RICARDO CORREIA

DETECÇÃO COOPERATIVA DE CONDIÇÕES CLIMATÉRICAS ADVERSAS DAS ESTRADAS USANDO SENSORES DOS VEÍCULOS

COOPERATIVE IN-VEHICLE SENSING OF ADVERSE ROAD-WEATHER CONDITIONS

Universidade de Aveiro
2021

RICARDO CORREIA

DETECÇÃO COOPERATIVA DE CONDIÇÕES CLIMATÉRICAS ADVERSAS DAS ESTRADAS USANDO SENSORES DOS VEÍCULOS

COOPERATIVE IN-VEHICLE SENSING OF ADVERSE ROAD-WEATHER CONDITIONS

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor Joaquim José de Castro Ferreira, Professor adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro, e do Doutor Paulo Jorge de Campos Bartolomeu, Investigador Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Manuel Bernardo Salvador Cunha
Professor Auxiliar, Universidade de Aveiro

vogais / examiners committee

Prof. Doutor José Carlos Meireles Monteiro Metrôlho
Professor Adjunto, Instituto Politécnico de Castelo Branco

Prof. Doutor Joaquim José de Castro Ferreira
Professor Adjunto, Universidade de Aveiro

agradecimentos / acknowledgements

Chegado o final de mais uma etapa da minha vida, não posso deixar de agradecer a todas as pessoas que fizeram parte deste caminho.
Aos meus orientadores, Prof. Joaquim Ferreira e Prof. Paulo Bartolomeu, agradeço por todas as ideias, apoio e orientação ao longo da realização deste trabalho. Ao João Almeida, pelo acompanhamento prestado ao longo deste trabalho.
Aos meus amigos do mítico Clube dos Mansos, quero agradecer por todo o apoio e momentos de partilha e descontração. Quero também fazer um agradecimento especial ao meu melhor amigo, Nuno Costa, por estar sempre cá para mim, mesmo quando a vida não é fácil de levar.
Aos meus avós, pais e tias, por todos os valores que me transmitem, por me proporcionarem sempre o melhor, por apoiarem todas as decisões que tomo ao longo do meu percurso e por toda a paciência que têm comigo.
À Adriana, pelo apoio incondicional e partilha em todos os momentos.
Estou-vos eternamente grato, obrigado!

Este trabalho é financiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER), através do Programa Operacional Competitividade e Internacionalização (COMPETE 2020) do Portugal 2020 [Projeto TRUST com o nº 037930 (POCI-01-0247-FEDER-037930)];

Cofinanciado por:

COMPETE 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

Palavras Chave

redes intra-veiculares, condições meteorológicas, condições de estrada, comunicações veiculares, sistemas inteligentes de transportes, segurança rodoviária

Resumo

A segurança rodoviária apresenta uma grande relevância durante os últimos tempos, dado o aumento da utilização de veículos e a necessidade de proteger os passageiros. Umas das principais causas dos acidentes rodoviários está diretamente relacionada com as condições meteorológicas, nomeadamente a chuva e o piso molhado, que diminuem a visibilidade e a estabilidade do carro, respectivamente. Atualmente, apenas são usadas estações meteorológicas para determinar as condições do tempo, porém, os sistemas inteligentes de transportes estão a tornar-se mais complexos e a adotar novos modelos para interpretar as condições do meio e difundir a informação de forma mais rápida e eficiente. Ao nível dos sistemas de transportes inteligentes cooperativos (C-ITS), pretende-se atingir esse propósito com base em sensorização cooperativa e Internet das Coisas (IoT), garantindo uma maior coordenação, quer pela informação disponibilizada ao condutor, a partir de sistemas de comunicação como V2V (veiculo-para-veiculo) e V2I (veiculo-para-infraestrutura), quer pela utilização dos próprios sensores dos veículos como referência das condições de circulação. Neste trabalho desenvolveu-se um algoritmo interativo com o condutor para identificar os parâmetros necessários para o estudo das condições meteorológicas, através dos sinais proprietários que se encontram especificados na comunicação intra-veicular, como por exemplo o estado do limpa pára-brisas para identificar situações de aguaceiros ou chuva intensa, a posição das luzes como forma de avaliar a visibilidade da estrada e a velocidade das rodas para analisar o estado do piso e possíveis causas de acidentes. Devido às diferentes representações que os parâmetros apresentam, o algoritmo é constituido por vários métodos, implementando diferentes processos para a detecção de parâmetros em que só apresentam um bit de informação e para parâmetros que dispõem de um ou mais bytes de informação. Com o recurso métodos de engenharia reversa, o objetivo passa, não só, por interpretar as transições que vão acontecendo nos sinais e, após um processo de filtragem, detetar a posição e o identificador da mensagem onde se encontra o parâmetro pretendido, mas também, por correlacionar e comparar esses sinais com as respostas de diagnóstico OBD-II (On-Board Diagnostics) sem ser necessário aceder a um vasto conjunto de dados, num curto espaço de tempo. Posteriormente, os parâmetros são transmitidos para uma plataforma OBU que difunde os dados através de mensagens cooperativas para a rede veicular. Foram realizados testes em dois carros, sendo os resultados obtidos satisfatórios. No primeiro carro foram encontrados todos os parâmetros binários, mas não se obteve o mesmo resultado nos parâmetros não-binários, contrariamente ao segundo carro. Contudo, estes acontecimentos permitiram adquirir conhecimento sobre a forma como os fabricantes de automóveis desenvolvem os seus sistemas intra-veiculares. Conclui-se assim que este trabalho acrescenta um passo importante nesta área.

Keywords in-vehicle networks, weather conditions, road conditions, vehicular communications, intelligent transport systems, road safety

Abstract Given the increasing use of vehicles and the need to protect passengers, road safety has been of great importance in recent times. One of the main causes of road accidents is directly related to weather conditions, namely rain and wet conditions, which respectively decrease the visibility and stability of the car. Currently, weather stations are used to determine only the weather conditions. However, intelligent transportation systems are becoming more complex and adopting new models to interpret the conditions of the environment and disseminate the information more quickly and efficiently. Cooperative Intelligent Transport Systems are meant to achieve this purpose based on cooperative sensors and the Internet of Things (IoT), while ensuring greater coordination, either by the information made available to the driver, from communication systems such as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I), or by using the vehicle sensors as a reference for traffic conditions. An interactive algorithm was developed, alongside the driver, to identify the necessary parameters to study the weather conditions. This algorithm is divided into several processes, that are specified within the intra-vehicular communication through proprietary signals, such as the state of the windshield wiper to detect rain, headlight position as a way to assess visibility and wheel speed to analyze the state of the surface and possible accident causes. The algorithm is composed by several methods, due to the different representations presented by the parameters. Thourght implementing different processes of detection parameters represented by only one bit of information and for parameters that have one or more bytes of information. Through using reverse engineering, the purpose is not only to interpret the transitions that occur in the signals and, after a filtering process, detect the position and the message identifier of the desired parameter, but also to correlate and compare these signals with the OBD-II (On-Board Diagnostic) diagnostic responses without a large set of data and in a short period of time. Afterwards, the parameters are transmitted to an On-Board Unit (OBU) platform that broadcasts the data using cooperative messages to the vehicle's network. Tests were performed in two cars and the results obtained were satisfactory. All binary parameters were found in the first car, but not in the non-binary parameters, whereas on the second car is founded. However, these events allowed to acquire knowledge about the matter in which car manufacturers develop their intra-vehicular systems. It can, therefore, be concluded that this work adds an important step in this field.

Contents

List of Figures

List of Tables

Glossário

FR	Front Right
FL	Front Left
RR	Rear Right
RL	Rear Left
CAN	Controller Area Network
OEM	Original Equipment Manufacturer
SAE	Society of Automotive Engineers
OBD	On-Board Diagnostics
ITS	Intelligent Transport System
C-ITS	Cooperative Intelligent Transport System
IoT	Internet of Things
ETSI	European Telecommunications Standards Institute
WAVE	Wireless Access in Vehicular Environments
IEEE	Institute of Electrical and Electronics Engineers
TRUST	Transportation and Roadmonitoring system for UbiquitouS real-Time information services
ECU	Electronic Control Unit
LTE	Long Term Evolution
BLE	Bluetooth Low Energy
DLC	Data Lenght Code
RTR	Remote Transmission Request
OBU	On-Board Unit
USB	Universal Serial Bus
RSU	Road Side Unit
NBR	Nominal Bit Rate
NBT	Nominal Bit Time
PDU	Protocol Data Unit
SID	Service IDentification
PID	Parameter IDentification
DTC	Diagnostic Trouble Code
CARB	California Air Resources Board
M2M	Machine-to-Machine
API	Application Programming Interface
SDK	Software Development Kit
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
GSM	Global System for Mobile Communication
VLC	Visible Light Communications
DSRC	Dedicated Short Range Communications
LTE-V	LTE-Vehicular
5G	Fifth Generation
mmWaves	Millimeter Waves
MAC	Medium Access Control
ML	Machine Learning
LSB	Least Significant Bit
MSB	Most Significant Bit
GAP	Generic Access Profile
GATT	Generic Attribute Profile
ATT	Attribute Protocol
CRC	Cyclic Redundancy Check
CiA	CAN in Automation
FD	Flexible Data
ISO	International Organization for Standardization
SOF	Start-of-Frame
EOF	End-of-Frame
SPP	Serial Port Protocol
RMSE	Root-Mean-Square Error
LIN	Local Interconnect Network
AFH	Adaptive Frequency Hopping
TDMA	Time Division Multiple Access
BER	Bit Error Rate
CAM	Cooperative Awareness Message
DENM	Decentralized Environmental Notification Message
TCU	Telematic Control Unit
ARM	Advanced RISC Machines
ABS	Anti-lock braking system
EPA	Environmental Protection Agency

CHAPTER 1

Introduction

1.1 Scope and Motivation

Mobility is not a recent concept. The innovation of transportation facilities arose from the need to transport people and is associated with the development of humanity. Nowadays, a vehicle is indispensable in our daily lives. This statement can be seen in the data on vehicles existing worldwide. The number of vehicles on the roads has been continuously increasing, reaching one billion units in 2010, which corresponds approximately to one vehicle for every seven people [1]. Vehicle production and sales growth continued, reaching a peak of 80 million cars sold in 2018. However, sales currently show a drop of 20 million compared to 2018, standing at 60 million, with the COVID-19 pandemic as one of the leading causes of this decline [2] [3]. As a result of the growth in the number of vehicles on the roads, concerns have arisen about road safety and efficiencies, such as road accidents, traffic jams, energy consumption, and air pollution. The implementation of measures to cope with the negative consequences and associated dangers such as Anti-lock braking system (ABS) or airbags has had very positive results worldwide, thus reducing the number of fatalities associated with road accidents [4].

Air pollution is now a key concern of automotive manufactures. In the 1960s, vehicle pollution became a cause for concern. Due to smog problems in Los Angeles, the state of California ordered them to create emission control systems to meet Environmental Protection Agency (EPA) emission standards. The companies started to produce their systems, but in 1988, the Society of Automotive Engineers (SAE) established a connector and a set of standard tests so that they all have the same technology. In 1996 the On-Board Diagnostics (OBD)-II was created with improvements in its standards and became mandatory, in the United States, for all cars manufactured since then. Thanks to this protocol and with the boost in the cars'

electronic development, it is possible to access several parameters of the vehicle through the OBD-II port present in the car [5].

The success of these security measures has not prevented the search for new solutions further to reduce the road risks. The main objective is to eliminate the number of deaths and severe injuries on the roads, and therefore, new measures are needed to achieve these goals. The introduction of information and communication technologies in road systems, associated with the development of Intelligent Transport System (ITS), allows the exchange of information between the various elements of the road system, anticipating dangerous situations. The growing interest in the study of vehicular communications has brought the publication of two standards: Wireless Access in Vehicular Environments (WAVE), defined by the Institute of Electrical and Electronics Engineers (IEEE) and ITS-G5, developed by the European Telecommunications Standards Institute (ETSI). Another decisive factor for road safety is the Internet of Things (IoT) technology associated with new sensors included in vehicles, road infrastructures, and compact devices with large capacities for data collection and transmission. Vehicle communication substantially impacts traffic handling, accident prediction, route optimization and direction [6].

