is a limit number of buttons it gives few alternatives to represent different variables.

3.1.2.2 User

After having the possibility of going to visit and see an Addvolt system implemented in a truck, I also had the opportunity to speak to the truck drivers which deal with this interface in a daily basis and find out what are the negative aspects they find in this interface. This way we can have a better understanding of the aspect that need improvement.

- The first struggle that everyone pointed out was, in day time, when sunlight is hitting the interface directly, it's very hard to distinguish which buttons have their light ON. As we know, the only way the interface gives information to the user is by lighting the buttons. If the driver fails to distinguish which buttons are being lighted the interface fails its purpose.
- The second struggle was not being able to know the autonomy of the battery. This is very important to the driver, because it determines if it has to stop to re-charge or it can continue to the next station. In the way that currently it's design the driver has to guess the current percentage and this can lead up to bad decisions and future system failures. The autonomy is a new variable to give to the user, but in the current user interface there is no room to add new variables.

3.1.2.3 Trucks with trailers

First there is a need to know that exists two types of trucks that will impact the problem and consequently the solution.

- Trucks can have trailers and this type is when the truck is divided in two, the tractor, where there is the engine, and the trailer, that attaches to the tractor and where all the cargo and what is being transported is.



Figure 3.2: Truck with trailer example

- Or it can be a normal truck where everything is connected.

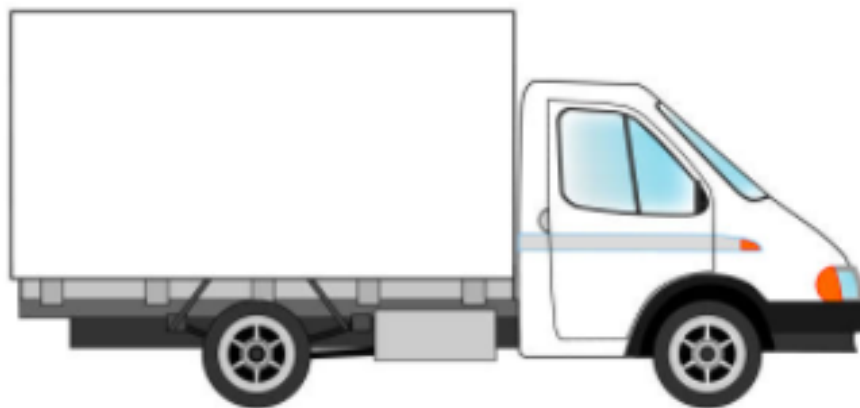


Figure 3.3: Truck example

The trailer, where the Addvolt system is connected and placed, can be used by several tractor units, and different truck drivers. This implies that if a trailer is used, the user interface can't be

within the driver cabin, but has to be outside. Not only the interface is outside, but also there is no way of notifying the driver if some problem happens or some alert is triggered. For example if the battery is close to being completely discharged. The only way to monitor the Addvolt system is if the truck is stopped, it can't be done while driving.

3.1.2.4 Debug and service

To perform maintenance, a routine inspection or when problems arise and the system needs to be checked, all of this tests are carried out and observed in the Addvolt application, from a computer. In order to connect to the Addvolt system there is the need of having a series of cables, adaptors that aren't cheap. As the number of client increases and the number of systems being produced increases accordingly, the amount of this adaptors for clients and for Addvolt increases.

3.2 Proposed solution

After all the problems were identified, the aspects that we have to improve and the new tools that we have to add, the next step is to develop the desired solution. Initially it will be revealed a new solution proposal and afterwards the requirements that validate the solution will be presented.

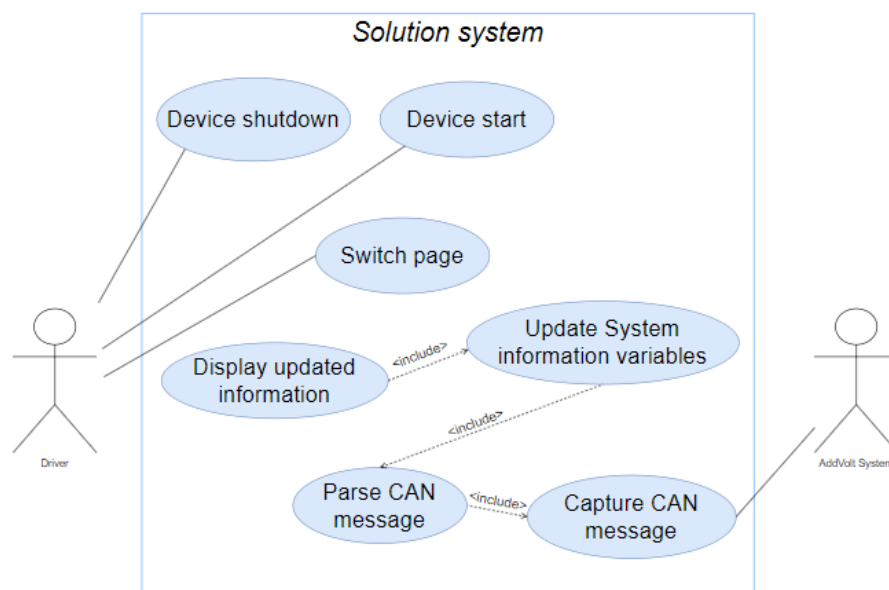


Figure 3.4: Solution use case diagram

Despite the problems are more evident in trailer solutions, this proposed solution can also be designed to replace the current truck solution. In this order we address the problems and maintain the coherency among different products. We know for either solution the main objective is to give the user all the information he needs in a objectively and simple way. What is similar to both

solutions is the incorporation of Bluetooth with the display. This technology will make it possible to eliminate the use of adaptors and allows the communication between Addvolt application and Addvolt system wireless. Also the new user interface will be a display screen instead of buttons. This way it can be used to display all the information desired, without restrictions of variables. It gives the possibility of having a more detailed and complete description of all variables and alerts. In addition, the capability of having different pages, each one with a different purpose, such as for debugging and maintenance, being these pages more orientated to engineers.

First solution: In case of a truck with a trailer, the display interface has to be placed outside of the truck. This solution needs to be more robust and has to be able to endure adverse environments. First as it is outside we have to ensure that the display can be read in all situations during a day, since this detail was particularly pointed from the truck drivers, especially during day time. Then, the interaction with the display interface has to be simple and objective. The driver needs to be able to see and find the information of the Addvolt system instantly. In order to interact with the device the use of push-buttons give a more secure way of interaction, because of their robustness, buttons aren't damaged by climate changes. A example is if its raining, a touchscreen can assume a different position in the screen of where was actually pressed. When a user presses a button it knows it was pressed. Also, buttons when labeled correctly give a straightforward and simple understanding of their function and use.

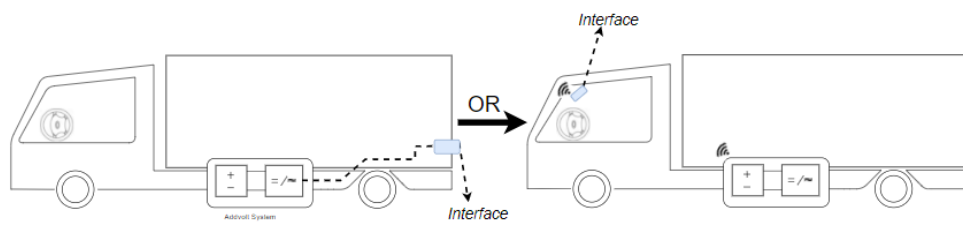


Figure 3.5: User interface in trucks with trailer

However, with Bluetooth integration the possibility of having this interface in the drivers cabin, and not outside, is now possible. The new system has integrated a Bluetooth device. Because we are talking about a truck with a trailer, and now the user interface being in the drivers cabin, it isn't connected to the CAN bus to extract the latest data that will update the display information. Instead Addvolt system will also incorporate a Bluetooth device and send the CAN messages, with the updated data, to the user interface system which will then update and show the new information.

Second solution: This solution appears when a truck isn't divided in tractor and trailer. Which means the display interface doesn't necessary need to be outside, it can be in the drivers cabin. This solution can also be a possibility when, as already described in the first solution, with Bluetooth integration the display interface isn't outside. Here, the display doesn't need to be as robust, but it will also be affected by sunlight, although not being as impactful. Because it is inside the option of having a touchscreen display is considered, which is only natural, is the next step

in technology, but it isn't necessarily the best. When driving, and if the driver wants to use the interface, being a touchscreen, implies looking to the screen to select the correct page or menu, a false touch in the screen will distract the driver even more because now he has to go back and select the correct page. With buttons even by touch and memory we now which one we are selecting. This are all scenarios we have to take into account.

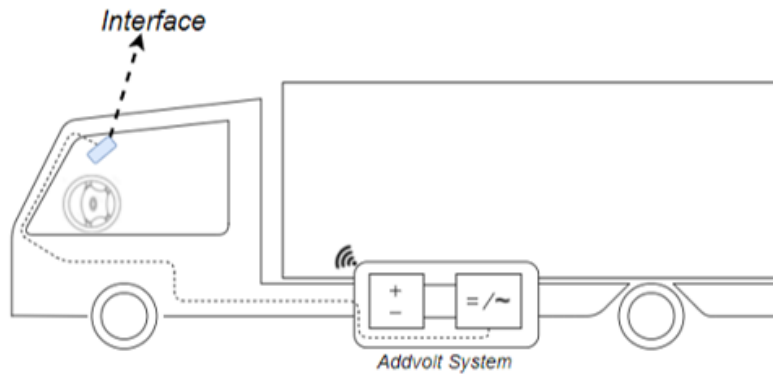


Figure 3.6: User interface with normal truck

3.3 Requirements

After the major functionalities intended to the new solution, and the key points that must have to be at least good enough that it can substitute the previous version, were collected, a list of requirements was elaborated. The list that fallows goes from high to low priority.

3.3.1 Device Communicates via CAN

Requirements	Test Procedure
- The device can receive/send standard and extended frames at a speed up to 1Mbps.	- When device sends start message to system, system must recognize it without need for additional configuration in system.
- (Optional) The CANOpen protocol can be at least partially supported.	- Functionalities of CANOpen are used.

3.3.2 Display shows user system status

Requirements	Test Procedure
- User must be able to see at least if system is ON/OFF, Battery state of Charge, if grid is connected to the system and if load is ON.	- Display shows unequivocally if system is ON/OFF. - Display shows the Battery State of Charge. - Display shows if grid is connected to the system. - Display shows if load is ON.
- Display must be easily visible in different conditions both during day and night.	- Information is visible with different light conditions (of day and night).
- (Optional) Temperature of chamber and other additional variables/information can be displayed.	- Display shows temperature and the additional variables defined.

3.3.3 User can interact with the device

Requirements	Test Procedure
- User can interact with the device by pressing its buttons.	- When each button is pressed the state of the button can be seen with the debugger.
- At least the same functionalities as the previous devices are required – turn ON/OFF and awake from sleep.	- Upon pressing the button ON/OFF the user interface turns ON/OFF.
- (Optional) A button can have more than one functionality.	- When pressed a button and the page refreshes the same or other buttons have other functions.
- (Optional) The device can display additional information about the system.	- Additional information about the system is displayed.

3.3.4 Device responds in real-time to user inputs

Requirements	Test Procedure
- Upon pressing any button, the response time is adequate.	- After clicking a button to start the system, command must be issued in CAN bus with a maximum 100ms delay.
- Display is responsive to system data Changes.	- Displaying a counter incrementing at each millisecond, the difference between 2 consecutive displayed values must be less than 1000 ms.

3.3.5 Device can be debugged and updated via CAN

Requirements	Test Procedure
- Adapt proprietary Addvoltage bootloader to device.	- Change FW of device using the CAN line only (no direct programmer connection).

3.3.6 Prototype is approved

Requirements	Test Procedure
- To be approved the device must have realistic assembly process, follow the dimensions guidelines and be usable/user-friendly.	- Total dimensions of prototype (without enclosure): max 8cm x 15cm x 5cm. - Display area: min 20cm² - Device must be waterproof or, if fit inside waterproof case needs to be operable (eg. through a membrane that lets buttons be pressed). - 120€/unit or less for 10 units (without case).

Chapter 4

Implementation

In this chapter it will be addressed the way this project was conducted in order to ensure that a final solution was reached.

To be able to fully benefit from the usage of a XMC4000 family microcontroller, Infineon offers the possibility to use, completely free, Eclipse based IDE DAVE. DAVE is a high-productivity development platform for the XMC microcontroller families to simplify and shorten software development. With DAVE, developers can generate the software library to efficiently use the innovative application-optimized peripherals such as, PWM, SPI, CAN, UART. Code generation is based on pre-defined and tested application-oriented software components, called DAVE APPs. DAVE Apps are autonomous building blocks that can abstract any simple or complex use case. DAVE provides the following enhancements:

- Faster and more robust design;
- DAVE APPs are based on low level peripheral drivers (XMC Lib), which makes generated code more efficient and more readable;
- A set of low level peripheral drivers (XMC Lib) is available and is used by DAVE APPs. The XMC Lib can also be used independent of DAVE or DAVE APPs with any other tool chain that supports XMC.