One of the leading causes of road accidents is weak weather conditions, which can cause poor visibility and loss of control of the car due to the state of the road in situations of rain and wet conditions. The monitoring of weather conditions and the early warning of adverse conditions through sensors and vehicular communications can drastically reduce the occurrence of accidents or, in cases of unexpected accidents, the dissemination of warning messages relying on the vehicular network, including those responsible for traffic control. Therefore, reducing the probability of secondary events arising from that accident.

Road weather conditions monitoring can be improved using vehicle sensors, albeit indirectly, and smartphone sensors. On-board data collected from the OBD-II port and the smartphone's camera can play a decisive role in assessing road weather conditions. For example, CAN messages from the OBD-II port with information of various parameters such as the state of the headlights, the state of the windshield wipers, vehicle speed, wheels speed, and external air temperature, can be used to inform other vehicles in the vicinity or even to the cloud. The increasing modernization of vehicles has brought a growing number of integrated sensors and actuators, causing new security concerns, particularly in automated vehicles [7] [8].

1.2 Background

This dissertation is part of a project called Transportation and Roadmonitoring system for UbiquitouS real-Time information services (TRUST) that aims to develop an environmental and weather monitoring system capable of identifying risk conditions for driving in road infrastructures, equipped with ITS technologies. Communication between vehicles and infrastructures is intended to alert drivers in transit and warn about the danger of accidents in specific locations, reducing the number of incidents and associated fatalities.

The work contributes to the collection of weather-related Controller Area Network (CAN) parameters. Nowadays, an Electronic Control Unit (ECU) in modern vehicles produces

enormous amounts of data, which can be used to collect information regarding each vehicle's condition and usage. Furthermore, if wireless communications are available, these data offer significant potential for the development of Cooperative Intelligent Transport System (C-ITS) applications, including traffic management, traffic information, road safety, weather-related notifications, among other purposes.

Data from vehicle integrated sensor systems provides an improved perception of the surrounding environment. As a result, the TRUST project aims to integrate the CAN/OBD-II interface with the ITS-G5/Long Term Evolution (LTE) vehicular communications platforms on the fleet vehicles, enabling sensor-data collection to infer weather-related events in the area. A shortlist of parameters that can, at least in some vehicle models, be measured in the OBD-II interface or directly from the CAN-BUS, are presented in Table 1.1. This literature overview identifies the target of the proposed ITS applications in each work and the specific data monitored.

Reference, Year	Goal	Parameters
Enriquez et al. [9], 2012	Slippery road conditions detection	Throttle pedal position, wheel speed, vehicle speed, steering wheel angle, and brake pressure
Geem et al. [10], 2016	Road surface distress detection	Wheel speed, steering wheel angle, ToF camera data
Hou et al. [3], 2017	Slippery road conditions detection	Wheel speed, vehicle speed
Iqbal et al. [11], 2017	Remote online diagnostic system	Engine speed, vehicle speed, engine fault detection
Galanis et al. [12], 2019	Speed recommendation based on road conditions	Engine speed, engine load, axle ratio, transmission rate, vehicle stability, position and speed
Abuali [13], 2015	Monitoring road artifacts and driver behavior	Throttle pedal position, brake pedal position, steering wheel angle
Bartos et al. [14], 2019	Weather assessment	Windshield wiper data
Chapman et al. [15], 2010	Road weather conditions evaluation	Ambient temperature, wiper status, ABS data, rain sensor, etc
Mahoney et al. [16], 2013	Weather and road conditions evaluation	Ambient temperature, wiper status, ABS data, rain sensor, headlamps, atmospheric pressure, sun sensor, engine speed, etc

Table 1.1: Vehicle sensor data collection for weather-related ITS applications, Adapted from TRUST report

1.3 Objectives

Developing an algorithm to discover weather-related parameters in the in-vehicle networks requires a few objectives to achieve this goal:

- Acquire knowledge about serial communication concepts and vehicular communications. Serial communication plays an important role in intra-vehicular networks, and therefore, the communication protocol used in vehicles is studied. The European standard that defines the functions of the various protocol layers in the communications between vehicles, and the highest layer on the cooperative messages, is also studied.
- Research articles that have contributed to the development in the area of identifying parameters in intra-vehicle messages. Studying state-of-the-art is essential to understand what already exists and needs to be improved, offering value to this work.
- Design and implement several processes and methods to decode the weather-related data more faster than the existing works.
- Discover the best solution to transmit the data to an On-Board Unit (OBU) platform and communicate it to the vehicle network with one of the cooperative messages defined by C-ITS protocols.
- Make the best contribution to the TRUST project to which this work is integrated.

1.4 Document Organization

The rest of the dissertation is organized as follows:

- **Chapter 2 - Fundamental Concepts and State-of-the-Art** - This chapter explains in detail the concepts covered in this dissertation;
- **Chapter 3 - System Architecture** - Exposes the contextualization of this dissertation and the architecture of the proposed system;
- **Chapter 4 - Implementation** - Describes the implementation of the solutions found;
- **Chapter 5 - Tests and Validation** - Presents the description of the various test scenarios and the results obtained to validate the implementation;
- **Chapter 6 - Conclusions and Future Work** - Presents the conclusion taken from the results and future work.

CHAPTER 2

Fundamental Concepts and State-of-the-Art

This chapter introduces the related concepts of this dissertation and the State-of-the-Art CAN reverse-engineering algorithms. Firstly, the CAN protocol that has been responsible for the communication in the intra-vehicular networks are presented, from the physical layer to the higher layers, such as OBD-II, used to communicate between the vehicle and an external equipment. Secondly, it is discussed the Bluetooth Low Energy (BLE) concept used for the communication between the CAN-Reader and the OBU placed in the vehicle, which transmits and receives the vehicles sensor-data with Road Side Units (RSUs). This communication of cooperative-data between these last two elements is specified in the following topic, which concerns ITS and C-ITS technologies. In the end, the description of the reverse engineering algorithms is considered chronologically in order to show the evolution over the last few years.

2.1 Controller Area Network

Robert BOSCH, in 1986, developed a serial communication protocol for the automotive industry called the CAN protocol [17].

CAN is a message-oriented transmission protocol. Every message has an associated message identifier, which is unique within the whole network and defines both the content and the message's priority. One of the main advantages of this protocol is its flexibility, as it can be used in systems of low latency or systems with a large bandwidth. Simplicity is another benefit of this protocol, whereas an extensive number of wires was reduced to only two communication wires. Formerly, controllers were connected by a very complex wiring system in the automotive industry. However, when this protocol appeared, the complexity of cable wiring vanished, and the communication became simpler, reaching now up to 1Mbit/s [18].

Here are some examples of other characteristics of this protocol:

- Every node can be master at one time and slave at another time (Multi-Master);

- When a message is transmitted, all nodes can receive it simultaneously (Multi-Cast);
- Messages with smaller IDs have a higher priority than messages with larger IDs;
- Detect message errors;
- Automatic re-transmission of corrupted messages;
- Distinction between temporary errors and permanent errors;

2.1.1 Physical Layer

One of the CAN physical layers, available in most of CAN transceivers, is specified in the International Organization for Standardization (ISO) 11898-2 standard [19]. This physical layer protocol defines the bit timing, synchronization, and physical signaling.

In terms of bit timing, the protocol defines the Nominal Bit Rate (NBR) as the number of bits transmitted per second to the bus. The equation is given by:

$$NBR = \frac{1}{NominalBitTime} \tag{2.1}$$

The NBT is the bit period and is distributed into three different segments. Figure 2.1 shows the division between them.

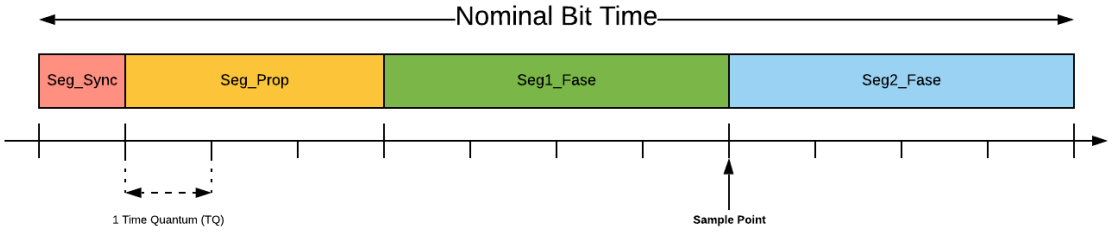


Figure 2.1: Representation of the segments in the NBT time-frame

The figure above displays unique colors to distinguish the different segments. The segment represented by the color red is the *Synchronization Segment* that is used to synchronize the nodes of the bus, followed by the *Prop Segment* represented with the color yellow. This segment compensates for the delays that may occur due to the bus line and delays caused by the transceivers. The next two segments can be adjusted, depending on the calculations of time differences, to synchronize the transmitter and the receiver.

The CAN physical layer protocol also specifies cabling types, electrical signal levels and the bus termination values to suppress reflections [20].

Using two termination resistors at each side of the bus is a critical factor for efficient communication. Only one resistor or none increases the delay of the transition between a dominant state and a recessive state. In some cases, the transition may even not exist, causing a bit-error [21]. The external equipment must have the termination shown in figure 2.2, and the values are described in table 2.1.

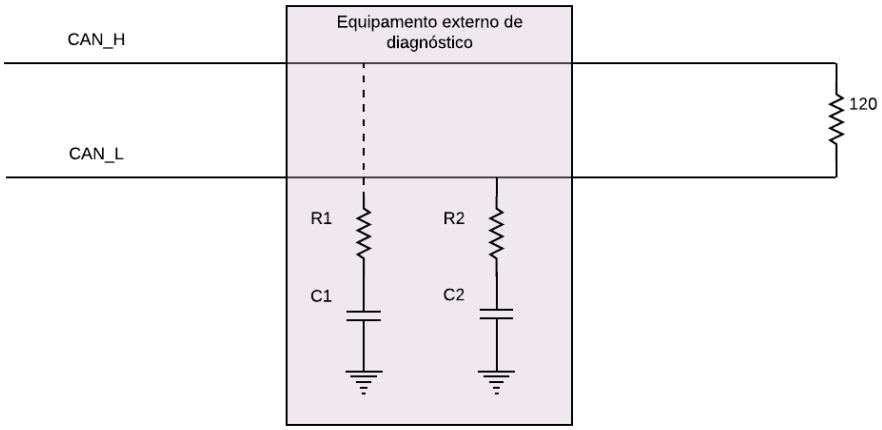


Figure 2.2: External components used for termination of the CAN bus

Table 2.1: Minimum, Nominal and Maximum values of the parameters represented in figure 2.2

Parameters	Minimum	Nominal	Maximum
R1, R2 (Ω)	90	100	110
C1, C2 (pF)	470	560	640
	R1 = R2		
	C1 = C2		

2.1.2 Data Link Layer

The CAN data link layer specifies the format, type, and structure of the CAN messages.

Until the appearance of the new 2.0 specification [17], the CAN messages were consisted of only 11-bit identifiers. However, CAN 2.0 released the possibility to use 29-bit identifiers, increasing the number of messages available. According to the specification, a message with an 11-bit identifier is called *Standard Frame* and a message with 29-bit identifiers is an *Extended Frame*.

A CAN frame begins with a start bit called Start-of-Frame (SOF) that indicates the start of the message. This bit is followed by the Arbitration Field, which consists of the 11-bit identifier and the Remote Transmission Request (RTR) bit used to distinguish between a data frame (dominant bit) and a remote frame (recessive bit). The IDE bit that follows the RTR bit indicates the format of the frame, distinguishing between a standard frame and an extended frame. In the case of an extended frame, this bit is in the Arbitration Field, and more bits are added to the identifier, thus changing to 29 bits. In the case of a standard frame, the IDE bit is in the Control Field, such as the Data Lenght Code (DLC) used to indicate the number of bytes in the Data Field. If the message type is a remote frame, the DLC contains the number of bytes of the message reply. The Data Field contains the message content, up to 8 bytes. As for the error detection, the messages contain a Cyclic Redundancy Check (CRC) Field with a CRC sum and a CRC delimiter. The ACKnowledge Field (ACK)

consists of the ACK slot and ACK delimiter. The transmitter sends the ACK slot bit as a recessive bit and, if the message was received correctly, it is written as the dominant bit by the receivers. The end of the frame is indicated by the End-of-Frame (EOF) bit.

Figures 2.2 and 2.3 represents the structure of a standard and extended message, referring to data/remote frames, respectively.

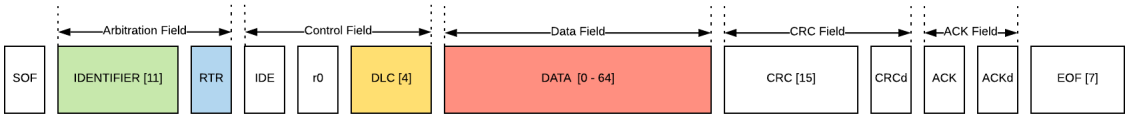


Figure 2.3: Structure of a CAN Standard Frame

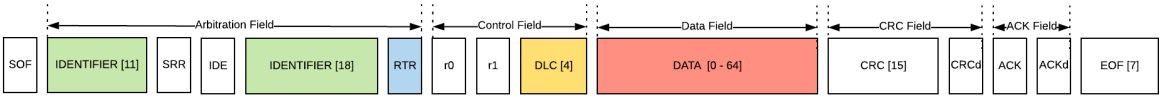


Figure 2.4: Structure of a CAN Extended Frame

Besides these two types of messages, there are four types of messages: *Data Frame*, *Remote Frame*, *Error Frame*, and *Overload Frame*. The *Data Frame* is the only type that carries data between the transmitter and the receiver. Depending on the identifier's size, the format can be standard or extended and can carry data up to 8 bytes, according to the value of DLC parameter. In this type of frame, the RTR bit is dominant ('0'). The *Remote Frame* does not have data associated and it is transmitted when a given node wants to request data from another node. This type of message is identified by the RTR bit, in which it is recessive ('1'). The Error Frame is only transmitted when it detects an error in the bus. This type of message consists of two distinct fields. The overlap error flags give the first field from different stations. The second field is the error delimiter. The node that detects the error in the message sends an error message with a specific flag. The active flag is transmitted when an *active error* is detected and consists of 6 consecutive dominant ('0') bits. The *passive flag* is transmitted when a passive error is detected, consisting of 6 recessive bits ('1'). The second field is the error delimiter, consisting of 8 recessive bits. The Overload Frame consists of two bit-fields: overload flag and overload delimiter. There are two conditions for an overload frame to be transmitted:

- Internal conditions of a receiver, which requires a delay of the next data or remote frame;
- The detection of a "dominant" bit during intermission.

After transmitting this flag, any node is attentive to the bus until it detects a "recessive" bit. The second field is the overload delimiter, consisting of 8 bits recessives.

It is important to note that the Data and Remote Frames are separated from preceding frames called by a bit field called Interframe Space [22]. It consists of two fields:

- **Intermission** - consists of 3 recessive bits and, during intermission, a transmission of a Data/Remote Frame to any node is not allowed to start. The only action that can occur is an overload condition.
- **Bus Idle** - The detection of a dominant bit during the bus-free state is interpreted as the beginning of the frame. This field can have a random period.

2.1.3 Detecting and signaling errors

Within the various concepts that have already been presented about the CAN bus, we can add one more called *Error Handling*. *Error Handling* aims to detect errors in the messages that appear on the CAN bus so that the transmitter can re-transmit an incorrect message. All nodes along the bus will try to detect errors in a message, and if an error is detected, the node that found it will transmit an error signal. The remaining nodes will detect the error caused by the flag node and discard the current message.

For error detection, the CAN protocol specifies five types of errors, three at the message level and two at bit level:

- **Cyclic Redundancy Check Error** – when the transmitter sends a frame, redundant bits are placed in the CRC field. The error occurs when the value of this receiver field does not match with the value sent by the transmitter;
- **Format Error** – The error is detected when a given field has a different bit than foreseen, i.e., in a field where the bit should be recessive and is dominant. Consequently, this violates the rules for space;
- **Acknowledgment Error** – When it is not detected a dominant bit in ACK field;
- **Bit monitoring** – The transmitter, when it is sending a message, can to monitor the bus. Then, if the node detects a different bit from the one it sent, this type of error is generated;
- **Bit stuffing** – When there are six consecutive bits with the same logical level.

The errors detected are flagged by sending an error message immediately after the error is detected. The type of flag to be sent is defined by the state of the node and can be considered *error active* or *error passive*. When an active error is detected, the node sends an *Active Error Flag*, instead of a passive error where a *Passive Error Flag* is sent.

Each node contains two error counters, one for transmission errors (Transmit Error Counter) and one for reception errors (Receive Error Counter). Each time an error is detected, the respective counter is incremented by 8.

- If both counters have a value below 128, the node is *error active*;
- If one of the counters has a value over 128, the node is *error passive*.

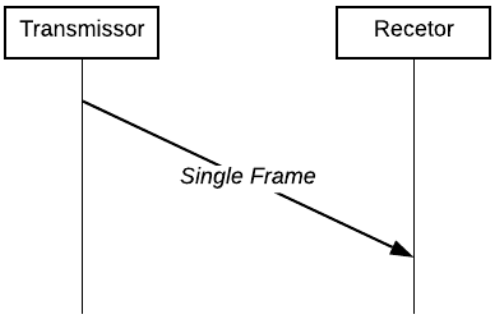
If the transmission counter reaches the value of 256, then the node goes into the bus-off state, and it can no longer communicate with the other nodes and cannot send messages to the bus. However, they can continue monitoring the bus, and after 128 consecutive 11 recessive bits, their counters return to 0.

If a message is sent without errors, the counters will be decremented by 1, unless they are 0.

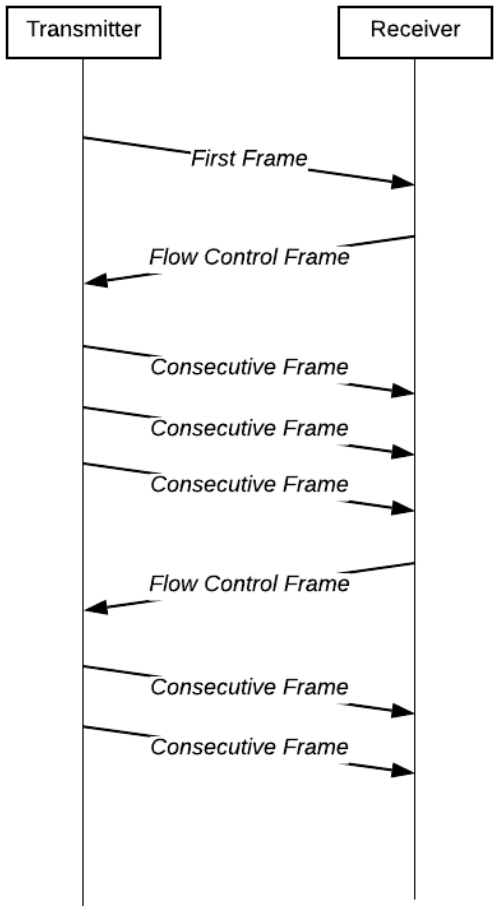
2.1.4 Network Layer

This layer is in charge of the data exchange between nodes. It also specifies a segmentation method for nodes that cannot fit all the data they want to transmit in a single CAN frame [23].

Figure 2.5a shows an unsegmented message transmission. In the case of segmented message transmission, the data split into various CAN frames and use a *Flow Control* technique to not losing any fragmentation. Figure 2.5b shows an example of a segmented message transmission.



(a) Unsegmented message transmission



(b) Segmented message transmission

Figure 2.5: Unsegmented and Segmented message transmissions

In this layer, the transmission and reception of messages with 6 or 7 data bytes are performed to transmit of a unique Protocol Data Unit (PDU). For unsegmented messages, the transmission of a PDU is called *Single Frame*.

When the message is longer, the transmission occurs with a multiple PDUs. The multiple PDUs are called *First Frame* (for the first message) and *Consecutive Frame* (for all the following PDUs). The receiver adjusts the transmission by responding with a *Flow Control PDU*. Table 2.2 provides a more detailed description of each one.

Table 2.2: Descriptions of the four types of Protocol Data Units

PDU	Description
Single Frame	Unique message to transfer unsegmented messages
First Frame	First message sent to the receiver, indicating the beginning of a fragmented message
Consecutive Frame	Consecutive segments sent after the First Frame
Flow Control Frame	Instruct the transmitter to start, stop or resume transmission of Consecutive Frames

2.1.5 Application Layer

CAN application layer is specified in the ISO 15765. This standard defines the requirements for the operations between the CAN vehicle-reader, and the CAN network. More generally, we can state that this standard is a CAN for vehicles [24].

In general, this layer specifies a set of diagnostic services performed on a vehicle using an external device. This standard features several services and standardized tables with various parameters to perform a reading of some sensors in real-time or even a code system to diagnose problems or failures in the vehicles [25]. The communication between the device and the vehicle is made through exchanging messages. All of the sensors available in the PID tables can be requested by a message with the type of service and the intended parameter. After that, a particular ECU sends a response-message with the value of the sensor.

These messages contain a Service IDentification (SID) that can vary between 01 to 09, a PID that identifies the standard parameters and data (in a response-message). Figures 2.6 and 2.7 show the request-message and response-message structures.

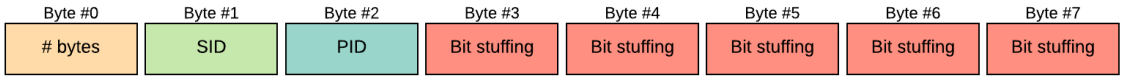


Figure 2.6: Structure of a request-message with the respective configuration

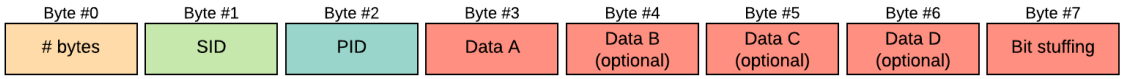


Figure 2.7: Structure of a response-message with the respective configuration

It is also important to note that, for systems with CAN bus based on ISO 15765, all ECUs must respond to a request message during a specific period. This timer is called P2CAN and has a maximum value of 50 ms.

The message exchanges between the device and the vehicle are through the OBD-II port.

2.1.6 CAN FD and CAN XL

The vehicles' modernization requires a greater capacity of CAN communication to follow this progress by the manufacturers. Therefore, CAN in Automation (CiA) has been designed new protocols with the requirements of future in-vehicle networks.

CAN Flexible Data (FD) was released in 2015 [26] and provides transmissions up to 5 Mbits/s with a payload of 64 bytes. This growth in the number of data reduces the overhead and improves the protocol's efficiency [27]. Moreover, this new protocol presents an improvement in error detection with an improved CRC field and the introduction of the "protected stuff-bit counter" [28]. This protocol can operate with the classical CAN simultaneously, i.e., each node can transmit messages of both types. When only one node is transmitting, the bit-rate can increase because no nodes need to be synchronized. However, when several nodes are transmitting simultaneously, they have the flexibility to change the type of messages and move to the classic style [29].

CAN XL is a current protocol, still in development, and it is specified as the third-generation of the CAN data link layer protocol by the CiA members [30] [31]. CAN XL can transmit messages up to 10 Mbits/s with a payload of 2048 bytes. It is protected with a cascaded CRC, which features a Hamming Distance of 6, meaning that five randomly distributed bit-errors are detected [32]. Additionally, this protocol will force the physical layer to be upgraded, producing new transceivers to operate with all the other transceivers that support CAN and CAN FD [33] [34].

2.2 OBD-II

Nowadays, the pollution caused by cars is no longer as debated as it was last century. Over the years, solutions to combat this problem have emerged, such as electric cars [35]. However, in 1988, the California Air Resources Board (CARB) introduced the OBD to reduce the air pollution caused by traffic. The emissions were controlled with an ECU, which monitored the sensors' data and restricted the emissions on that basis [36].

SAE released the first board OBD-I. Years later, with the improvement of the technologies, more sensors are added, improving the number of ECUs and the diagnostics, creating the OBD-II. Since 1996, cars have been equipped with OBD-II technology, and this interface provides real-time information/data from any part of the vehicle.