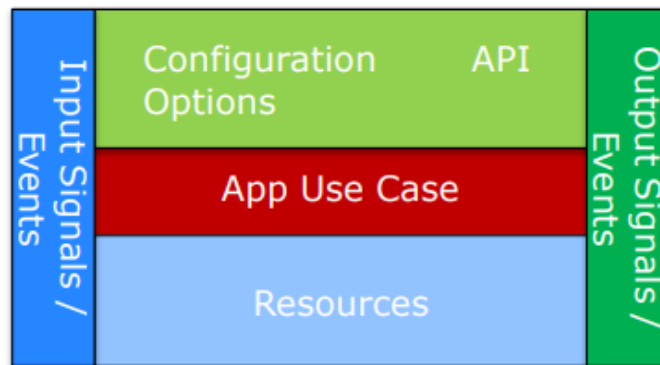


Figure 4.1: DAVE Apps

- Configuration, for example to determine the initialization status;
- API used in the user code (like a library);
- Output Signals that can be used to trigger, or enable a functionality of another DAVE App;
- Resources that are required for the App Use Case, can be chip resources or an API of another DAVE App;
- Input signal / events the triggers, enables or disables a functionality.

4.1 Displays

A very important aspect of this project is how to display and show the information to the user. Therefore, given the study already done in the literature review, the problems described and the solution proposed, initially we will be taking advantage of the click modules compatible with our microcontroller. This way we can assess if the type of displays selected can be a potential solution for our problem in hands. After analysing the solutions that click modules offer in this area and taking into consideration the cost and the features of the displays, a LCD click module and an OLED click module were selected.

4.1.1 LCD Click

One of the identified problems was the inability of displaying descriptive information as well as, when in direct sunlight, the user couldn't tell what information was being given to him. A LCD display is a good solution for the limitations listed. With this module we will try to display any information. In the end every feature a LCD display has must be mastered.

4.1.1.1 Overview

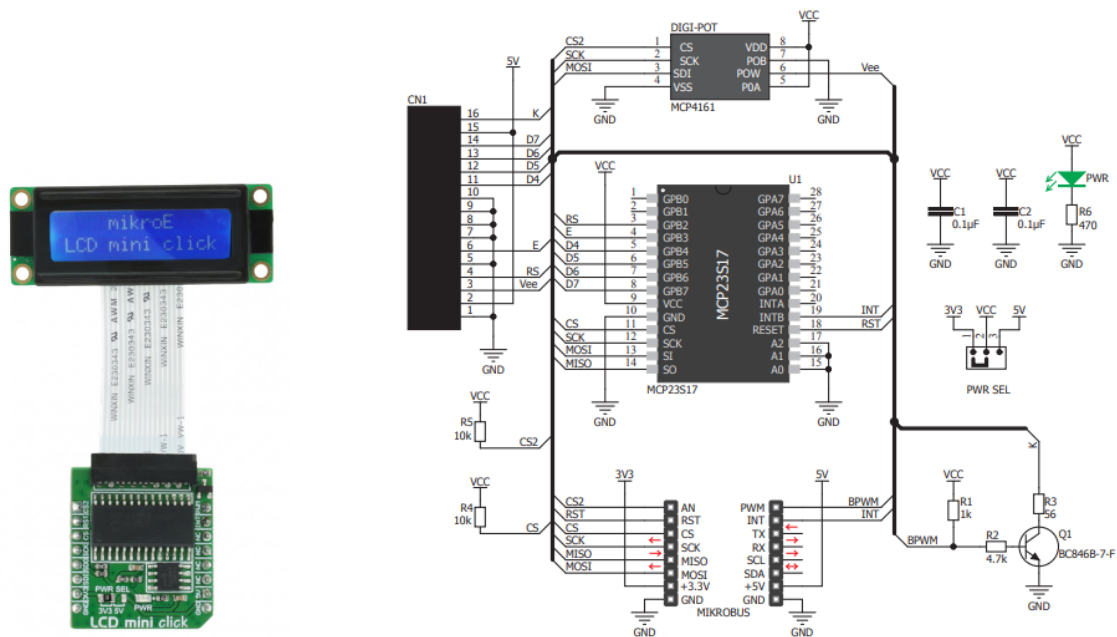


Figure 4.2: LCD click module & LCD click schematic[20]

With both figures we can see we used a 16x2 LCD display, LMB162XFW. The click, as presented in the schematic, has a port expander, MCP23S17 which is the main focus in order to program the LCD as we desire. MCP23S17 has a SPI interface, it consists of 8-bit configuration registers, for input, output and polarity selection and has 16-bit I/O ports divided in half, GPA and GPB.

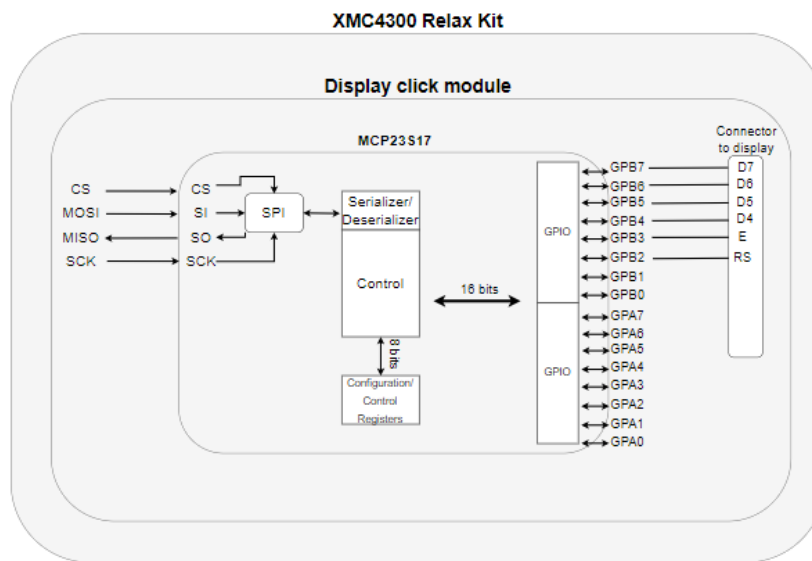


Figure 4.3: Interactions

By analysing the schematic we can see the fifth pin, of CN1, the display connector, is connected to the ground. This pin corresponds to the R/W pin, and when pulled to logic 0 performs a write operation. We conclude in this module the only operation we can do is to write to the display. The sequence to make this operation is as it follows: First the opcode byte, then the register address byte and then the data that we want to send, figure 4.3. The device address also can be seen in the schematic. We can observe that all three pins are connected to ground which means A2, A1 and A0 are all "0". For the register address we only have to take into consideration the IODIR and the GPIO registers, where the first one defines the pins of the MCP23S17 as inputs or outputs, which we want as outputs, and the second writes the desired value on the pin. Take into account that whenever data is being sent, chip select must be pulled down.[18]

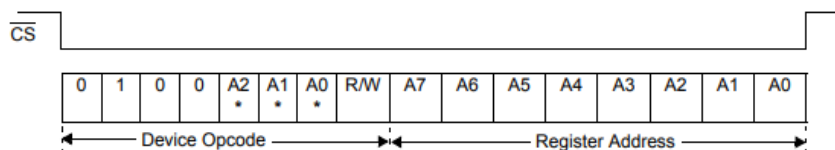


Figure 4.4: SPI addressing registers[18]

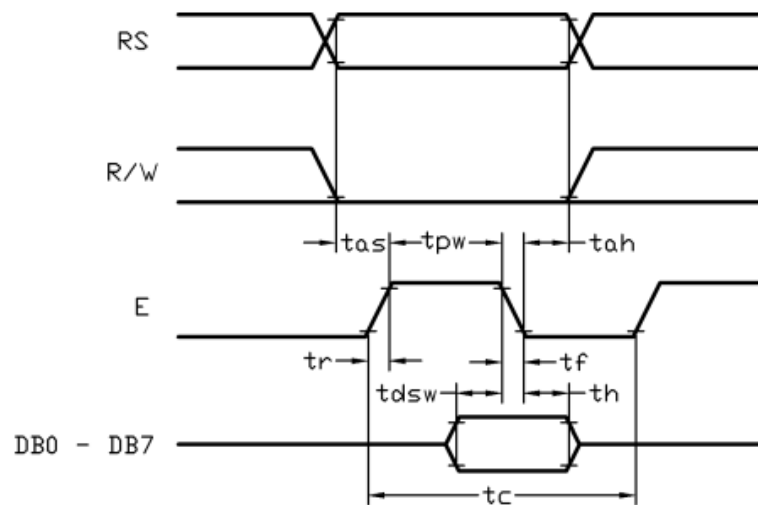


Figure 4.5: Write timing diagram[18]

To perform the correct transmission of data certain timings must be fulfilled, figure 4.5. Because R/W is always "0", as stated before, we can only perform write operations, but there are two types of writes, Instruction write operation and Data write operation. To perform an Instruction write both RS and R/W must be pulled down, to perform a Data write operation RS must be a logic 1 while R/W is logic 0. The difference between them is, we use Instruction write for defined instructions as clearing the display or setting the cursor in determined position, and several more as we will see ahead, Data instruction is used to access an internal memory in which all characters that can be used to display is accessed.[18]

b7-b4 b3-b0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
CG RAM (1)				0aP`P							-	9	3	α	p	
0001 (2)			!	1AQa9							.	7	チ	4	ä	q
0010 (3)			"	2BRbr							「	イ	ツ	×	β	θ
0011 (4)			#	3CScs							」	ウ	テ	ε	ε	∞
0100 (5)			\$	4DTdt							、	エ	ト	ト	μ	Ω
0101 (6)			%	5EUeu							・	オ	ナ	1	ε	Ü
0110 (7)			&	6FVfv							ヲ	カ	ニ	ヨ	p	Σ
0111 (8)			'	7GWgw							フ	キ	ヌ	ラ	g	π
1000 (1)			(	8HXhx							イ	ク	ネ	リ	フ	Σ
1001 (2)			)	9IYiy							ウ	ケ	ル	ル	”	y
1010 (3)			*	:JZjz							エ	コ	ン	レ	j	≠
1011 (4)			+	;K[k<							オ	サ	ヒ	ロ	×	π
1100 (5)			,	<L¥11							カ	シ	フ	ワ	φ	π
1101 (6)			-	=M]m>							ユ	ズ	ヘ	ン	も	÷
1110 (7)			.	>N^n÷							ヨ	セ	ホ	”	ñ	
1111 (8)			/	?O_O←							ッ	ソ	マ	”	ö	■

Figure 4.6: List of characters available[28]

Instruction	Instruction Code										Description	Description Time (270KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM, and set DDRAM address to "00H" from AC	1.52 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.52 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 us
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	37 us
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	37 us
Function Set	0	0	0	0	1	DL	N	F	x	x	DL:interface data is 8/4 bits N:number of line is 2/1 F:font size is 5x11/5x8	37 us
Set CGRAM address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	37 us
Set DDRAM address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	37 us
Read Busy flag and address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 us
Write data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	37 us
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	37 us

Figure 4.7: List of instructions available[28]

4.1.1.2 Initialization

As we understood how to communicate with LCD display, after being powered, there has to be an initialization process so that we can make use of the display, so we can start to write anything in the display. First we must again look at the schematic of this module and see that only D7 to D4 pins are being used, this has to be taken into consideration because this chip has two kinds of interface, 4-bit and 8-bit, the way our connections are, the interface used is 4-bit. Knowing this the initialization steps, for a 4-bit interface, are the following:

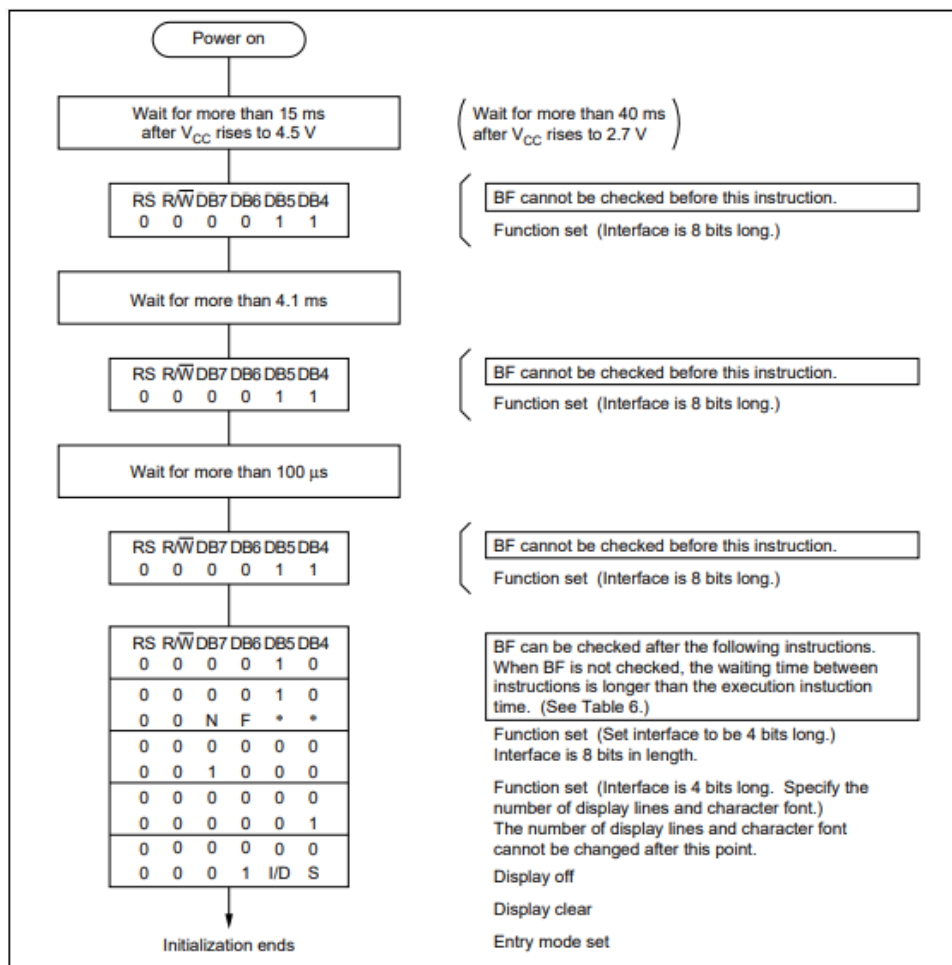


Figure 4.8: Initialization process[28]

During the process of initialization, the final 4 commands sent, which in reality are 8 because of the 4-bit interface, first we send the MSB then the LSB, always respecting the timings. First, establish the number of lines, 1 or 2 lines, for our display and the size of the characters, 5x8 or 5x11, we clear the display, which deletes everything that the display has written and puts the cursor in the first position of the screen, then we establish which way the cursor moves to write the next character. After this process we should see the cursor in the first position of the display, figure 4.8



Figure 4.9: After initialization

After initialization is complete we can now write any information we want, do not forget however in order to write a character in the display we must access it's internal RAM and select the character desired, figure 4.5, and for this we must put pin RS in logic 1, after selecting the character, it will be drawn in the position that the cursor is in, this case the first, 00h, and after writing, the cursor will shift one position to the right as specified in the initialization. If we want to write in any position of the display, we first need to set the cursor to that position, instruction set DDRAM address, and then write to the internal RAM as before.

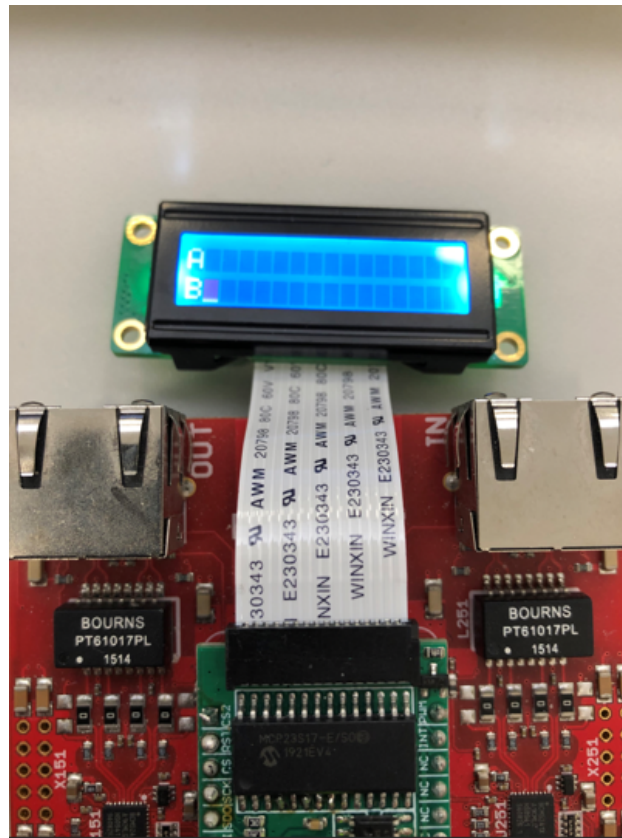


Figure 4.10: LCD click validation

4.1.2 OLED Click

OLED display was selected because of all his characteristics which distinguish him from a LCD display. OLED gives us the freedom of controlling each and every pixel available in the display and beyond that colours can be added. It has great vision angle and because we control every pixel, the capability of drawing figures, images or characters can be possible which can be more intuitive for the use to read and can save screen space.

4.1.2.1 Overview

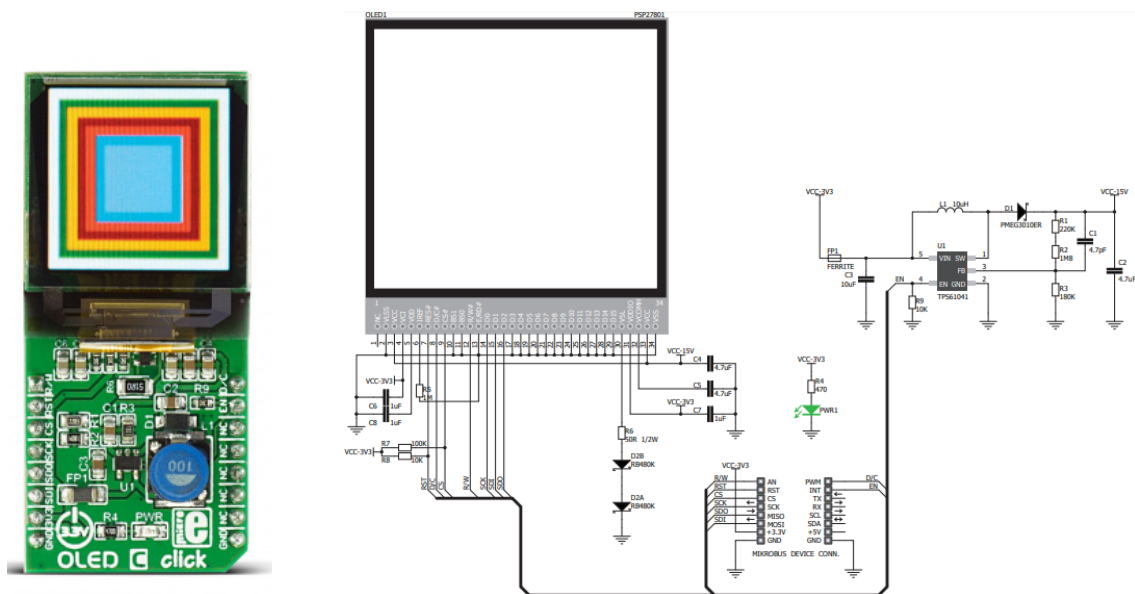


Figure 4.11: OLED click module & OLED click schematic [21]

This click has a 96x96 high colour OLED display, PSP27801. It can display 65k different colours. The graphics driver used in this display is the SSD1351, it can support two colours modes and up to 128x128 pixels. Highlight the fact this OLED, as said, only has visible 96x96 pixels. This controller has a SPI interface. We can see from the schematic that pin D0 acts as the SCLK, D1 as the SDI and D2 as SDO. The D/C pin determines if the device is receiving commands, logic low, or data, logic high. Upon receiving a byte is a type of data it will be written directly to the Graphical Display Data memory, but if it's a command it will be decoded and written into the corresponding command register. To perform the correct transmission of data certain timings must be fulfilled, figure 4.12. [23]

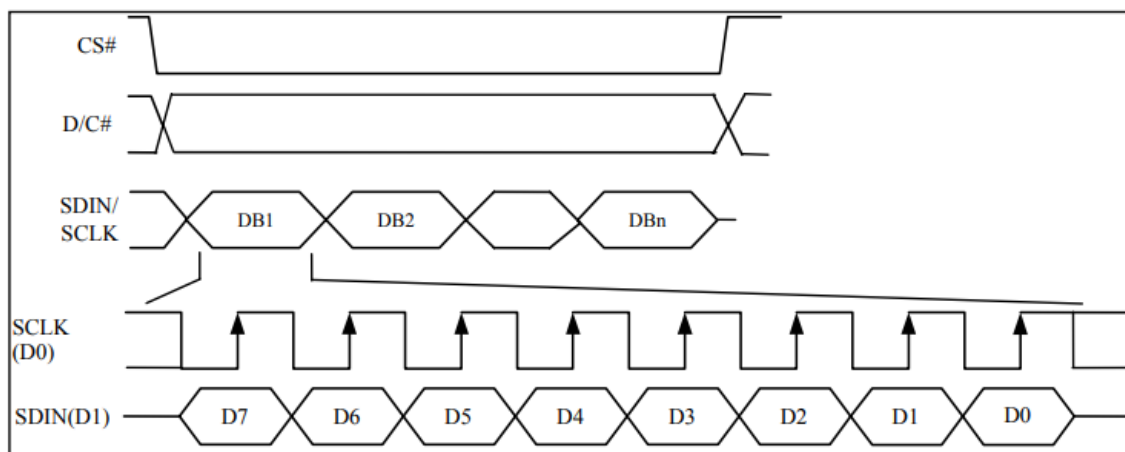


Figure 4.12: OLED click write timings[29]

Because this OLED works with 65K colour mode, only two 8bit words are required per pixel. The 5 MSB of the first word, is interpreted as the red colour data, the remaining 3 LSB are interpreted as the GREEN colour data. The second 8bit word contains 3 more bits for the GREEN colour, and the remaining 5 LSB are the BLUE colour data. This comprises the 5:6:5 pixel format.

Read Data			Data bus																	
Bus width	Color Depth	Input order	D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
8 bits	65k	1st	X	X	X	X	X	X	X	X	X	X	C ₄	C ₃	C ₂	C ₁	C ₀	B ₅	B ₄	B ₃
		2nd	X	X	X	X	X	X	X	X	X	X	B ₂	B ₁	B ₀	A ₄	A ₃	A ₂	A ₁	A ₀

Figure 4.13: RGB[29]

The R/W pin, when using the SPI interface should be pulled to a LOW logic level. The EN pin is connected to the small, low power onboard step-up converter, used to provide voltage for the display. It uses the 3.3V of the mikroBUS and converts it to 15V on the output. To enable this converter, the EN pin should be pulled to a HIGH logic level.[29]

4.1.2.2 Initialization

Just like the LCD display, here, after understanding how to communicate an initialization process must be followed so we can use the display. After being powered we have to wait at least 1ms to pull the RST pin to a logic low for at least 2μs and after that bring it to high again. After this send the command AFh which the driver will understand as turning ON the display.

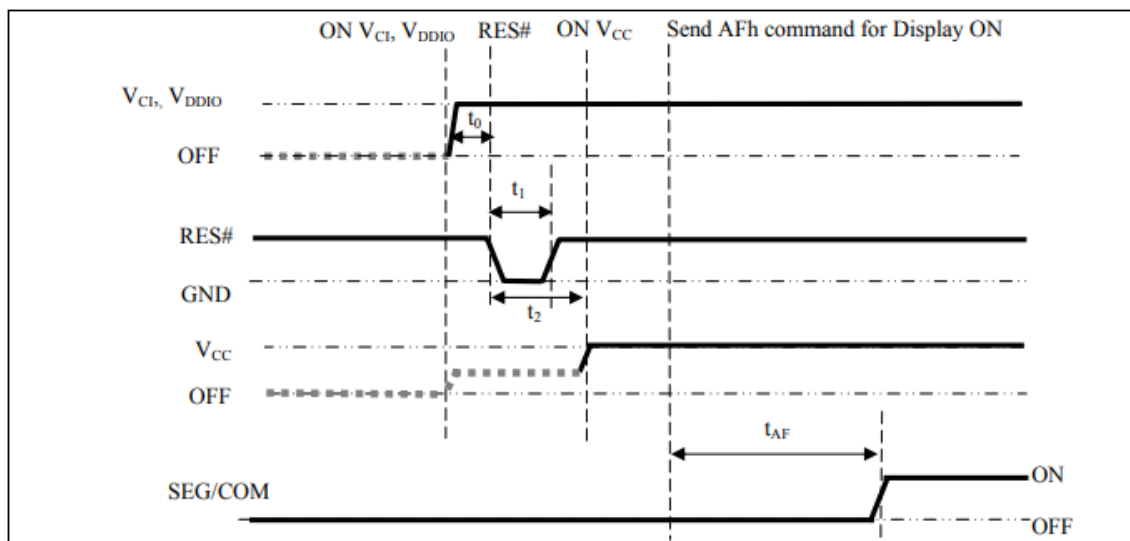


Figure 4.14: OLED initialization[27]

After initialization is complete we can now use the display. Note that the SSD1351 controller supports 128x128 pixels but this display only has 96x96 pixels visible. After realising the 96x96 pixels used in the 128x128 are from pixel 16 to 112 in columns and 1 to 96 in lines (this means is centered with the 128x128 but to the top) we can write a character in the display with any colour we desire, figure 4.15



Figure 4.15: OLED click validation

4.2 Bluetooth

Bluetooth gives us different types of new solutions which can be valuable. One of each, the case of a truck with a trailer. The user interface doesn't need to be outside and directly connected to the CAN bus. This allows the use of a different kind of display in this scenario. It also gives the

version of the CodeLess software via the JTAG interface, which is required by the DA14531 module itself. This board can only operate with 3.3V logic voltage level.[22]

DA14531 supports a list of several AT commands. These commands can control BLE connectivity, connect or disconnect to other device, scan for nearby devices, report it's address, and several others. But not all of this commands are available. There are three types of configurations in DA14531, each one has a list of available AT commands.

- **Datapump:** This configuration is optimized for making binary data transfers, supports 30 commands, including all binary data commands;
- **Standalone one:** The standalone configuration is optimized for standalone devices which may not be connected to an external mode. It doesn't have binary commands, they have been replaced for a more suited standalone configuration. It supports 30 commands;
- **Standalone two:** Similar to the standalone one configuration having the extra of having 42 AT commands. instead of 30

After seeing the list of all AT commands available, and for the purpose of why Bluetooth was introduced to this project, commands to send data are the most important ones, for this, the commands that stand out are, AT+MEM and Binary mode, where for this last one, a sequence of AT commands must be performed in order to entry this mode.