The system is accessed with a 16-pin connector [37], named J1962 connector. This connector is used for OBD communications and has two pins, 6 and 14, to allow access to the CAN bus. The other pins represented in figure 2.8 belong to other communication protocols that can be accessed through the OBD-II port.

The connector also has power pins (+12V/GND) to power the devices that connect to the vehicle.

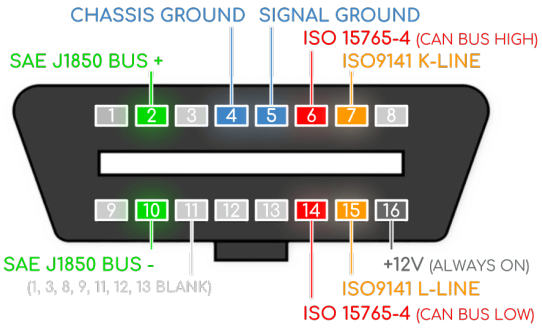


Figure 2.8: J1962 Connector, Font: [37]

2.3 CAN and OBD-II Readers

The number of CAN and OBD2 readers is quite large, from the simpler and cheapest device to sophisticated devices capable of reading standard OBD2 data or raw CAN data and sending it to an application on the mobile phone or to a cloud.

This section discusses four devices with different characteristics and different functionalities.

2.3.1 ELM 327

The ELM327 is a tiny and low-cost micro-controller designed to act as a bridge between the OBD ports and a standard RS232 serial interface [38]. Many OBD2 readers are using this micro-controller integrated with WiFi or Bluetooth to report the real-time data to an Android App. Figure 2.9 illustrates an example of a Bluetooth ELM 327 interface.

Figure 2.9: Bluetooth ELM 327 interface

These devices use Bluetooth Serial Port Protocol (SPP) to replace RS232 cables. This protocol is also suitable for sending and receiving a burst of data between two devices [39].

2.3.2 CANCrocodile

CANCrocodile is designed for safe data reading from the vehicle CAN bus, without damaging CAN wires and without electrical connections, in a contactless way. This device only works in the *listen-mode* by detecting magnetic fields around CAN-High and CAN-Low wires.

Figure 2.10: CANCrocodile

2.3.3 CLX000

The CLX000 is a CAN Bus data logger with the following characteristics:

- Plug and Play - configure in 2min and the baud rate is automatically detected;
- Standalone - log CAN data into SD card;
- Free Software;
- Live Stream - easily stream data in Wireshark;
- Compact;
- Low Cost;

The CLX000 can discover proprietary parameters through reverse engineering using Wireshark software. After configuring the dedicated software and plug the CLX000 in the car, the CLX000 stays in *listen-mode*, reading the messages on the bus. As the messages are released on the PC, the software can filter parameters in which data does not change, facilitating the search for parameters.

Figure 2.11: CLX000 products

To this work, the information from the vehicle's sensors must be transmitted to the OBU in a wireless configuration, choosing the Bluetooth technology, more specifically BLE.

2.4 *Bluetooth Low Energy*

Bluetooth Smart, also known as Bluetooth Low Energy (BLE), is designed to perform wireless communications between two short-distance devices. BLE was introduced in 2010 to reduce energy costs between data transfers, compromising a longer battery life for devices [40].

Machine-to-Machine (M2M) and IoT applications are those where BLE is most used, where the transmission of data does not need to be fast or in large quantities. This version of Bluetooth is also adopted by major brands, such as Apple and Google, that provide great BLE Application Programming Interface (API) as part of their respective mobile development Software Development Kits (SDKs). In terms of hardware, Nordic Semiconductor and Silicon Labs produce system-on-a-chip implementations for BLE that can be used to integrate with this technology [41].

2.4.1 BLE Protocol Stack

Like the classic Bluetooth, BLE has the protocol divided into two parts: the Controller and the Host. The Controller consists of the Physical Layer and the Link Layer, typically implemented in a system-on-chip with integrated radio. The Host comprises L2CAP, ATT, Generic Attribute Profile (GATT), Generic Access Profile (GAP), and the communication between the Host and the Controller through the HCI [42].

Although BLE implements the same architecture as classic Bluetooth, a device that supports only BLE cannot communicate with a device that only implements classic Bluetooth. This type of device is called a single-mode device. When implementing both types, it is called a dual-mode device [43].

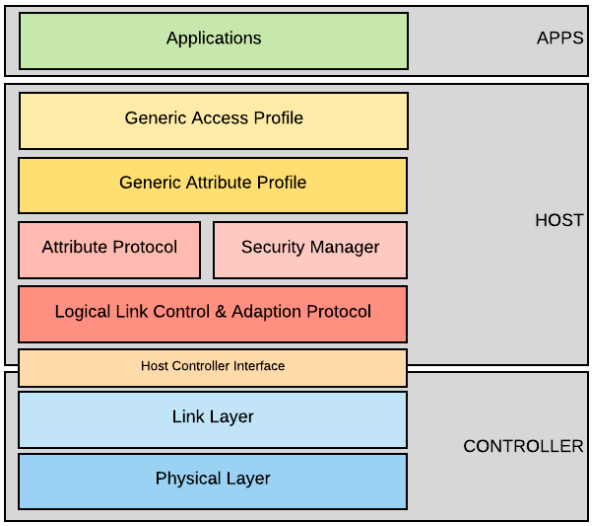


Figure 2.12: BLE Protocol Stack

Physical Layer

BLE operates in an ISM band at 2.4GHz and contains 40 radio frequency channels, separated by a 2MHz spacing. There are two types within these channels: advertising channels (used to discover devices, establish connections, and broadcast transmissions) and data channels (used for communication between two devices).

The Adaptive Frequency Hopping (AFH) technique is used on all channels to make the transmission signal more robust and reliable, adapting to interference that may arise, mainly

from Wi-Fi. Also, the channels use GFSK modulation, and the transmission rate of data is 1Mbps.

For there to be a connection between two devices, it is necessary to share the same channel of the piconet, tuning in to the same frequency at the same time. In BLE. there is no maximum number of devices that can connect to just one. However, in Classic Bluetooth, that number was limited to seven devices [42].

Link Layer

The Link Layer controls the connections between devices, defining the packets' structure, discovering, sending, and receiving the data. When a device intends to broadcast data, it transmits the data in advertising packets through the advertising channels, being the advertiser. When devices are only receiving data through advertising channels, they are called scanners.

As soon as two devices establish a connection, one becomes the master and the other the slave, and the master may have more than one slave. The slave is in sleep mode and wakes up periodically to check if the master intends to send any data packets. The master determines this period and coordinates the average access using a Time Division Multiple Access (TDMA) [43].

The operation of this layer can be described using the following state machine in figure 2.13.

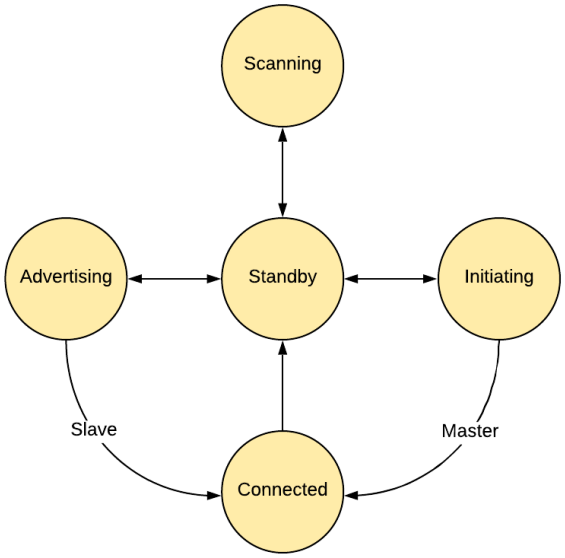


Figure 2.13: State machine of the Link Layer

The format of the packets defined by this layer is unique, both for data channel packets and for advertising channel packets, and has a length that can vary between 80 bits and 376 bits. Figure 2.14 represents the package structure.

However, we have different values in the package's various fields for the two different types of channels. The first two fields (preamble and access address) have fixed size but differ depending on the channel type.

The PDU field is the only field with variable size, varying its structure through the physical channel through which it will pass. Figure 2.15 also shows the structure of the field for each type of channel.

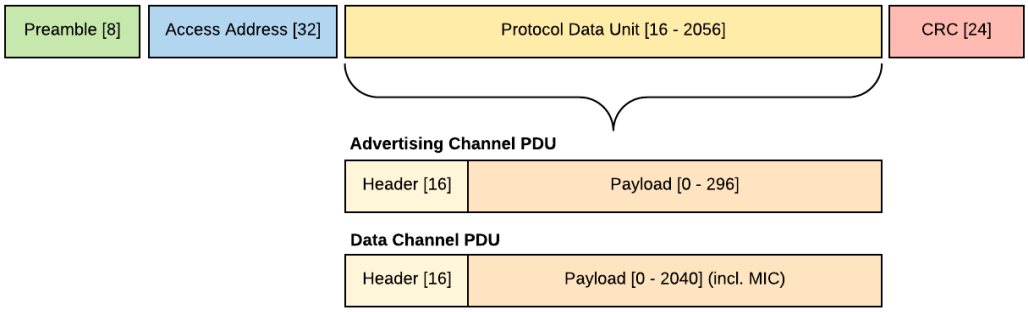


Figure 2.14: Package format and PDU field structure of two channels

The advertising channel PDUs has a 16-bit header and 6 to 37 bytes to the payload. In the payload, 6 bytes are for advertiser addresses, and 0 to 31 bytes are for data. The data channel PDUs has a 16-bit header, a payload of up to 246 bytes, and other small fields: Message Integrity Check (MIC), L2CAP Header (L2 He), and ATT Operation Code (Op) [44].

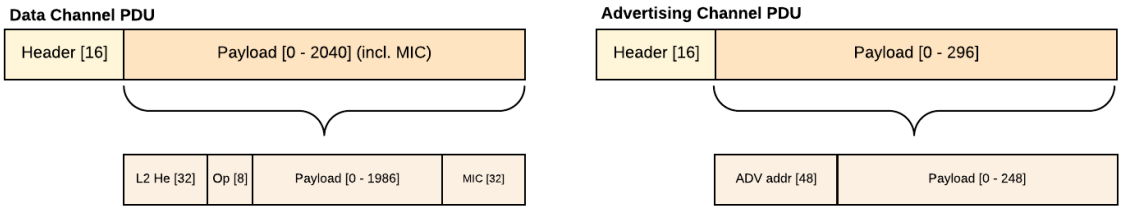


Figure 2.15: Structure of Data Channel and Advertising Channel PDUs

The CRC field is the last field of the packages and is used for error detection.

Host Controller Interface

The HCI allows and controls the communication between the host and the controller through a serial interface. Due to the controller's time requirements and the contact with the physical layer, it is important to separate from the host. Although the Host is responsible for more complex implementations, it is not demanding in time [45].

Logical Link Control and Adaptation Protocol

This protocol allows the transmission and reception of data packets from the upper layers. L2CAP supports multiplexing, segmentation. and reassembly protocols, ensuring the quality of information in the upper layers.

Attribute Protocol

In the Attribute Protocol, two roles are specified: a server role and a client role. A server can expose a group of attributes to a client that is available using the ATT protocol.

The attributes are discrete values with the following properties:

- Attribute type, defined by a Universal Unique IDentifier (UUID);
- Attribute handle;
- Group of permissions for higher layers that utilize the attribute.

Generic Access Profile

GAP aims is to control connections and advertisements, using rules described by each profile and which must be followed by other profiles. It is the GAP that also makes the device visible, establishes connections, and describes security procedures.

In BLE, there are 4 GAP roles: Broadcaster, Observer, Peripheral and Central. These four roles can be divided into two groups: those that allow the establishment of a connection, such as the the central and peripheral roles, and those that do not allow a connection, such as a broadcaster or an observer.

When a device is operating in the Broadcaster role, it only sends advertising events and, in the Observer role, it only receives advertising events. Any device in the Peripheral role accepts a connection request and will have the Slave role in the Link Layer Connection state. A device in the Central role initiates a connection request and will have the Master role in the Link Layer Connection state. This device supports multiple connections, unlike the peripheral [46].