4.2.2 Connection and Communication

In first place, we ought to test the AT commands, for this purpose, to better show case this tests, TeraTerm and a USB-to-UART bridge controller, PL2303, are used.

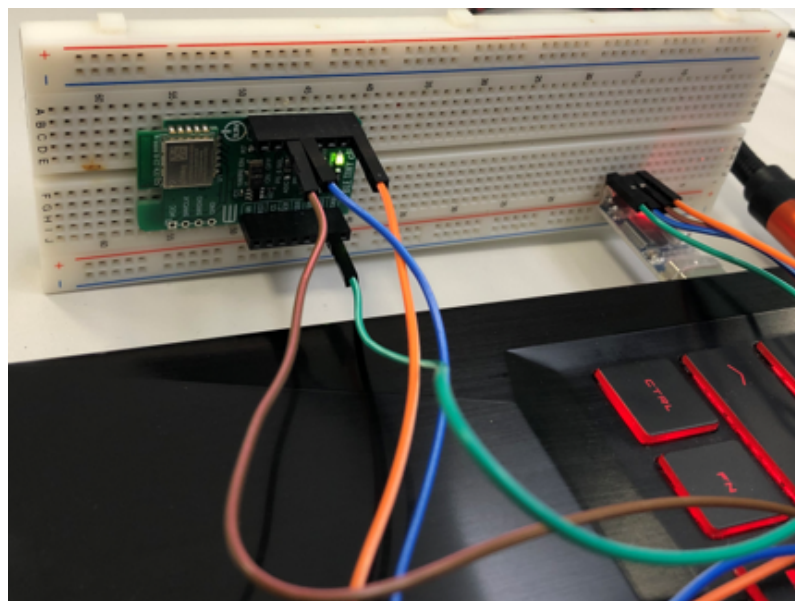


Figure 4.18: BLE click connected to PC

In the connections shown in figure 4.18, we need to have into consideration Rx and Tx of the ble click module. This pins connect to the Rx and Tx of the controller, respectively. Also to correctly operate, 1WIRE enable switch in the click board must be off. After selecting the correct COM port and setting the correct baud-rate to send AT commands, 57600, we can simply type "AT", simplest AT command, in the Tera Term panel and expect a "OK" response, then, typing "AT+BDADDR" will give us the Bluetooth device address has a response to this command:

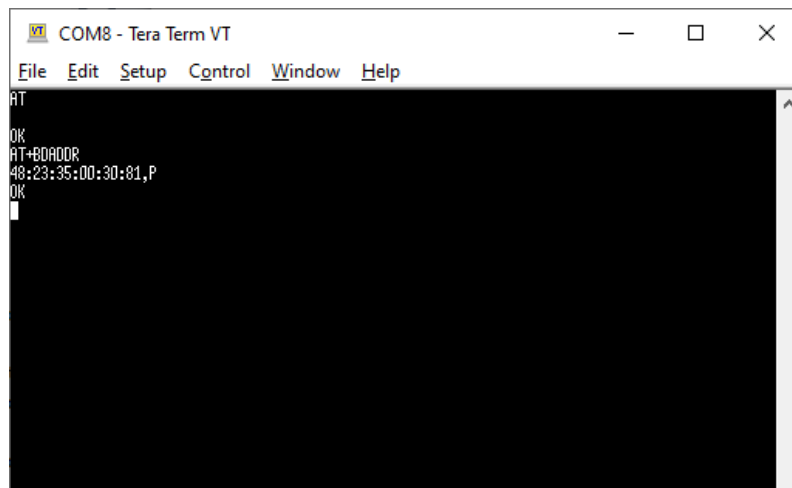


Figure 4.19: Basic AT commands

The next natural step is to be able to establish a connection to another device via Bluetooth and to be able to send any data that we want. BLE devices have two roles in a connection:

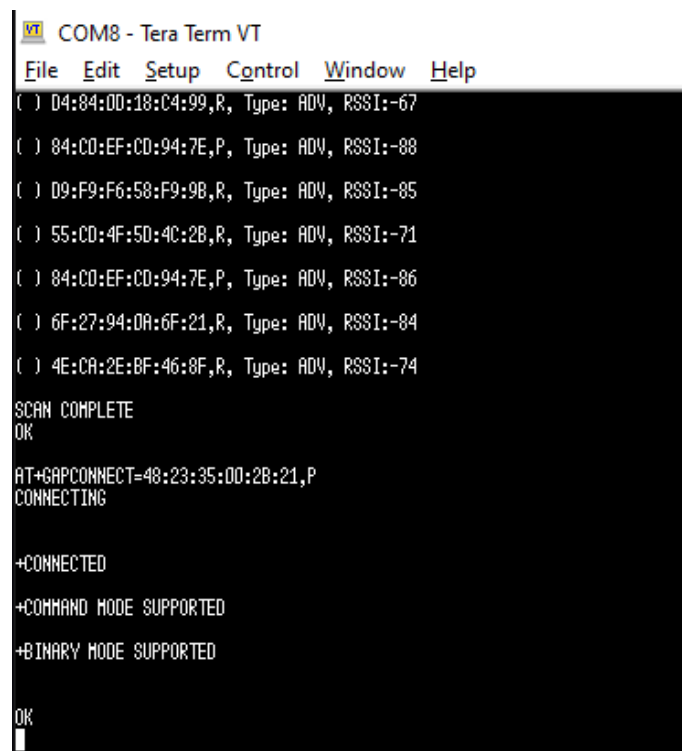
- **Central role:** This role is the one who scans for another devices and the one who establishes the connection to the desired device;
- **Peripheral role:** In this role, the device advertises his signal, it becomes visible and ready for connection. By default the DA14531, when powered, is in Peripheral mode.



Figure 4.20: Powering two BLE clicks

figure 4.20 shows two BLE clicks, one connected to the PC and the other connected to the XMC4300 microcontroller, used in order to validate this technology and test establishing a connection and sending data via Bluetooth.

After knowing the default configuration the BLE click modules comes with, Standalone two, the only way to transfer data is by using the command `AT+MEM`. A common memory is shared between both devices, with 4 blocks each of 100 characters. To transfer data, first, one of the devices must set the information in one of the blocks and afterwards the other device can access it, by selecting that same position. This way of data transmission isn't fluid. The data isn't being passed bidirectionally and it isn't practical. Because we were not satisfied with this solution, in order to be able to use the binary mode, we have to program the SPI flash of the device, via JTAG or UART, to be able to use the Datapump version. After flashing both clicks with the new configuration we can now use binary mode. Following the steps to establish a connection between the two devices, after connecting immediately it's recognized the capability of entering into the binary mode.



```

COM8 - Tera Term VT
File Edit Setup Control Window Help
( ) D4:84:00:18:C4:99,R, Type: ADV, RSSI:-67
( ) 84:CD:EF:CD:94:7E,P, Type: ADV, RSSI:-88
( ) D9:F9:F6:58:F9:9B,R, Type: ADV, RSSI:-85
( ) 55:CD:4F:5D:4C:2B,R, Type: ADV, RSSI:-71
( ) 84:CD:EF:CD:94:7E,P, Type: ADV, RSSI:-86
( ) 6F:27:94:DA:6F:21,R, Type: ADV, RSSI:-84
( ) 4E:CA:2E:BF:46:8F,R, Type: ADV, RSSI:-74
SCAN COMPLETE
OK
AT+GAPCONNECT=48:23:35:00:2B:21,P
CONNECTING

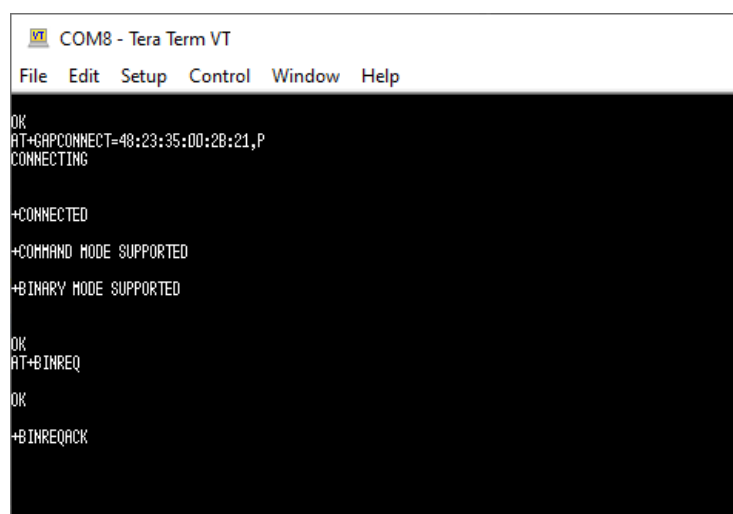
+CONNECTED
+COMMAND MODE SUPPORTED
+BINARY MODE SUPPORTED

OK

```

Figure 4.21: Connection established with Datapump configuration

To enter binary mode either device must send the comand "AT+BINREQ", upon receiving this, the other one must respond using "AT+BINREQACK", after this both devices have entered the binary mode and can now send any data bidirectionally, being that whatever we send is received instantaneously in the other device. This mode suits much better the needs for this project, we can now be sure that any data sent by whatever device is received in the other.



```

COM8 - Tera Term VT
File Edit Setup Control Window Help
OK
AT+GAPCONNECT=48:23:35:00:2B:21,P
CONNECTING

+CONNECTED
+COMMAND MODE SUPPORTED
+BINARY MODE SUPPORTED

OK
AT+BINREQ
OK
+BINARYREQACK

```

Figure 4.22: Binary mode

4.3 User Interface Design

Before designing the user interface mock-ups, we have to know what kind of screen and display we are using in our solutions. After testing click displays modules, and having into account the qualities that the display has to have, we decided to select two kinds of displays. The displays selected have the capabilities to answer the problems pointed in the last chapter, as well as the two possible solutions described.

- **LCD 20x4 module:**

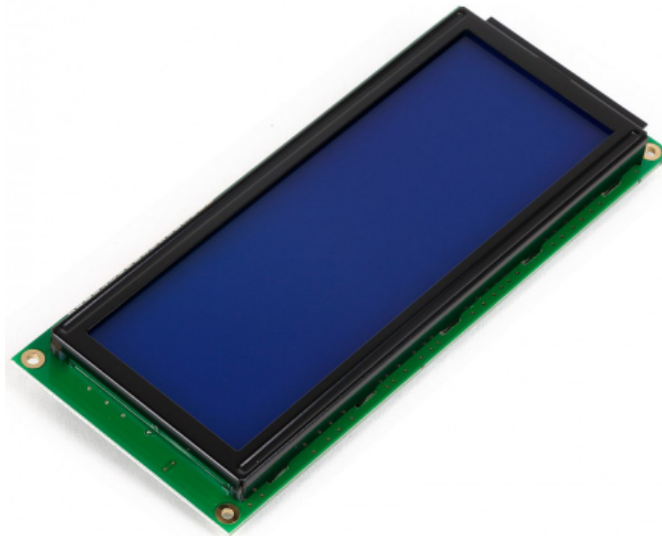


Figure 4.23: LCD 20x4

This display was selected because of the first tests performed with the LCD click module. A bigger version was selected to ensure that the information displayed isn't too small. Because tests were already performed in the same kind of display, being able to write to this LCD is already ensured without losing time understanding how the module works. The only difference with the LCD click is, is because of the lack of pins available in a click board the port expander, MCP23S17, is used. As we have seen, the port expander has a SPI interface to convert the data to a parallel communication with the Display connector. Because this type of display is visible with direct sunlight, and is robust against climate changes, if the user interface needs to be outside this LCD module provides a good solution.

- **TFT LCD 4.3 inch, touchscreen display:**



Figure 4.24: TFT LCD display

A TFT display was selected because it is a type of LCD with a thin film transistor (TFT) attached to each pixel. Which means better visibility in situations of high ambient lighting. Although OLEDs give better viewing angles than a TFT LCD display, they have poor visibility when light is pointing directly to the display. TFT LCD displays are also cheaper and have lower power consumption compared to OLED. Similar to OLED, in TFT LCD we can control each pixel, making it easier to create symbols or figures, whereas in LCD we can only use characters already established. The TFT module selected also uses the RGB interface already tested and validated in the OLED click module. This display also has the feature of being a touchscreen display. Given its qualities this module is best suited to be in an indoor scenario. Although it has the feature of touchscreen, buttons can still be used, giving this module versatility.

4.3.1 LCD Display

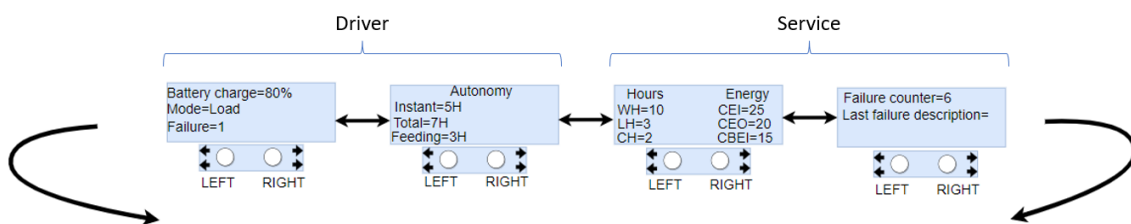


Figure 4.25: LCD mock-ups

After talking with the truck drivers and discussing with Addvolt the way the user interface is going to work we ended up with the mock ups in figure 4.20. The first page is the most important one. It gives the driver all the information he needs to know.