Generic Attribute Profile

GATT describes how BLE devices transfer data back and forth using concepts like Services and Characteristics. A generic data protocol – Attribute Protocol (ATT) – is used to store Services, Characteristics and other related data in a clear lookup table using 16-bit IDs for every entry in the table [47].

GATT arises as soon as a dedicated connection is established between two devices, being present during the GATT advertising process.

It is important to remember that GATT and its connections are exclusive. When a BLE peripheral connects to a central device, it will stop advertising, and it will be impossible to other devices to connect until the prevailing connection is broken. It is essential to highlight that this connection is the only way to enable back and forth communication. The central device sends data to the peripheral and vice versa. If there is a need for data exchange between two peripherals, a mailbox system will have to be used, and consequently, all messages go through the central device. Furthermore, the communication can take occur in both directions, contrary to the one-way broadcasting approach, which uses only advertising data and GAP.

The GATT profile can take on two roles: client and server. However, a device does not need to play a role exclusively, as the same device can be a server and a client simultaneously, despite different services.

The GATT server stores the data that will be transported by the ATT, receives commands, requests and sends responses, notifications and indications to the GATT client.

The GATT client sends commands and requests to the server, receiving the respective responses, notifications, and indications sent to it about the status of the server.

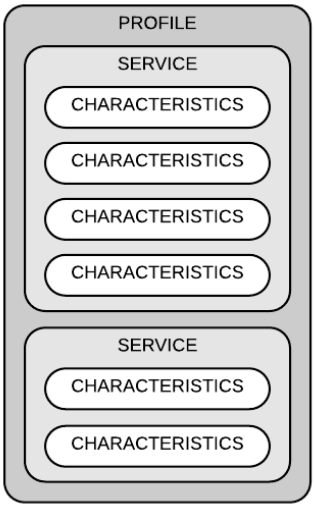


Figure 2.16: Hierarchy of BLE data

2.4.2 Communication between BLE devices

In the link layer, there is an exchange of advertising and scan packets. The advertiser, with a specific time interval, sends an advertising packet to the scanner. The first packet should indicate the type of event, and this event will influence the rest of the communication between the devices. If the scanner is interested in the scan response payload and if the event packages are *scannable* type, the scanner can send a request and receive its response.

At the GATT layer, communication is initiated by the GATT client (central) with a request to the GATT server (peripheral). As soon as the connection is established, the server responds to the client, suggesting a *Connection Interval* to the central to send requests to the server with that specific time interval to check new data from the peripheral. Figure X shows the data exchange procedure.

2.4.3 BLE Performance

It is crucial to see the performance that a BLE device can have to understand this technology's limits [48] [49]. In [43], three researchers performed an overview and evaluation of BLE performance based on four factors: energy consumption, latency, piconet size, and throughput. The security of BLE is tested by Mike Ryan [50], presenting different techniques to eavesdrop into BLE communications.

Energy Consumption

The following figure illustrates a slave's lifetime for one-way and round-trip approaches and different parameter configurations, based on CC2540 current measurements.

The slave was expected to consume less energy when the *connSlaveLatency* is at its maximum. The maximum value obtained was 14.1 and 12.4 years for the one-way and round-trip method, respectively, resulting from a *connInterval* of 86.25ms and *connSlaveLatency* of 370 (the master obtained readings every 32 seconds). The smallest value obtained had a

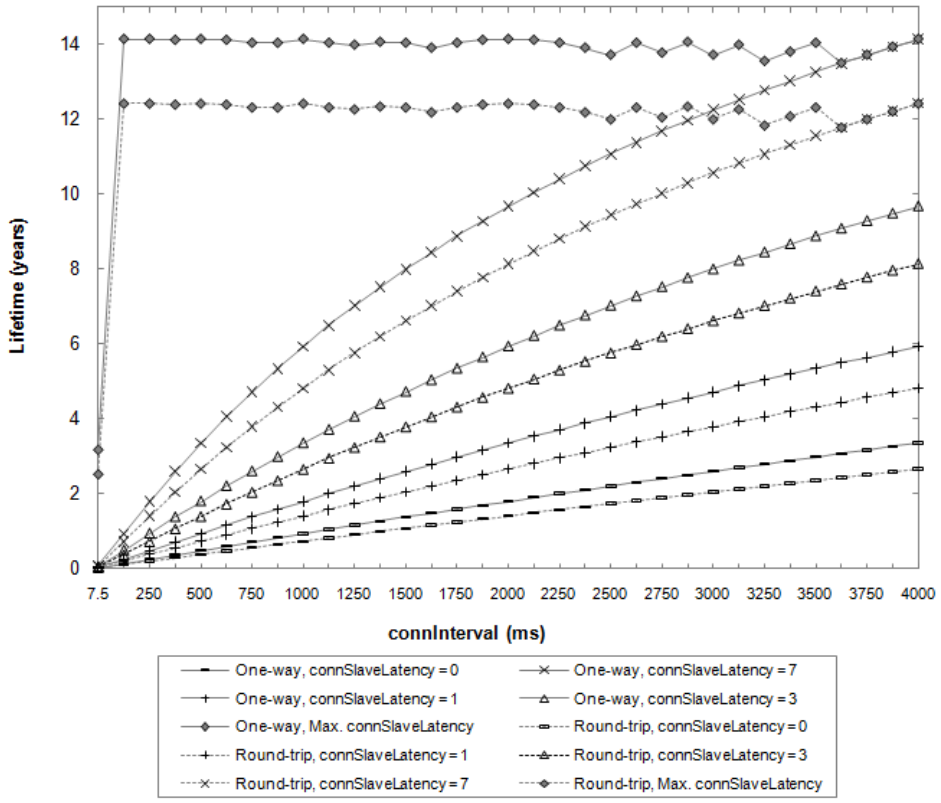


Figure 2.17: Theoretical lifetime of a slave for one-way and round-trip ATT message exchanges, and for different parameter configurations, based on CC2540 current measurements [43]

connInterval of 7.5ms and a *connSlaveLatency* of 0, with a duration of 2.6 and 2.0 days for each method. Note that the slave's lifetime, when the *connSlaveLatency* has the maximum value, does not differ with the increase in the *connSlaveLatency* due to the value of the *connInterval*, which cannot exceed 499.

Latency

The next figure illustrates the average latency for one-way and round-trip message exchanges, for various *connInterval* and BER values.

Latency was measured between the first message transmitted and the correct reception of the last message. The figure represents an average of more than ten million simulations. The average latency for smaller Bit Error Rate (BER) values for one-way and round-trip presents values smaller than 2ms and 1ms, respectively, for different *connInterval* values. For higher BER values, the value of *connInterval* influences latency levels. The correct data transmissions are slower due to the tendency to have more than one connection for a single transmission to be successful.

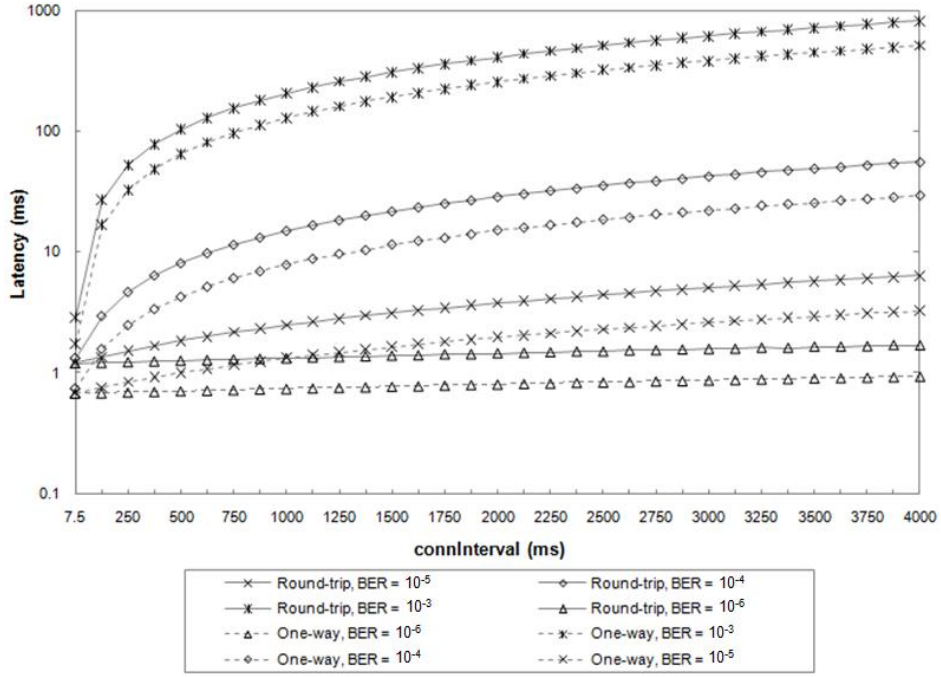


Figure 2.18: Average latency for one-way and round-trip message exchanges for various connInterval and BER values [43]

Security

In this article [50], Mike Ryan presents techniques for listening to BLE conversations, showing how packets can be picked up and reassembled in connection streams, and revealed an attack that makes encryption useless to the key exchange protocol, removing any confidentiality associated with the protocol.

2.5 Intelligent Transportation Systems and Vehicular Communications

One of the most important of the ITS and C-ITS is the Vehicle-to-Vehicle (V2V) communications. The transmission of messages between vehicles will have a more efficient role in road safety, rather than vehicles working individually for this purpose. Exchanging vehicle data can prevent, for example, chain collisions if the information about the first accident is being reported to the other nearest cars.

2.5.1 ITS

ITS allows more efficient use of society's infrastructure to improve traffic flow and road safety and reduce environmental impact. C-ITS focuses on the communication between those systems.

Over the years, several communication types have been suggested to support the information exchange within the vehicular network. Among these types are Global System for Mobile Communication and ZigBee [51], BLE [52] [53], and Visible Light Communications (VLC) [54]. However, the most used technology is based on Dedicated Short Range

Communications (DSRC), designed to support applications based on V2V and Vehicle-to-Infrastructure (V2I) communications. Other alternatives are being studied, such as LTE-Vehicular (LTE-V) or Fifth Generation (5G), and the 5G technology is considered the successor of the DSRC technology in vehicular communications [55] [56]. The expectation is that Millimeter Waves (mmWaves) is the basis of 5G, allowing a quick exchange of information from sensorization such as fine-grade object detection, real-time road video and high-resolution maps [57].

Dedicated Short Range Communication

In DSRC systems, there are two main stacks of protocols for vehicular networks: IEEE WAVE and ETSI ITS-G5. The first stack was initially presented in America and developed by IEEE and the second was developed by ETSI for Europe. Figure 2.19 illustrates both of them.

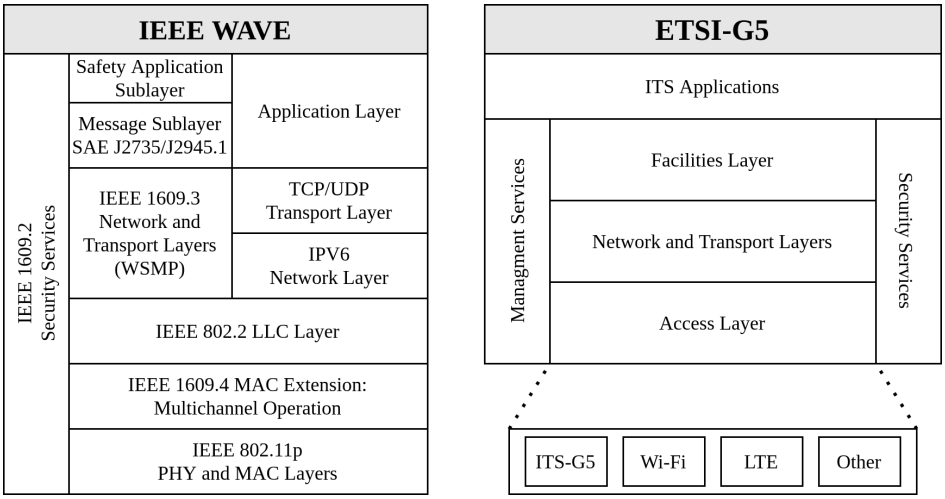


Figure 2.19: IEEE WAVE and ETSI ITS-G5 protocol stacks

These protocols depend on IEEE 802.11p to implement the physical layer and the Medium Access Control (MAC) layer. The physical layer uses OFDM with BPSK, QPSK, 16-QAM, and 64-QAM models, identical to the IEEE 802.11a standard, but with double-time parameters to obtain less interference due to the multi-path propagation and the Doppler shift effect.

In these protocols, the MAC layer presents some differences to the IEEE 802.11a standard, as it differs in some characteristics from the traditional Wi-Fi deployment.

2.5.2 ETSI ITS-G5

In the standard OSI model, the upper protocol layer is the application layer, which consists of all the applications in a system. However, in the ETSI protocol stack, the application layer was split in two: the upper part that continues to be called the application layer and the lower part called the application support layer (ITS Facilities).

As the name implies, this layer provides resources to the various ITS applications, such as information collection and support and authentication support. It is also responsible for managing messages that allow collaboration between vehicles - Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM).

The ETSI-G5 protocol defines these two types of messages that enable exchanging of events and beacons between ITS stations. CAM and DENM are the messages used in-vehicle communications that help, for example, the driver in preventing road accidents.

Cooperative Awareness Message

CAM are messages shared by each node in the ITS network that periodically provides information to neighboring nodes. Therefore, each ITS station maintains awareness of the environment surrounding it through status information such as time, position, motion state, and attribute information, including data of dimensions, type of vehicles, and function in the road traffic. The risk of collision can be estimated using the information that a nearby vehicle provides, comparing his data with its own, thus estimating this value and may even inform the driver of that vehicle [58].

The control of sending and receiving CAM messages is done by a CAM manager, located in the *ITS Facilities* layer. The generation of a message is created from the *time management* information, *station state monitoring*, and *mobile station dynamic* modules. In order to be transmitted to the network, the message is passed to the layer below. After the manager has validated the reception message,it passes to the Local Dynamic Map (LDM) management, responsible for maintaining a local georeferenced database with information around the vehicle.

Depending on the various *use cases*, there are different requirements in terms of the transmission timing.

Table 2.3: CAM Use Cases

Use case	**Min Frequency (Hz)**	**Min Latency (ms)**
Emergency Vehicle Warning	10	100
Slow Vehicle Indication	2	100
Intersection Collision Warning	10	100
Motorcycle Approaching Indication	2	100
Collision Risk Warning	10	100
Speed Limits Notification	1 to 10	100
Traffic Light Optimal Speed Advisory	2	100

The messages will have to be transmitted with a minimum interval of 0.1s (10Hz) and a maximum interval of 1s (1Hz) [59].

The CAM message format consists of a header and a body. The header contains information about the message, such as the message identifier, the generation time, and the version. The body contains information about the ITS station, such as the station identifier and type, reference points (latitude, longitude, elevation, and heading), among other CAM parameters [60]. Figure 2.20 shows the complete message structure.

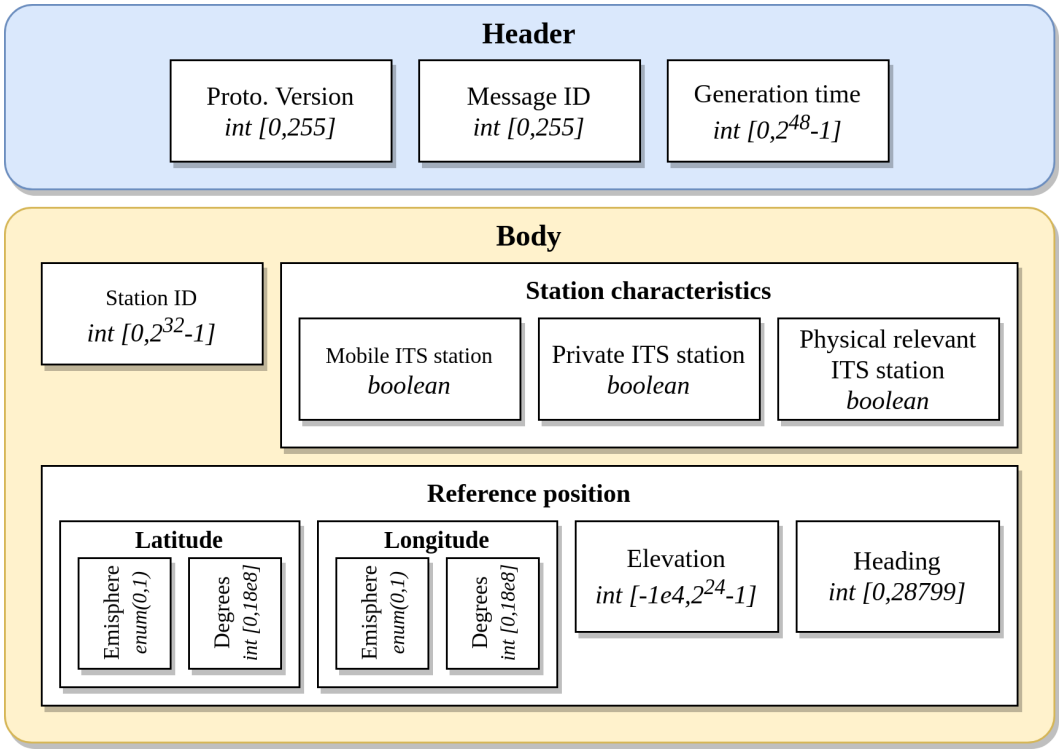


Figure 2.20: Structure of the CAM message

Decentralized Environmental Notification Message

Unlike CAM, DENM messages are only transmitted when the ITS station detects an event on the road. DENMs are used in Road Hazard Warning-related applications (RHW), and these applications are event-based and composed of multiple *use cases* that will define the final message structure. Table 2.4 show some of the *use cases*.

Table 2.4: DEMN Use Cases

Use case	Condition
Emergency Eletronic Break Light	Hard breaking of a vehicle
Traffic Condition Warning	Traffic Jam Detection
Collision Risk Warning	Detection of a collision risk by a RSU
Precipitation	Detection of a heavy rain or snow by a vehicle
Wind	Detection of a strong wind condition (stability control of the vehicle)
Visibility	Detection of a low visibility condition (activation of some lights or anti-fog)

As in the CAM messages, DENMs are generated by the message manager from the information provided by the *time management*, *station state monitoring* and *mobile station dynamic* modules and also from the LDM georeferenced database. Once the message is constructed, it moves to the lower layers [61].

The general procedure for dealing with these types of messages is as follows:

- The ITS station, as soon as it detects a situation corresponding to a known *use case*, starts by broadcasting the respective DENM message to the stations located within a relevant area;
- The message is repeated with a specific frequency and persists until the event is resolved;
- The DENM broadcast ends automatically after a particular time or when an ITS station reports that the event has ceased to exist;
- ITS stations that receive DENM process the data and choose the type of warning they show the user.

The format of the DENM message, as shown in figure 2.21, consists of a header and a body. The header is identical to the one of the CAM message, but the body contains more fields, organized into three different categories:

- **Decentralized Situation Management Group** - general information about the event;
- **Decentralized Situation** - details about the event informed;
- **Decentralized Situation Location** - inform about the event location data.

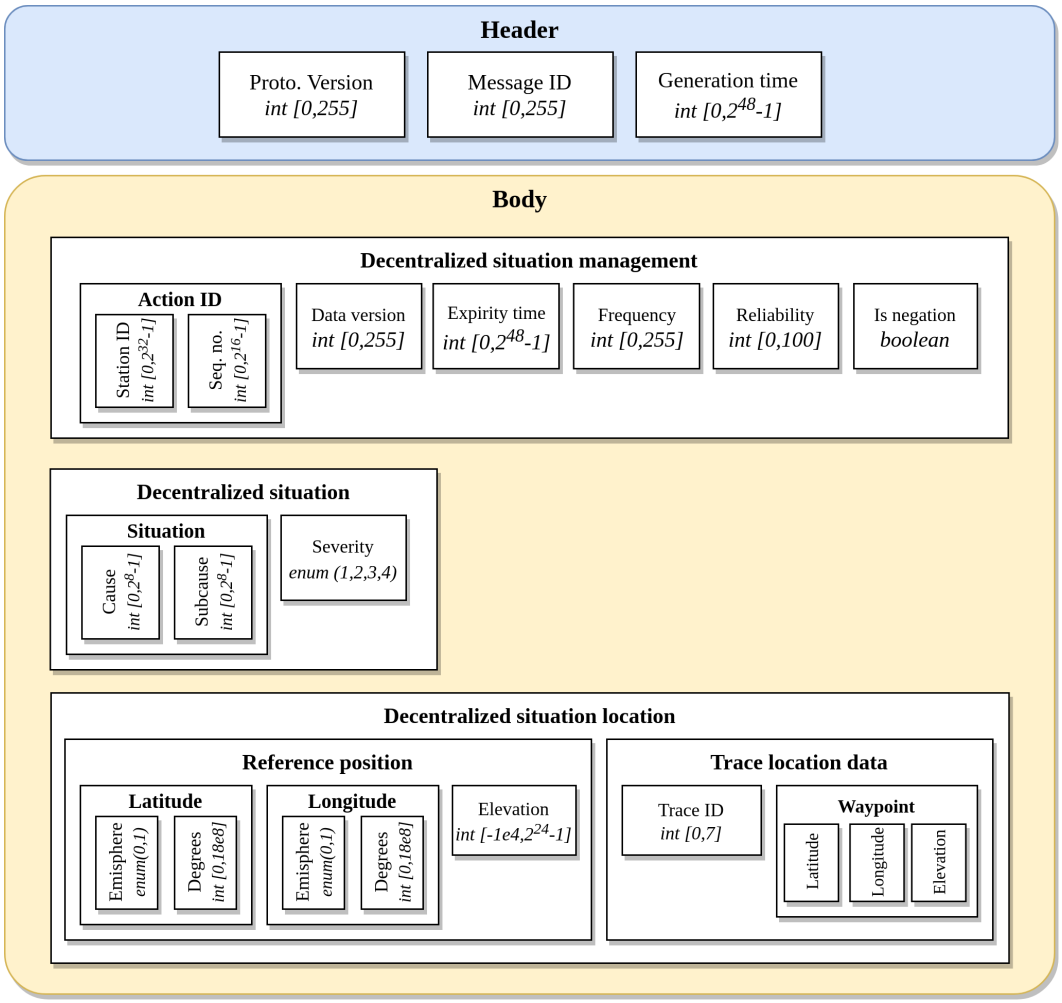


Figure 2.21: Structure of the DENM message

2.6 Review of the state-of-art of algorithms for CAN data processing

As already mentioned in section 2.1.5, the OBD-II diagnostic system consists of a *request-response* method with several services available, such as real-time vehicle status data. The SAE J1969 standard defines a list of Parameter IDentifications (PIDs) that must be the same in all vehicle models, allowing access to different types of parameters without knowing the vehicle's CAN specification. Acessing these type of parameters can be used to analyze the raw CAN traffic. The acquired data is used to correlate with the proprietary signals of each vehicle.

Recently, Hyun Min Song and his co-worker [62] presented a method for decoding these unknown messages comparing raw CAN messages and PIDs *response-messages* with a score method, discovering which proprietary signals are closest to the data of the PID signals. Thomas Huybrechts *et al.* [63] uses the same logic, but it is performed with two different approaches to analyze the data: a Machine Learning (ML) and a Arithmetic method (Root Mean Square).

Although these methods show promising results, they cannot identify parameters beyond the PID list. The other parameters, i.e., the headlight and windshield's status and the wheel's speed, does not appear in this list. Therefore, different methods are developed to decode the available raw CAN signals.

Marchetti *et al.* [64] (2018) proposed READ that consists of an algorithm to extract signals, analyzing all the bits of the CAN messages. The algorithm starts by counting the number of bit-flips occurring in consecutive messages, dividing by the total number of messages. Moreover, it uses a logarithm function to represent the different orders of magnitude of the bit-flip rates.