- Battery state of charge: the driver now knows the actual percentage;
- Mode: it can be Grid or Load connected;

- Failure: in case of a failure the interface gives fills the space in front with which failure occurred.

The second page allows the user to know the autonomy of the system in hours, it gives three different metrics.

- Instant: autonomy in that moment;
- Total: autonomy in considering all the time of use;
- Feeding: autonomy when is being charged;

The third page is directed to service and tracking of the Addvolt system. The last page gives the user a understanding of how many failures the system had, as well as the description of the latest failure.

To go through all the pages the user uses two buttons, one allows to go the left page and the other to the right page. If the user is in his last page it can still go right, making its way to the first page. The reverse also happens. In this interface all the text that is behind the two points is fixed information. Only what shows after is going to be updated and modified during use.

4.4 User Interface Implementation

After the designing and acceptance of the user interface mock-ups we now have to implement it. Due to the fact the TFT LCD display can only be connected through a FPC connector the implementation of his user interface was not developed. The only display used in this implementation was the LCD 20x4 module.

XMC4300 relax kit has integrated two push buttons. Button1 and Button2 will emulate the buttons described as right and left, respectively, in the user interface mock-ups. Taking advantage of DAVE APPS we can detect if this pins are pressed or not. The DIGITAL_IO APP, when instantiated configures a port pin in the XMC, in this case the pins connected to Button1 and Button2. Using DIGITAL_IO_GetInput(const DIGITAL_IO_t *const handler), we can receive the current status of the button. If pressed is connected to ground , 0, otherwise is pulled to VCC, 1.

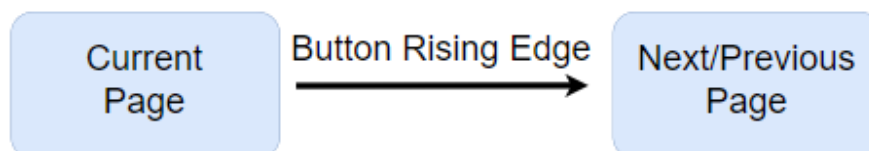


Figure 4.26: How to change page

Figure 4.30 illustrates the action to change page in the user interface. A page will only change when the user releases the button he pressed. As seen in the previous section, each page has information that will remain always the same, independently of the time spent in that page. This information is the one that is printed as soon as the user changes page. To perform the page change in the rising edge of the button press, a variable is needed for each buttons. This variable will be 1 when the button is pressed. It will only increment or decrement the view variable, figure 4.31, if this variable is 1 and at the same time the current status of the pin is 1 also. Which means the button has also been released. In each view the page will be printed. This switch case only happens inside the if condition described now.

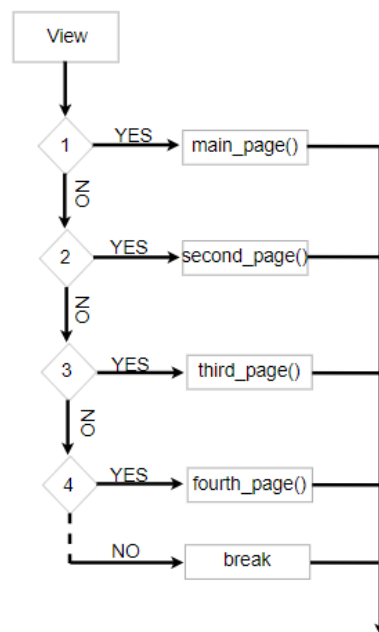


Figure 4.27: Page views

4.5 Solution

Following the previous validations, making sure that each individual part relevant to the project can be manageable, we can now add complexity and integrate different technologies together. By uniting all technologies we start reaching the desired solution. In the following sub sections we will start to join technologies, trying to achieve is purpose together. Figure 4.24 shows the solution with everything connected.

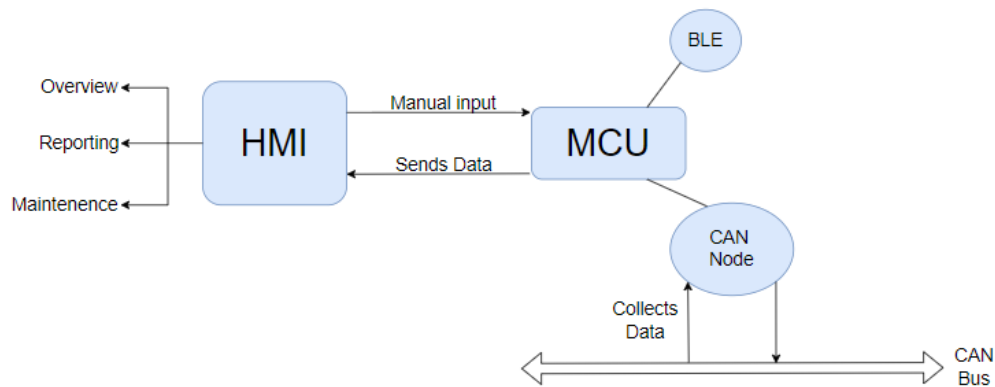


Figure 4.28: Interface system

In the last sub-section we will dive into a PCB designed and produced to incorporate all technologies, the first prototype, but because there was a time limit to this project, it's incorporation was delayed, and so that we could achieve all the functionalities desired and simulate what a real solution would look like, we had to connect everything with the use of a breadboard. Because the board we have been using could only have one click we had to choose which click to use, Bluetooth or the display. This being the case and because both click display modules were small, for the solution we are looking for, and were used just to make sure we could freely display the information we want, a LCD 20x4 was selected as an upgrade of the click one, and using a breadboard and the pins available in the XMC4300 board we could now try and use all technologies together. figure 4.27 shows the display connected with the micro controller and this has connected in his click socket the BLE module. In the left we use, just as in the Bluetooth validation, another BLE click module in order to simulate the transmission and the reception of data from the PC to the solution.

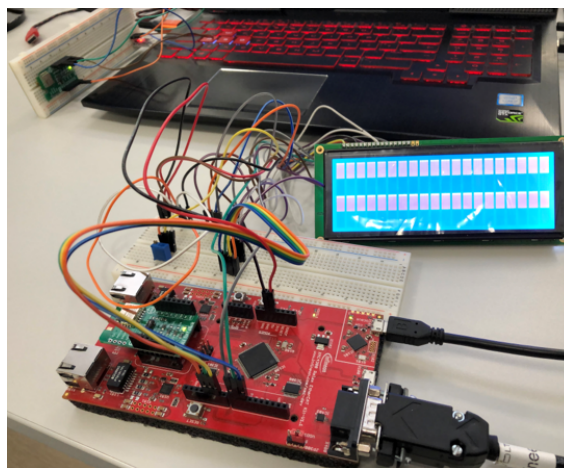


Figure 4.29: Prototype developed

4.5.1 Display & CAN

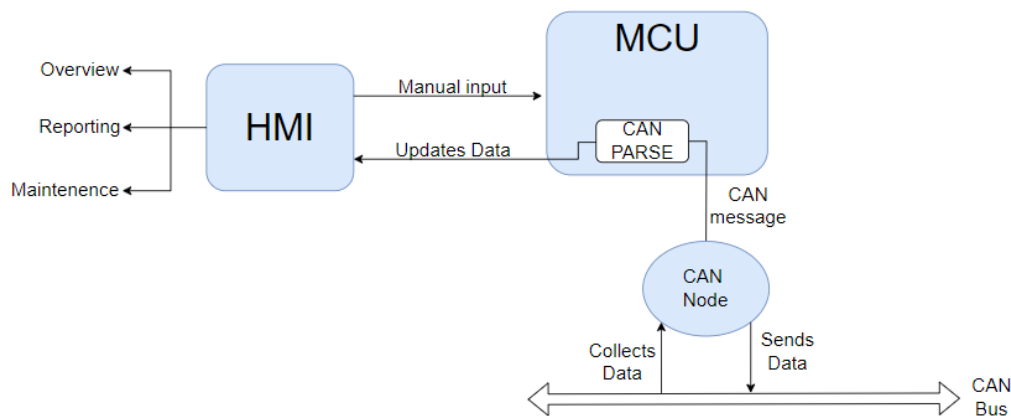


Figure 4.30: Display with CAN

Figure 4.30 shows how a CAN message is treated all the way to being shown in the display. Several messages, at a different cadence, are being sent continuously from the Addvolt system to the CAN bus. The display must be able to show different variables, updated, in real time. When a CAN message is received it comes in a raw state, and it needs to be treated so that it can be used. Our new solution is going to be reading all the messages in the bus, but is only interpreting the messages that interest us, CAN parse. This messages are the ones that include variables from the pages of our interface.

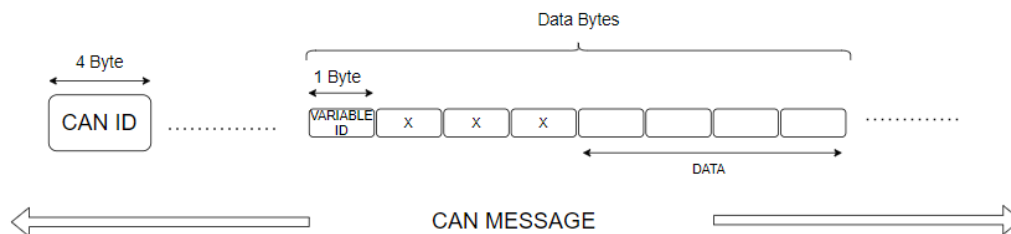


Figure 4.31: CAN msg AddVolt protocol

We also need to have in consideration that a variable received may not be displayed in the current page that the display is in. Nevertheless that variable is updated and if requested, when the user changes page, the displayed information is showing the latest data concerning all variables. The way to do this is having a structure with all variables used just like we see in the figure below. Every time a CAN message is received a interrupt is activated. If the ID is a match the data of the variable is updated in the structure so that all information is updated.

```
typedef struct display_variables {
    float btry_SOC;
    float btry_voltage;
    float btry_current;
    float fail_first;
    float btry_auto_inst;
    float btry_auto_total;
    float btry_auto_feedng;
    float failure_counter;
    float loading_hours;
    float charging_hours;
    float working_hours;
    float CEI;
    float CEO;
    float CBEI;
    float load_connected;
    float grid_available;
} display_var;
```

Figure 4.32: Variables structure

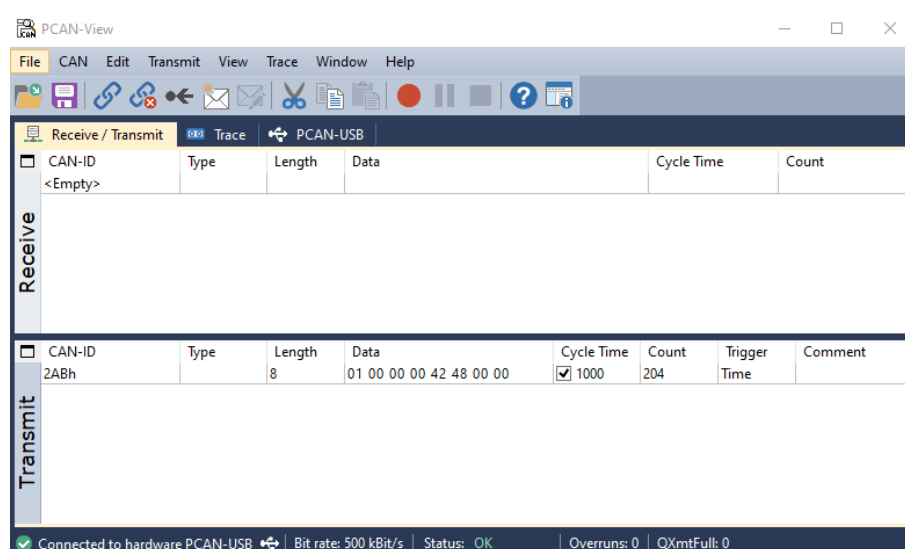


Figure 4.33: message represented in PCAN view

Assuming this CAN ID matches with one ID we are expecting and the first byte of the data array indicates the variable to be updated is the battery percentage, we now proceed to read the last four bytes, 0x42480000 in hexadecimal is equivalent to 50 in decimal, so we update the correspondent variable with it's new data, and print it.

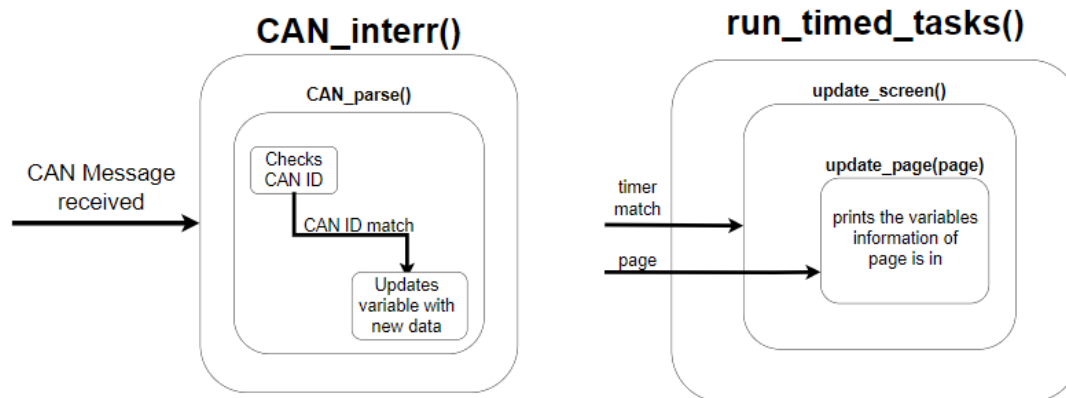


Figure 4.34: Architecture

After the variable is updated it doesn't get shown on the screen right away, instead we implement a timer. This timer is a dave APP, it uses the internal clock and it is associated to a interrupt that will, from time to time, update the page by printing all the variables considering the page the user is currently in.