In the first phase, the magnitude value is used to detect the signal boundaries. The algorithm senses that the Least Significant Bit (LSB) of the signal will have more bit-flips than a Most Significant Bit (MSB) of the adjacent signal. Consequently, if the magnitude of the bit i is greater than the magnitude of bit $i+1$, the algorithm identifies a boundary between those bits.

In the last phase, the algorithm classifies the detected signals into three categories: counters, checksums, and physical signals. The counters are more easily detected because these signals increments in every sample. So, the bit-flip rate of LSB is always 1 and from the MSB to LSB, the bit-flip rate doubles to the value of 1. Checksums are detected when the magnitude of all the bits are 0, and the normal probability distribution of the bit-flip rates centers at 0.5.

After 25 traces analyzed, Marchetti *et al.* [64] reveal that READ is much more accurate at finding signal boundaries than Markowitz *et al.* algorithm FBCA [65].

In conclusion, READ is a great CAN reverse engineering algorithm but can only found the boundaries between signals. However, in the same year, a new algorithm that can simultaneously tokenize, translate and interpret CAN signals appears. ACTT [66] is a method that collects raw CAN data when the user is driving, identifies the position of a possible

physical signal, and correlates with the PIDs signals.

The algorithm starts by categorizing all bits into constants (0's and 1's) and non-constants. Afterward, the time-varying sequence of bit strings is converted into a sequence of integers to make a linear regression of the two types of signals (extracted signals and signals obtained by responses to PIDs) and apply a linear fit score. Lastly, an algorithm is applied to identify those not related signals, to improve the score. The interpretation of the physical signals is measured using the formulas of the related PID.

As in READ, ACTT only considers signals that are unsigned and with big-endian byte order. However, ACTT shows promising results, finding 69.6% of constants 0's and 1's and matching 16.8%, leaving 13.6% of the signals unidentified.

LibreCAN is presented by Pesé *et al.* (2019) [67] and consists of a three-phase algorithm:

Phase 0 is inspired by the READ algorithm, with a slight difference in identifying signal boundaries and classification of the signals. The signals are classified as CONST (constant), UNSED (unused), and POSS (possibly COUNTER, MULTI, CRC, or PHYS).

Phase 1 is identical to the ACTT translation. The signals obtained from phase 0 correlate with signals obtained from the PIDs diagnostic responses, using linear regression to discover the signals' scale and offset. Also, the researchers use external sensor-data to improve the results.

Phase 2 involves a new method that allows identifying body-related signals, such as windshields, doors, and lights over a filtering process, before and after acting on the parameter. The theory is to collect signals before turning on the car entirely as a reference. After saving some of these signals, the key is flipped and the signals are collected again. Three types of signals can be identified: CONSTANT (messages that change after turning on but still a constant), REFERENCE (that do not change after turning on, maintaining the reference state). and POWERTRAIN (which changes after the trigger and change their value over the time).

Once again, this algorithm works only for signals that assume big-endian byte order and unsigned encodings.

The research in this field has been growing. Many authors contributed to detecting and identifying proprietary CAN signals, decoding the messages that Original Equipment Manufacturers (OEMs) attempt to cover up. The year of 2020 had new papers/works coming out, presetting new improvements for this area.

Clinton Young *et al.* [68] present a method identical to phase two of LibreCAN, combining CAN IDs with the vehicle functions through changes in the data field. They use a clustering algorithm to associate the IDs based on their function, classifying these unknown IDs under the previous labels. Miki E. Verma *et al.* [69] criticized the submitted proposal, who claims not to consider "a true signal reverse engineering algorithm".

Miki E. Verma and his co-workers [69] introduce CAN-D, a four-step pipeline for decoding CAN data. Unlike all the other algorithms presented so far, it can extract and distinguish between big-endian and little-endian order, and signed and unsigned signals.

The first stage is the classification of the signals, which can be performed using two methods: heuristics or ML. The heuristics method is identical to the method used in phase zero of LibreCAN and achieves identical accuracy with the ML method. After training with many CAN databases, a supervised classification is performed with several classification methods to detect which method presents the best results. The *Random Forest* classifier achieved the best result in predicting the probability of a single bit been an LSB of a signal.

In phase 2, several procedures are created to optimize tokenization and discover the signal's byte order. Once the signals are tokenized, phase 3 starts to classify whether the signals are unsigned or signed.

Phase 3 starts similar to phase 1, experimenting with supervised classifiers to classify the type of signal. However, as the output of phase 2 already indicates the LSB and MSB of the signal, the detection of signedness is simpler to solve through a Heuristics Signedness Classifier, which verifies how the two MSBs would behave if the signal were unsigned or signed. After that, the CAN-D use a combination of phase two of LibreCAN and ACTT to interpret and discover the integer value of the signals in phase 4. CAN-D prove higher results than the rest of the algorithms, with a better performance in terms of the signal boundary classification and error rates.

Daniel Frassinelli *et. al.* [70] presents AutoCAN, a software to analyze privacy and data security. This tool also extracts unknown CAN data from the vehicles and tries to discover parameters by establishing relationships based on physics laws.

AutoCAN defines four signal categories: *uint*, *enum*, *rand*, and *cyclic*. The algorithm starts from bit 0 and increases the analyzed signal size until it identifies that the bit *i+1* is no longer part of the signal, indicating the start of a new signal. After that, the algorithm implements a plausibility analysis verifying if the signal had an equivalent behaviour to one of the four categories specified. The analyze divides into four parts:

- **Auto-correlation** - to verify if the signal is well segmented or has an error;
- **HAMD** - detects bit endianness and encoding size by measuring bit changes whenever the signal changes. The number of changes is normalized, getting a value between 0 and 1, where 1 means that the bit has always changed and 0 means that it has never changed. If it has a higher HAMD value from right to left, the order of the bytes is little-endian, otherwise, is in big-endian configuration;
- **Post-HAMD** - MSB validation of *uint* signals;
- **Removal of Signals Error** - removes signals where all bits change to 1 at the same time.

After the segmentation, the algorithm identifies the signals using the correlation between them. Notwithstanding, unlike the other algorithms that correlate the segmented signals with PID signals, AutoCAN applies physical and mathematical properties, relating all signals simultaneously, i.e., integrating vehicle's speed and the odometer data. The correlation is classified using Pearson's coefficient.

AutoCAN was tested in four different vehicles, resulting in a percentage of correct segmentation between 70% and 89% of the known signals and 6% to 11% of average bit-error, based on a ground-truth with several documents, OBD analysis and manual reverse engineering.

This algorithm demonstrates a better segmentation work than READ, where there is a distinction between the different physical signals (uint, enum, cyclic) which in READ is limited to only one category (PHY).

Daniel Frassinelli *et al.* [70] also explored the purpose and functionality of Telematic Control Unit (TCU) in modern cars and the device's security mechanism and how TCU extracts information from the user and the vehicle's routes without his consent. This non-consent further revealed that any car dealership might have all the travel data made by the car. The following table 2.5 summarizes all the properties described in this section of each work.

Table 2.5: Summary of CAN reverse engineering algorithms for each of the four properties

	Boundary	Endianness	Signedness	Interpretation
READ (2018) [64]	✓			
ACTT (2018) [66]	✓			✓
LibreCAN (2019) [67]	✓			✓
Young, C. et al. (2020) [68]				✓
CAN-D (2020) [69]	✓	✓	✓	✓
AutoCAN (2020) [70]	✓	✓	✓	✓

After reviewing the literature, it is proven that the studies show a significant improvement in the segmentation and identification of unknown CAN parameters. Consequently, OEMs are increasingly investing in security and message encryption to prevent attacks on the vehicle's system. However, all the algorithms described process the vehicles' data after several traces and with databases with millions of messages to decode them.

The work developed during the dissertation, described in detail in the following sections, shows a different approach from the works previously presented. The main difference stays in the identification of the weather-related parameters before or during a trip.

CHAPTER 3

System Architecture

In the development of this work, a system is created capable of reading, decoding and identifying various parameters in the CAN messages of the vehicle related to the road and weather conditions around it, from an external equipment connected to the respective CAN pins of the OBD-II port.

Chapter 2 presented a summary of the fundamental concepts for developing this work and a review of several algorithms proposed by researchers from various universities for the problem exposed by this dissertation. However, and as described at the end of the previous chapter, these algorithms have a disadvantage to this work: it only detects the parameters after collecting data from several traces and making extensive analysis of all messages extracted from them. Although the results of the presented algorithms are positive, they are not suitable for anyone's needs. The architecture used is sophisticated, and it is still very focused on research and not on a solution to fill a position in the market. The solution proposed for this work is more simplified, more economical and prepared to immediately act on the vehicle without any prior reading of the vehicle's CAN messages. The detection of these parameters improves the vehicles' traffic on the roads, preventing congestion and accidents related to weather conditions through cooperative communications such as V2V and V2I.

This chapter presents the system's global architecture where this work fits, followed by the detailed architecture about the proposed solution, describing the functionality and the function of the elements involved.

3.1 Global Architecture

The global architecture of an Intelligent Transport System (ITS) can be introduced by explaining SARWS. SARWS is a project that contributes to the evolution of these systems in terms of weather conditions.

Usually, the meteorological data of the area are collected through weather stations. SARWS proposes a new technique to gather and share this data using different units, such as vehicles, road-weather sensors and smartphones.

One of the advantages of SARWS is that all the vehicles can benefit from it, even if they do not have CAN readers,OBUs or Internet access. SARWS notifies drivers about any information related to the weather conditions. Furthermore, this project can evaluate the traffic pollution by tracking the air quality from the vehicles' sensors and reporting these to the public authorities.

Therefore, the principal objective is to provide weather services in real-time that ensure safety, efficiency, and energy sustainable mobility that cannot be seen in traditional approaches. That is where ITS play a part, contributing to human comfort and security, providing tangible results quickly and efficiently.

As mentioned in the first chapter, this work is part of the TRUST project, which belongs to a part of the SARWS project, and has several objectives such as:

- Obtain information about the climate from sensors available in vehicles and road infrastructures;
- Develop new sensors to evaluate the state of the environment and traffic conditions;
- Use of communications protocols over the vehicular network (V2V, V2I);
- Remote analysis and processing of all information to identify and generate warnings to drivers about the conditions and the most critical areas to drive;
- Dissemination of road safety alerts almost in real-time through vehicle communication networks of central servers, but also of neighbouring nodes in more urgent situations;

Figure 3.1 and Table 3.1 illustrate and details the architecture workflow, focusing on the high-level connectivity and the functionalities layers.

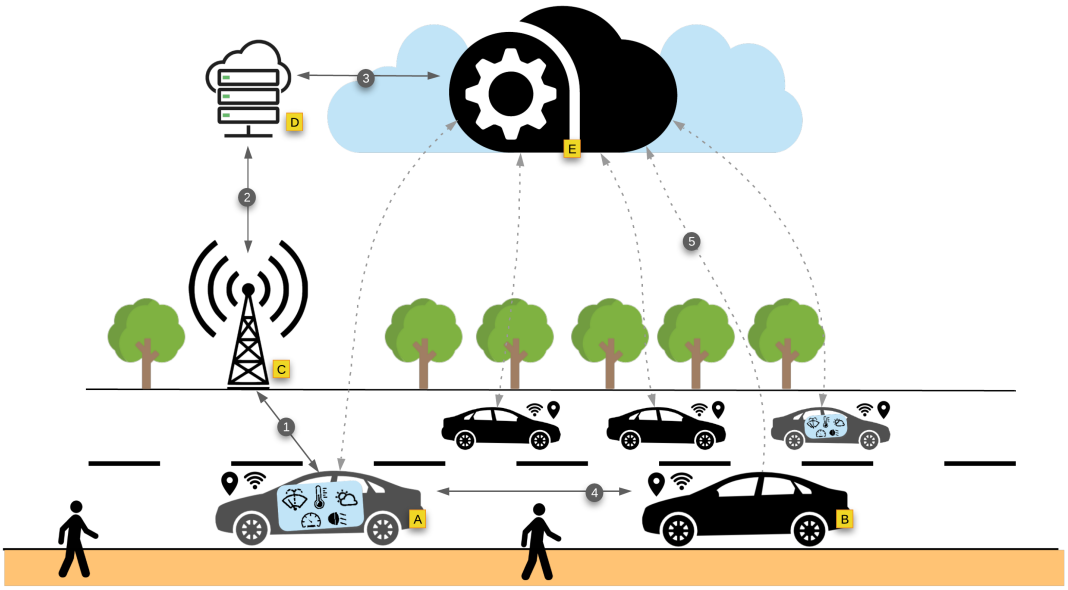


Figure 3.1: Scheme of the global architecture workflow

Table 3.1: SARWS architecture components and description

Components	Description
A	Weather-Pollution V-ITS-S station (WV-ITS-S)
B	Vehicle ITS station (V-ITS-S)
C	Roadside ITS station (R-ITS-S)
D	Central ITS station (C-ITS-S), Traffic Management Center (TMC)
E	Broker, Cloud
1	Vehicle-to-Infrastructure communication (V2I)
2	C-ITS-S to R-ITS-S
3	Broker to C-ITS-S
4	Vehicle-to-Vehicle communication (V2V)
5	V-ITS-S to cloud

The present architecture has four functionality layers. The first one is regarding the *Data Acquisition Layer*, which consists of all the sensors that notify about weather/pollution and traffic information. In this case, the data (windscreen wiper movement, the surrounding temperature and headlights status) from the sensors are extracted from the vehicles through CAN-BUS.