4.5.2 Display & CAN & BLE

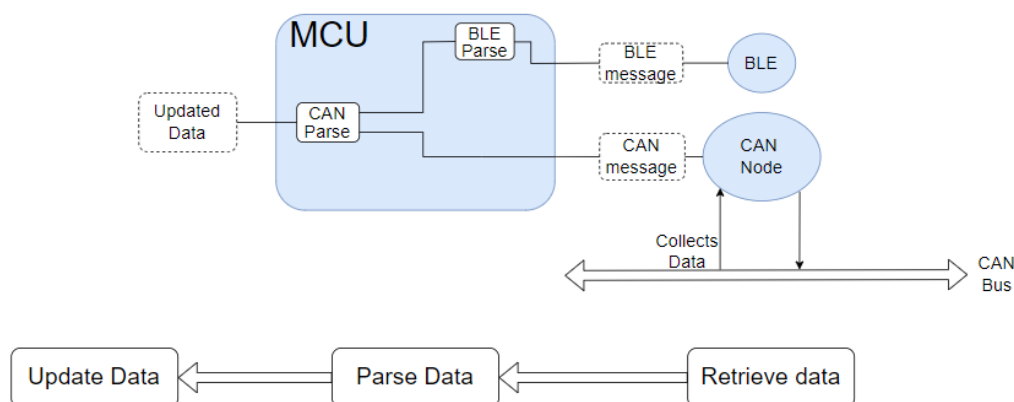


Figure 4.35: CAN and BLE

The integration of Bluetooth adds complexity, there are new messages to be treated, but this new data is fully customizable, giving the possibility to simplify this process. So, a proprietary Bluetooth protocol was created. The idea is to encapsulate a CAN message in a BLE message, created by us, in order to create a bridge with CAN. What we have is a gateway between Bluetooth and CAN, that differentiating from the treatment a CAN message has in the CAN node, what

happens here is the unwrapping of our Bluetooth messages, BLE parse, so they can, after this, suffer the same treatment already implemented before, CAN parse, of a CAN message.

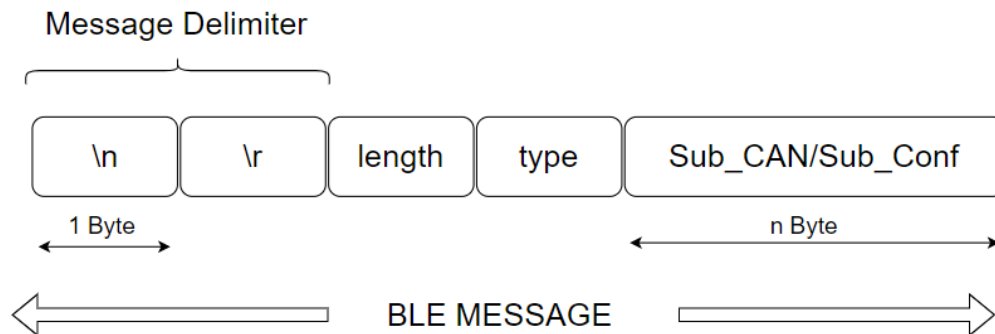


Figure 4.36: BLE message format

- **Message Delimiter:** When connected to a Bluetooth device, and in binary mode, data can be sent at any time so to determine when it's the start of a BLE message, two bytes are sent, before all the important information, that mark the beginning of that message;
- **Length:** Indicates the total length of the message from this byte forward, 1 byte;
- **Type:** Indicates what kind of message is being sent, Sub_CAN or Sub_Conf are the only specified, where the first is if the message sent contains information from a variable and the second can be for configuration or debug, 1 byte;
- **Sub_CAN/Sub_Conf:** The last bytes of a BLE message are the part that contains all the data, having a variable size depending on the type of BLE message. Due to priority, Sub_CAN is the only one that it will be explained and was implemented.

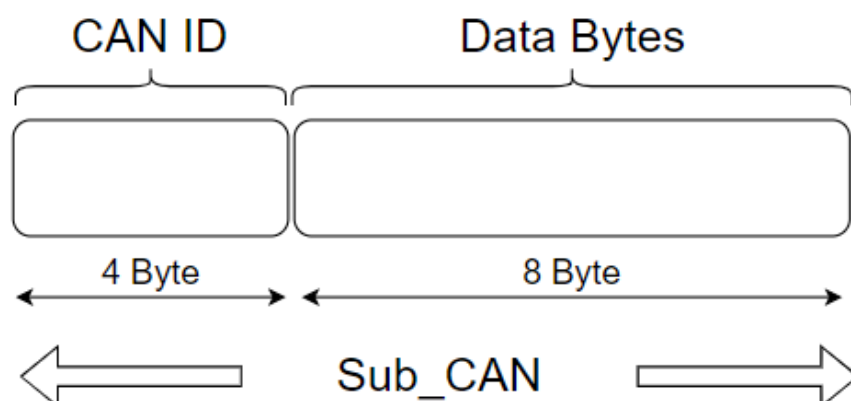


Figure 4.37: sub_can format

- **CAN ID:** 4 bytes are necessary for the case that the ID is 29 bits, extended, or 11 bits, standard, but take into consideration that for example, 0x4f0 is different than 0x000004f0, so the first bit if the ID determines if it is extended or a standard message;
- **Data bytes:** The same way as a CAN message, the 8 Data bytes contain which variable is going to be updated and the new information.

Now, two ways of communication are possible, from the MCU connected to the display, having integrated the click BLE module and being also connected to PCAN simulating the CAN BUS, to the click BLE module connected to the pc and using TeraTerm as interface, or vice versa. It allows all scenarios to be tested, we can retrieve a CAN message from the CAN bus and send it via Bluetooth to another BLE device, we can send a BLE message containing new data from a variable, receive it and display it in our display interface, or even receive this BLE message and send it to the CAN bus.

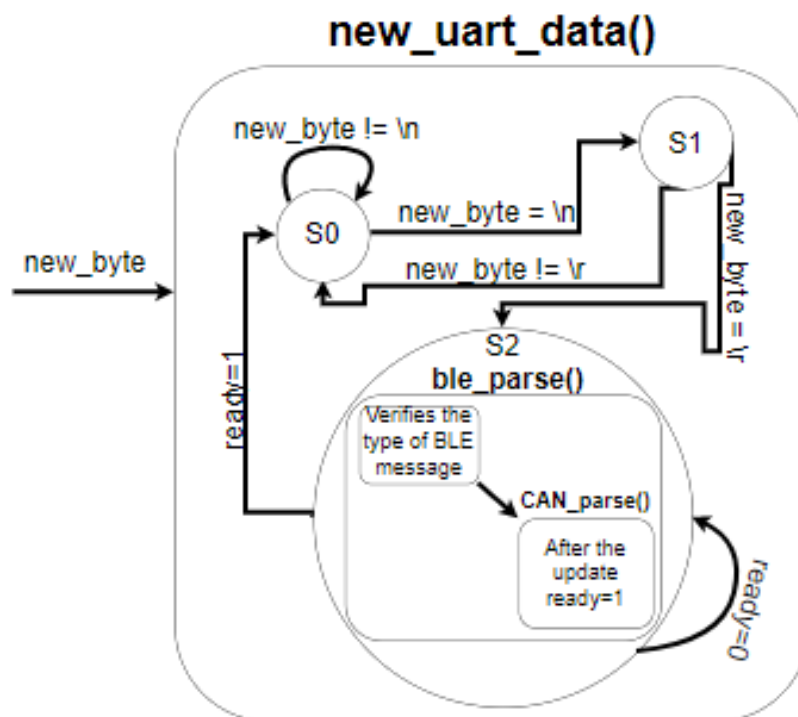


Figure 4.38: Bluetooth messages handling

Every Bluetooth data is received via UART, and is received byte by byte, so in order to check if the data is a BLE message like we established, it goes through a state machine. Because the third byte is the length of the message, the only way to get out of state 2 is if the flag ready gets activated, this is when the new_byte counter matches the length of the message.

4.5.3 PCB

After all the technologies were tested and the full purpose of each one was accomplished, the next step was to design a board that is able to incorporate all technologies and can carry out the idea of this project.

Because the schematic is too big in order to show it in one figure, it will be divided into different parts and in each one the explanation of why these components were selected to go on the board.

4.5.3.1 Protections

This circuit represents the protections of the power supply. It was given by Addvult and has the purpose of protecting the system against, high voltages, high loads, polarity inversion and short circuits.

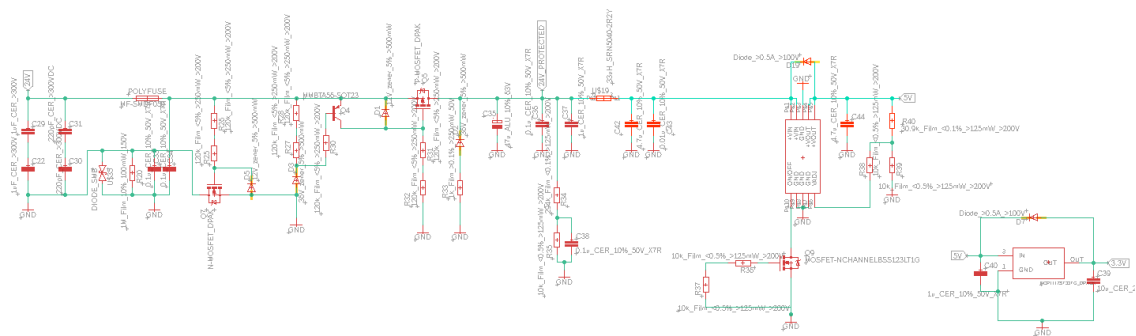


Figure 4.39: System Protections

4.5.3.2 Microcontroller

Highlight the use of a mikroE bus socket if we want to use click module. Also the integration of a buzzer. When a warning or an alert is being displayed in the user interface the buzzer can draw attention to the interface. It also has the capability of being powered by USB. All the guidelines to design a PCB with a microcontroller from XMC4000 family are followed.

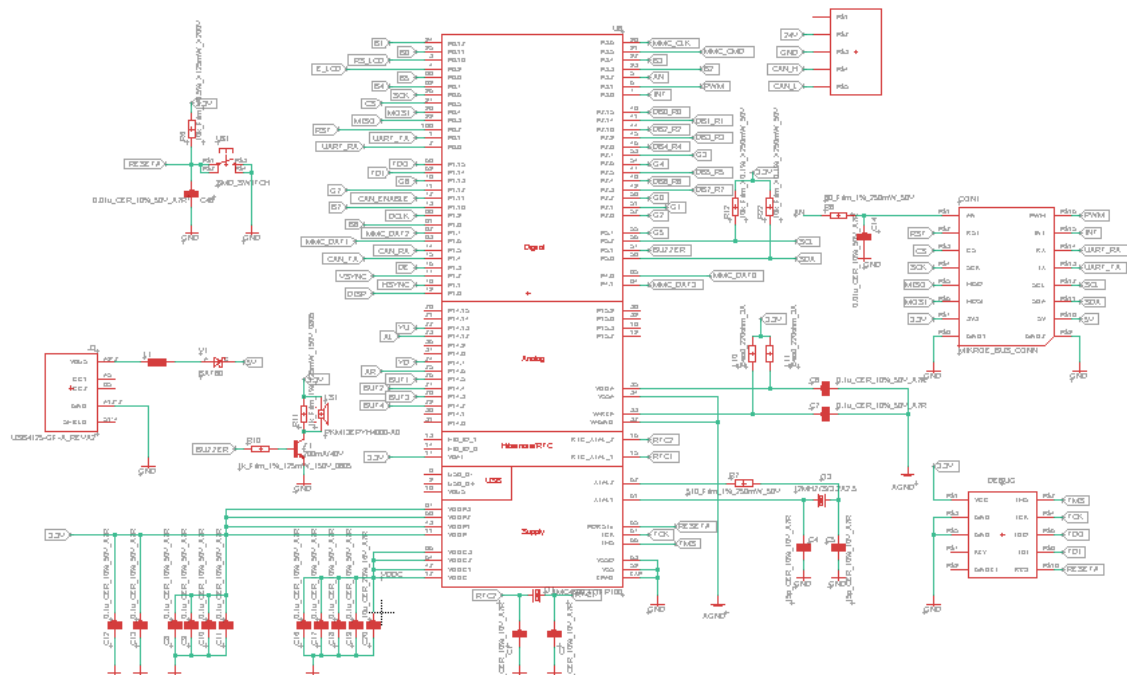


Figure 4.40: Microcontroller connections

4.5.3.3 Displays

This board will have the capability of having two different types of displays. A LCD 20x4 and a 4.3 inch TFT display. Both will be connected to the microcontroller using a FPC connector. We also have 4 buttons embedded in the board and the possibility of connecting another four buttons. This gives the opportunity to test different types of buttons.

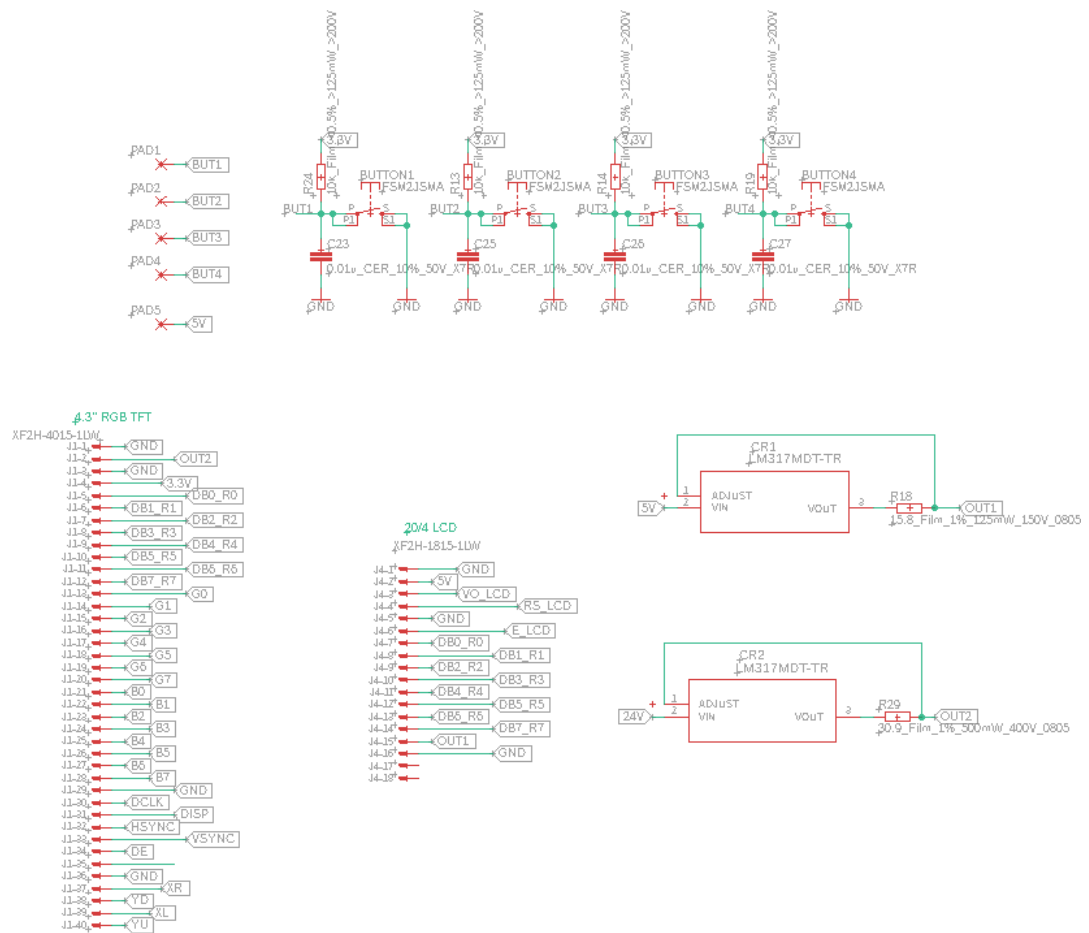


Figure 4.41: Display connections

4.5.3.4 CAN

This part of the circuit is everything related to CAN. The CAN transceiver, CAN bus termination and a 5 pin connector. The transceiver in the schematic wasn't selected because its EoL (End of Life). Just its footprint is important.



Two Bluetooth devices were selected, the first one is the the same as the one in the BLE click module, but because it wasn't available on the market another one was choosed, RN4871 that was already in stock in Addvolt. Despite being selected we can still take advantage of the click socket and incorporate the BLE click module.



After the selection of all components and the connection between them, the next stage is designing the board. This board must follow guidelines in order to be valid solution, such as dimensions

and cost.

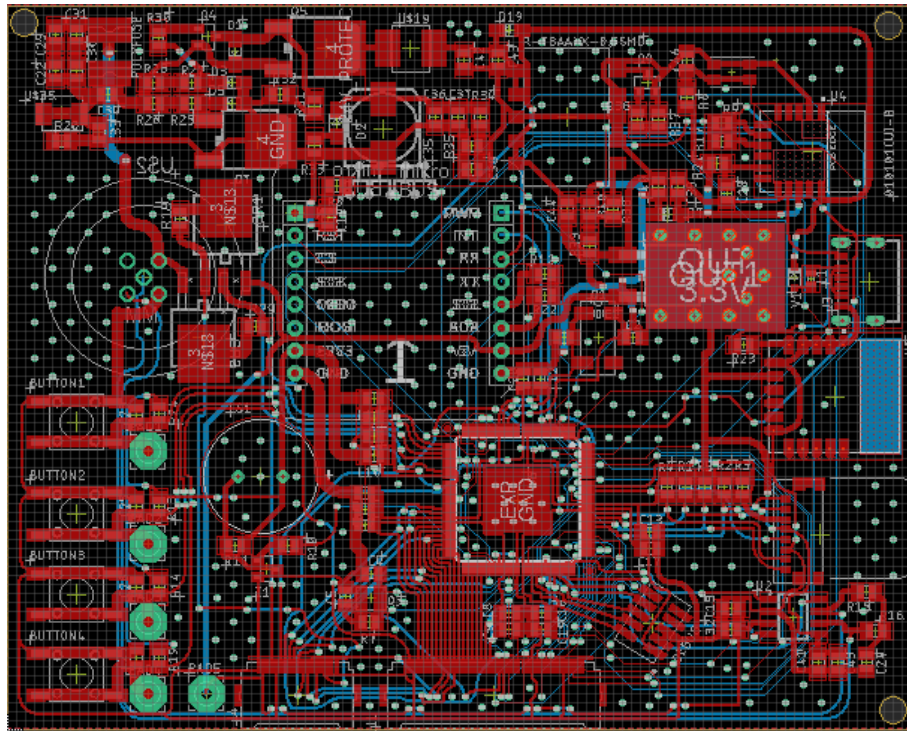


Figure 4.44: Board connections

Figure 4.40 shows the disposition of every component in the board. In order to have the best possible connections, without having to change layers, separating the schematic in smaller parts and doing those connections in the board allowed me to obtain the solution in figure 4.40. I also had to have attention to the fact that both displays had to be in the edge of the board so that display could be easier to maneuver. In addition, the buttons had to be placed in a way that the user can use them, both the ones already in the board and the ones that can be still implemented. In the figure below we can also observe a high number of vias, they are used to dissipate the heat generated by the components and also to make sure that both layers are at the same potential, in this case is ground.

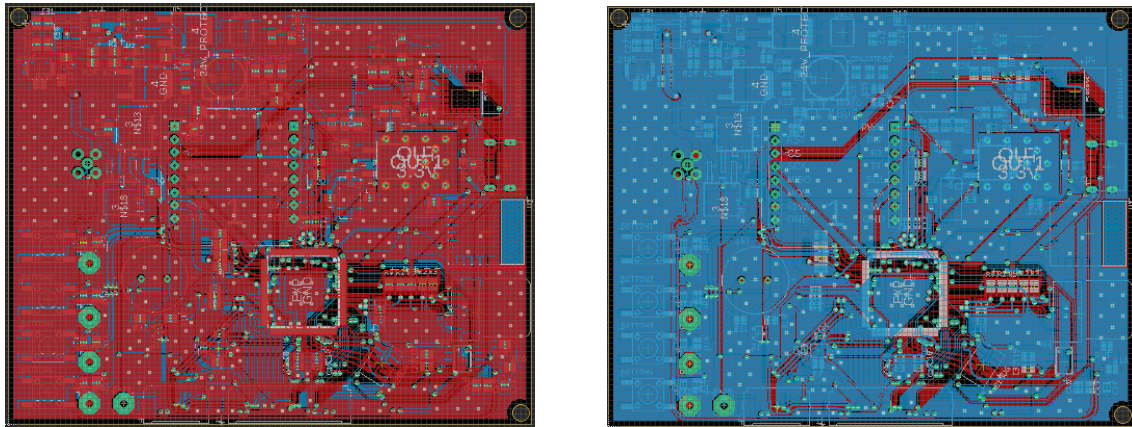


Figure 4.45: Top layer and Bottom layer

To minimize the cost of the board, the use fewer layers implies a lower cost, figure 4.41 shows both layers, top and bottom, and finally figure 4.42 shows how the board when printed will look.

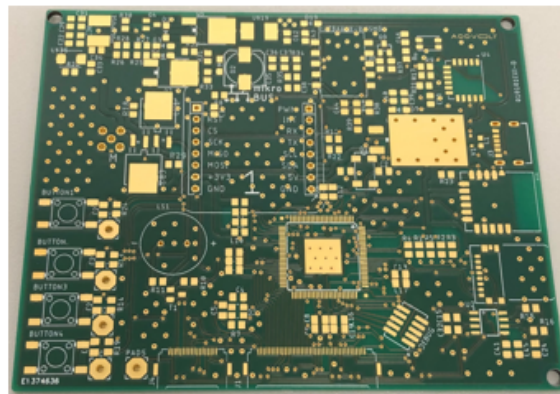


Figure 4.46: PCB printed

Chapter 5

Work results

After the implementation, the solution created has to be tested in all areas in order to be considered a solution. In this chapter, the solution developed, will be validated by the requirements listed before in order to know if this new solution is reliable.

5.1 Requirements validation

- **User interface is visible in different conditions:** This tests were carried out by putting the user interface in different lighting environments. At all times the information could be seen. LCD viewing angle is the most worrying, being difficult to understand what is being shown if not standing in front of the display.
- **User can go through all pages:** When user pressed one button of the XMC4300 relax kit, the page of the interface changed without delay and in the RE of the button. If the button is being pressed it doesn't changes.
- **Receiving a CAN message and display it:**

To perform this tests PCAN-VIEW and PCAN-USB was used primarily. Tests with actual CAN messages from the Addvolt protocol were carried out. After being able to receive and display them in different pages at different rates, the same tests were carried out but now being connected to a actual Addvolt system, figure 5.1. Because with PCAN we can emulates the sending of CAN messages, it was expected with the actual system being connected to the user interface we could display all the information it was set out to. And as expected no problem occurred.



Figure 5.1: Prototype connected to Addvolt system

- **BLE and CAN gateway:**

Using two BLE devices, one connected to the microcontroller, and the other one with the PC we first tried to receive a message from the CAN line, transform it to a Bluetooth message format and retrieve it in the computer. Then the inverse was carried out, send a Bluetooth message with CAN information in it, from the PC to the microcontroller, and afterwards being able to send it to the CAN line. To perform all this a python script was elaborated, in order to emulate what TeraTerm was doing. This script allows us to know and personalize what messages we are sending and receiving, at what rate and the if there are any losses.

After completing the transmission of a single message in both ways, we tried sending this message at different rates. What was visualized was the lost of messages and also incorrect messages being received. So we had to know where the bottle neck of this data transmission is. Once again, with the help of PCAN and sending, from the PC a BLE message to the MCU, a message every 0.01 seconds, in 10 seconds it is expected to receive 1000 messages, and sent to the CAN line. Only around 700 messages were sent to the CAN bus. If we increased the rate of the messages even more, fewer messages were sent to the CAN line and wrong messages were being sent to the bus line. This means at a high rate, bytes from different messages were being sent together, what resulted in receiving CAN messages different from what was expected. In the opposite direction the BLE messages were also being lost.

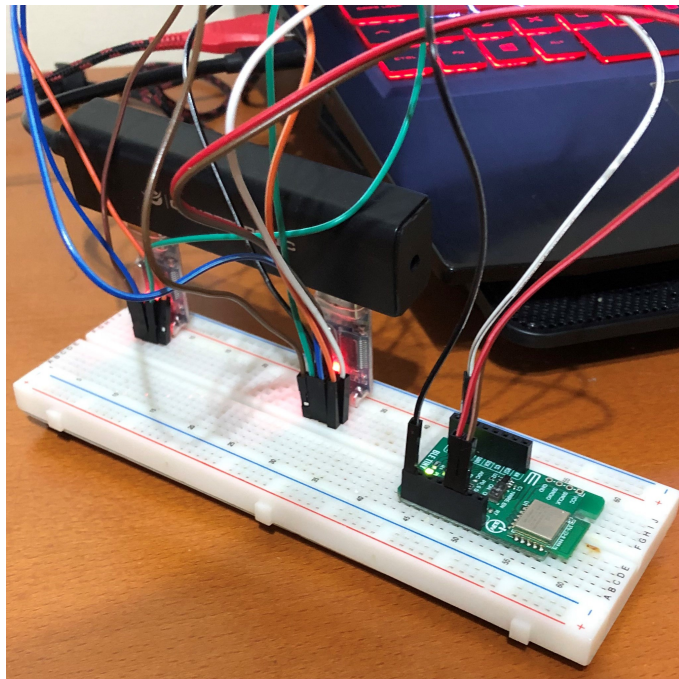


Figure 5.2: First test of BLE messages

After understanding data was being lost, and being received or transmitted incorrectly, we needed to discover where the errors were occurring. So we divided the transmission of data into smaller ends. Figure 5.4 shows the first test. This set up allows us to know if all the messages being sent by the PC are received correctly in the BLE click. We are testing if the TTL (USB to UART converter) isn't imposing a baud rate smaller which is causing the lost of messages. After testing different kinds of rates and amounts of messages being sent we concluded that TTL wasn't causing this errors to occur.

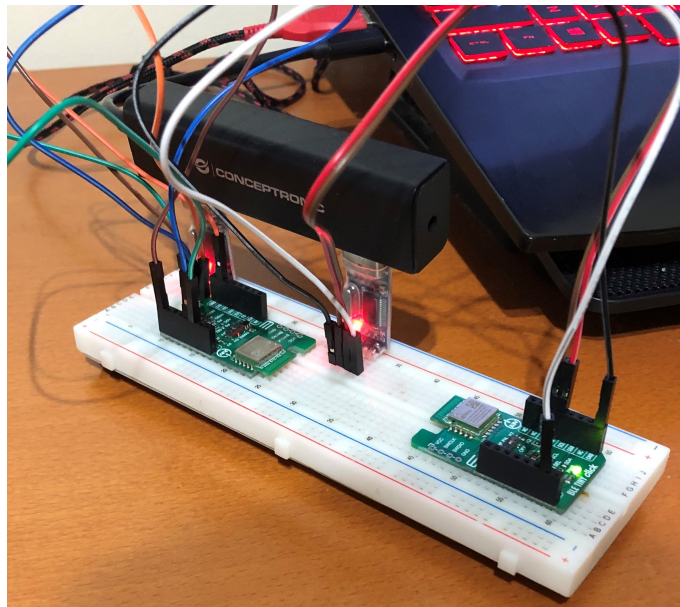


Figure 5.3: Second test of BLE messages

After knowing for a fact the error isn't sending messages from the PC to the BLE click, the second set up, figure 5.5, we want to prove that all messages sent from one BLE click to another, via Bluetooth, are being received correctly. After performing tests identical to the ones before, and sending a message repeatedly over different rates we had discovered that as stated before messages were being lost and some were being received incorrectly. After the testes we realize sending one message 2000 times with a rate of 9 ms apart from each one, all were received correctly. From here, every millisecond below would increase the amount of lost messages as well as messages being incorrectly received.

The DA14531 chip has a connection interval of 15 ms and a MTU, which is the payload of a BLE message, of 65 bytes by default. What is happening is for each BLE message being sent, more than one Bluetooth message developed by us is filling the 65 bytes available in a BLE message. This errors and lost messages appear because the connection events are bigger than the connection intervals.

5.2 Comparison with previous solution

The solution developed is able to simulate every functionality the older solution had. It fulfills all the requirements it was set out to in terms of functional requirements, and it eliminates the problems identified in the interface Addvolt currently uses. Performance wise, the problem found was in the throughput of BLE where messages were being lost. One idea to surpass this limitation is given the fact that in our interface we don't update the display every time a new CAN message is received, we allocate the information in a structure and with a routine, that happens every x seconds, we update the display, we can do the same in the Bluetooth communication. We only

send messages every x seconds to make sure every message is being sent and received correctly. Another possibility is to take advantage of the size of the MTU, payload of a BLE message. Right now what is happening is a BLE packet is sending multiple messages from the protocol we created, until the 65 bytes are fulfilled, or not. What we can do is adapt the packet format of the protocol designed to have the same bytes as a BLE packet. Here we can add multiple messages.

Chapter 6

Conclusions and Future Work

In this last chapter we do an overview of all the work that was carried out, pointing out what was done and expected. Then, the work that still has to be done, as well as new improvements that can still be achieved in the overall work will be pointed out.

6.1 Conclusion

After the results and the observations made in the previous chapters, it can be concluded the new user interface developed is an improvement compared to the current Addvolt solution. The main problems identified not only by Addvolt but also from the drivers were solved. There is no limit of variables to be displayed. After designing and implementing the new user interface, we can freely choose and decide what variables to display. There is no need to choose between variables because there is no limit of information we can display. Bluetooth integration allows the user interface not to be necessarily connected to the CAN bus. This is a great solution for the case of trucks with trailers. Bluetooth also allows to abandon the use of the adaptor and the costs this impose. Although it isn't a finished work, this project has given the initial steps to a better and more complete solution from what was first thought of.

6.2 Future Work

After the conclusion of the project, there are always aspects that need improvement and others that can still be done. A project will never be a finished job as technology is continuously improving and new ideas take form, here is some work that still needs development:

- A natural step is to solder the PCB developed with all its components. Test it and afterwards connect it to an Addvolt system.
- The implementation of the TFT display in the solution needs to be developed. It has the advantages of not only give descriptive information but also has the capacity of developing icons that can substitute unnecessary text.

- After being able to receive messages via BLE and having the capability of sending them to the CAN bus, we can test this solution to emulate and substitute the adaptors used when configurations in the system are needed.
- Right now all the Bluetooth pairing is being made manually and with no security. For the future a mechanism that creates this connection would be ideal. As well as being protected from attacks.
- After developing the python script to read and send the Bluetooth messages through Bluetooth from the computer to the microcontroller, the integration of this script in the Addvolt app would give better overview of the protocol.
- After the PCB is implemented and validated as a new solution there needs to be work done in the enclosure of this device.
- After testing the receiving and sending of data through the BLE connection, at different rates, we concluded that messages were lost. An improvement in the protocol created needs to be carried out.

Appendix A

Python script developed

```
with open('input.txt', mode='r') as file:
    lines = file.readlines()

sleep_amount = 0.1# In seconds
start_time = time.time()
seconds = 10
s = serial.Serial('COM8', baudrate=57600)
i=0
a=0

while True:
    current_time = time.time()
    elapsed_time = current_time - start_time

    if a:
        break

    for line in lines:
        line = line.strip() # We strip the newlines from the strings

        x = bytearray.fromhex(line)
        res = s.write(x)

        if res != len(x):
            print("Didnt all bytes!")

        s.flush()
        i+=1
        time.sleep(sleep_amount)

    if i > 199:
        print(' ' + str(i))
        i=0
        a=1
```

Figure A.1: Python script to send BLE messages

This python script is used to send Bluetooth messages from the PC through the serial port. We write any messages we want in hexadecimal in the file 'input.txt' and this script will read each line of that file and then send it. We can also control how many times we send the messages in the file and at what rate.

```

import serial
import time
import binascii
from datetime import datetime

previous_date = datetime.now()

s = serial.Serial('COM7', baudrate=57600)
start_time = time.time()
seconds = 10
a=0
i=0
var_name = []

while True:

    if a:
        break

    res = s.read(16)
    current_time = time.time()
    elapsed_time = current_time - start_time
    var_name.append(res.hex() + '\n')
    i+=1

    if elapsed_time > seconds:
        print('messages recieved: ' + str(i))
        start_time = time.time()
        i=0
        a=1

with open('output.txt', mode='w') as file:
    file.writelines(var_name)

```

Figure A.2: Python script to receive BLE messages

References

- [1] Anonymous. Developing a Bluetooth Low Energy Application, 07, 2016 , https://software-dl.ti.com/lprf/simplelink_cc2640r2atest/docs/blestack/ble_user_guide/html/ble_stack_3.x/gap.htmlconnection-parameter-considerations, [Last accessed 2022-1-20].
- [2] Anonymous. I2C Bus Protocol, 09, 2020 , <https://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/general-introduction/i2c-bus-protocol.html> , [Last accessed 2021-8-10].
- [3] Anonymous. I2C (Inter-Integrated Circuit) Bus Technical Overview and Frequently Asked Questions, 09, 2020 , <https://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus.html> , [Last accessed 2021-8-10].
- [4] Rune Bearson. What is the difference between Bluetooth versions?, 3, 2020 , <https://earrockers.com/difference-between-bluetooth-versions/> , [Last accessed 2021-4-2].
- [5] Jason Blackman. Overview of 3.3V CAN (Controller Area Network) Transceivers, 1, 2013 , www.ti.com/lit/an/slla337/slla337.pdf , [Last accessed 2021-9-2].
- [6] Jim Blom. Bluetooth Basics, 8, 2018 , <https://learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works> , [Last accessed 2021-4-2].
- [7] Steve Corrigan. Introduction to the Controller Area Network (CAN) Application Report Introduction to the Controller Area Network (CAN), 5, 2016 , <https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1648084727424> , [Last accessed 2021-9-2].
- [8] Ashish Derhgawen. Maximizing BLE Throughput Part 4: Everything You Need to Know, 09, 2020 , <https://punchthrough.com/ble-throughput-part-4/> , [Last accessed 2022-12-1].
- [9] Embeddedcentric. BLE profiles, services, characteristics, device roles and network topology, 04, 2020 , <https://embeddedcentric.com/lesson-2-ble-profiles-services-characteristics-device-roles-and-network-topology/> , [Last accessed 2021-1-11].
- [10] John Griffin. Why are termination networks in CAN transceivers so important?, 7, 2016 , https://e2e.ti.com/blogs/b/industrial_strength/posts/the-importance-of-termination-networks-in-can-transceivers, [Last accessed 2021-9-2].
- [11] Hemant. AMOLED vs OLED vs LCD Display explained, 12, 2020 , <https://www.thewindowsclub.com/amoled-vs-oled-vs-lcd-display-explained> , [Last accessed 2021-8-20].

- [12] Igor. What Is E-Paper Display? What Are Its Advantages And Disadvantages?, 12, 2020 , <https://www.electronicsforu.com/technology-trends/learn-electronics/e-paper-display-advantages-disadvantages> , [Last accessed 2021-11-20].
- [13] Infineon. XMC4300 Relax EtherCAT® Kit-V1.1, 11, 2015 , <https://www.infineon.com/dgdl/Infineon-XMC4300RelaxKit-UM-v0100-EN.pdf> , [Last accessed 2021-8-20].
- [14] Jackson. Bluetooth 5.1 vs 5.2, 2, 2021 , <https://audio46.com/blogs/headphones/bluetooth-5-1-vs-5-2> , [Last accessed 2021-4-2].
- [15] Kumar. UART Communication Protocol – How it works?, 1, 2022 , <https://www.codrey.com/embedded-systems/uart-serial-communication-rs232/> , [Last accessed 2022-1-20].
- [16] Mutjaba Lane. The Serial Peripheral Interface (SPI), 09, 2015 , <http://avrbeginners.net/architecture/spi/spi.html> , [Last accessed 2021-8-10].
- [17] Zach Michel. Maximizing BLE Throughput Part 3: Data Length Extension (DLE), 04, 2019 , <https://punchthrough.com/maximizing-ble-throughput-part-3-data-length-extension-dle-2/> , [Last accessed 2022-12-1].
- [18] MicroChip. MCP23017/MCP23S17, 8, 2006 , <https://download.mikroe.com/documents/datasheets/mcp23017-datasheet-21952b.pdf> , [Last accessed 2022-8-19].
- [19] MikroElektronika. BLE TINY Click schematic v101, 4, 2018 , https://download.mikroe.com/documents/add-on-boards/click/ble_tiny_click/ble-tiny-click-schematic-v101.pdf , [Last accessed 2021-11-20].
- [20] MikroElektronika. LCD mini click schematic v101, 8, 2019 , <https://download.mikroe.com/documents/add-on-boards/click/lcd-mini/lcd-mini-click-schematic-v101.pdf> , [Last accessed 2021-8-20].
- [21] MikroElektronika. OLED C click board™ Schematic v101, 8, 2019 , <https://download.mikroe.com/documents/add-on-boards/click/oled-c/oled-c-click-schematic-v101.pdf> , [Last accessed 2021-8-20].
- [22] MikroElektronika. BLE TINY Click, 4, 2018 , <https://www.mikroe.com/ble-tiny-click> , [Last accessed 2021-11-20].
- [23] MikroElektronika. OLED C CLICK, 8, 2019 , <https://www.mikroe.com/oled-c-click> , [Last accessed 2021-8-20].
- [24] Nora. Bluetooth network connection pairing, 5, 2015 , <https://www.electronics-notes.com/articles/connectivity/bluetooth/network-pairing-connection.php> , [Last accessed 2021-4-2].
- [25] Eric Pena. UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter, 12, 2020 , <https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html> , [Last accessed 2021-8-10].
- [26] Pat Richards. A CAN Physical Layer Discussion, 2, 2002 , <https://ww1.microchip.com/downloads/en/appnotes/00228a.pdf> , [Last accessed 2021-9-2].

- [27] SHENZHEN. PS11096096FR013, 10, 2013 , <https://download.mikroe.com/documents/datasheets/psp27801-product-specification-a0.pdf> , [Last accessed 2021-8-20].
- [28] Sitronix. Dot Matrix LCD Controller/Driver, 10, 2010 , https://www.newhavendisplay.com/app_notes/ST7066U.pdf , [Last accessed 2021 – 8 – 20].
- [29] Solomon Systech. SSD1351, 10, 2009 , <https://download.mikroe.com/documents/datasheets/ssd1351-revision-1.3.pdf> , [Last accessed 2021-8-20].
- [30] Topway. Comparison - TFT LCD vs OLED vs Micro LED, 1, 2021 , <https://www.topwaydisplay.com/en/blog/lcd-oled-microled-comparison> , [Last accessed 2021-8-20].
- [31] Conal Watterson. AN-1123 APPLICATION NOTE Controller Area Network (CAN) Implementation Guide, 11, 2017 , <https://www.analog.com/media/en/technical-documentation/application-notes/AN-1123.pdf> , [Last accessed 2021-9-2].
- [32] Staff Writer. 5 Advantages of CAN Bus Protocol, 8, 2019 , <https://www.totalphase.com/blog/2019/08/5-advantages-of-can-bus-protocol/> , [Last accessed 2021-9-2].
- [33] Yida. UART vs I2C vs SPI – Communication Protocols and Uses, 3, 2020 , <https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/> , [Last accessed 2022-1-20].