The second layer is the *Communication Layer*, which lists and uses all communication types (1 to 5), as shown in Table 3.1, assuring the inter-station connection and providing the required safety and weather/pollution contents to both Central ITS-S and the broker in the cloud (D, E).

The third layer is the *Management Layer* responsible for handling, storing, and analysing the data obtained from the installed ITS stations. This management process is restricted to the central ITS station (C-ITS-S) with limited interaction with the cloud broker.

The fourth layer is the *Data Diffusion Layer*, where the information collection and distribution occurs. Furthermore, this layer is responsible for diffuse the handled messages, generate high-level data, and interrelate with the end-users and brokers. Additionally, in this layer, different high-level algorithms are typically controlled by the security manager's entities.

3.2 Proposed Solution

As already mentioned, this works aims to develop a system to read and identify messages from the CAN in-vehicle network. A few elements with particular technologies are needed to implement this system to achieve this goal.

Figure 3.2 presents the developed architecture scheme, in which three main components are distinguished.

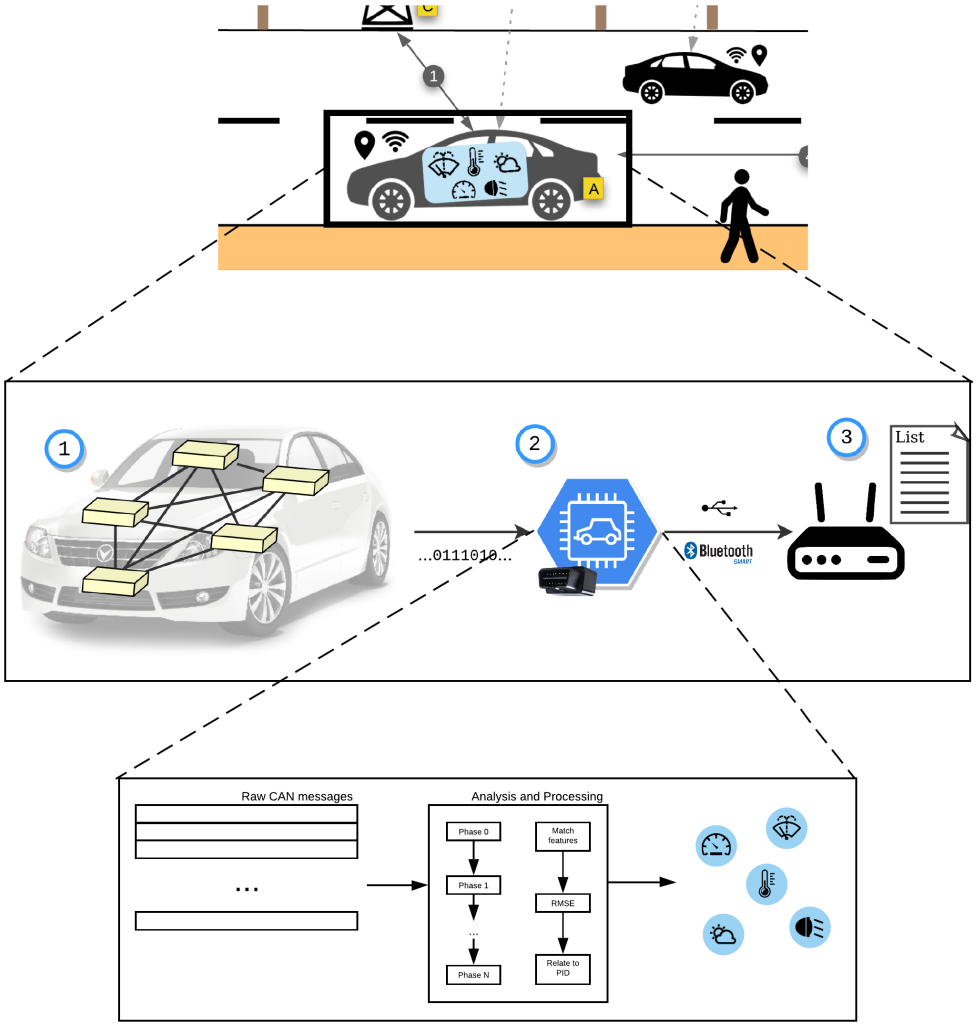


Figure 3.2: Proposed system architecture. In 1 the vehicle is represented, in 2 the implemented algorithm and in 3 the IT2S platform

The previous figure shows the three elements that belong to this system: the vehicle, the algorithm and the IT2S platform.

Vehicle: Represented by 1 in figure 3.2, this element consists of a CAN network responsible for monitoring and controlling the various parts of the vehicle through control units and sensors installed inside of them. There is also an output port, the OBD-II port, which also allows to monitoring the data being communicated between the units. Over this OBD-II port, it is also possible to communicate into the car, so the OBD-II is used to collect this data.

CAN-Reader & Algorithm: Represented by 2 in figure 3.2, this element consists of a controller and a corresponding algorithm to detect and identify the raw CAN messages. The communication between the OBD-II port and the controller is not direct because, between these two elements, it needs a CAN transceiver to enable the communication between the

network nodes and the micro-controller. Sometimes it may also be necessary to use a CAN controller, if the micro-controller does not have that capability.

Figure 3.2 also shows the two protocols used for communication between the controller and the IT2S platform. It is understood that the controller must also contain these two forms of communication, BLE and Universal Serial Bus (USB). The proposed algorithm is described in the next chapter.

IT2S platform: Represented by 3 in figure 3.2, this element is built on top of the apu3c4 system board from PC Engines with several features and a set of extensions such as a GPS receiver, a WiFi module, an LTE module, and a pair of appropriate antennas for each module. This platform can operate either as an RSU or as an OBU. For this architecture, the platform works as an OBU, receiving the various parameters via USB or BLE. It is prepared to transmit this data to an outside RSU with cooperative messages [71]. Figure 3.3 shows a picture of the platform.

Figure 3.3: IT2S platform

Figure 3.4 depicts the on-board devices that will be installed inside the vehicle.

In the following figure, in addition to the elements described above, a smartphone is also present. Even if it is not directly related to this work, the smartphone has an essential role in the TRUST project. It is used to show a dashboard with different pieces of information and gathers light measurements and camera images (besides its core function of reporting/disseminating road-weather warnings).

Figure 3.4: Equipment inside the vehicle

CHAPTER 4

Implementation

The third chapter exposed, in detail, the proposed architecture of this work. The parameters collected by the On-Board Diagnostics (OBD) port have a fundamental role for the On-Board Unit (OBU), transmitting essential data about the vehicle's status and performance. This chapter explains the practical implementation with the concepts that were exposed previously in chapter 2.

First, it is described the materials used for the development of this work and, in the following sections, it is explained the implementation of the algorithm used to extract the weather-related parameters.

4.1 Main components

This section presents the main components that were used to implement this work.

4.1.1 CAN-BUS OBD Simulator

The simulator is an OBD development board to simulate the functionality of the OBD system. This platform can simulate up to three ECUs (ECM, TCM and ABS ECU), generate Diagnostic Trouble Codes (DTCs), and control five parameters in real-time, using five potentiometers [72]. Figure 4.1 shows the simulator.

The controller used on this board is the OE91C1610 produced by OZEN Elektronic. This controller has the following main features:

- Compatible with ISO 15765-4;
- Multiple variable and fixed PIDs;
- MIL Led output;
- Supports 11-bit and 29-bit identifier;
- Supports communications with 250 or 500k baud-rate [73].

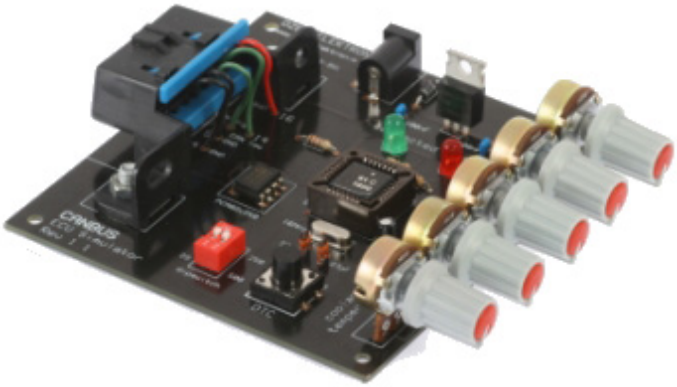
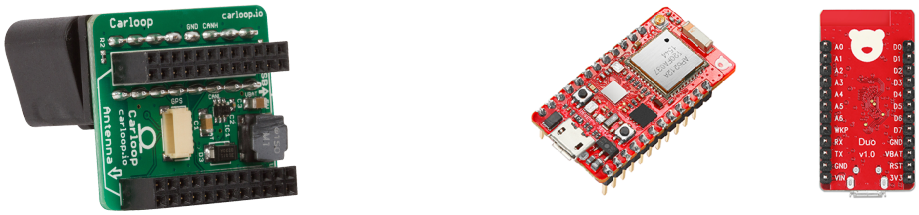


Figure 4.1: CAN-BUS OBD Simulator

4.1.2 Carloop

Carloop is a development kit to interact with the vehicle over the OBD port and link up to a cloud via 3G, Wi-Fi, and Bluetooth [74].

The Carloop (figure 4.2a) used is constitute with a RedBear Duo (figure 4.2b) [75], a development board created to simplify the process of building IoT products, developing projects with Wi-Fi and BLE communications, making it a useful tool for this work. Duo uses STM32F205, an Advanced RISC Machines (ARM) micro-controller with a CAN interface, so Carloop only needs a CAN transceiver, avoiding implementing a CAN controller.



(a) Carloop

(b) RedBear Duo

Figure 4.2: Carloop and RedBear Duo

4.2 Collecting data from the ECU simulator

In the early stage of this work, the simulator provides CAN data that can be used to choose the best protocol to communicate between the Carloop and the OBU before working with the vehicles.

The principal purpose for this part is to establish a communication between the Carloop and the IT2S platform with these two types: USB and BLE, using the CAN-BUS Simulator messages. Furthermore, it is appropriated to test and validate the capability of the Carloop on the exchange of CAN messages with the CAN-BUS Simulator. The parameters chosen to extract values from the simulator are associated with the potentiometers to change their values in real-time. The following table 4.1 present the name of the parameters and the respective PIDs.

Table 4.1: List of PIDs extracted from the CAN-BUS Simulator

PID (HEX)	Description	Var. Value
05	ECT	0..255
0C	RPM	0..65535
0D	SPEED	0..255
10	MAF	0..65535
14	O2 volt	0..255

For each parameter, Carloop gives 160 milliseconds to guarantee that the response is successfully received. Since the number of parameters is five, the transmission to the OBU is set to 200 milliseconds, allowing all the values to be updated with 1 second. During the 160 milliseconds, Carloop needs to send a *request-message* with the corresponded ID for external requests, the type of service that intends to read (mode 1) and the respective PID parameter. After that, waits until the CAN-BUS Simulator sends the *response-message*. Figure 4.3 shows the explanation of the process implemented in the Carloop.

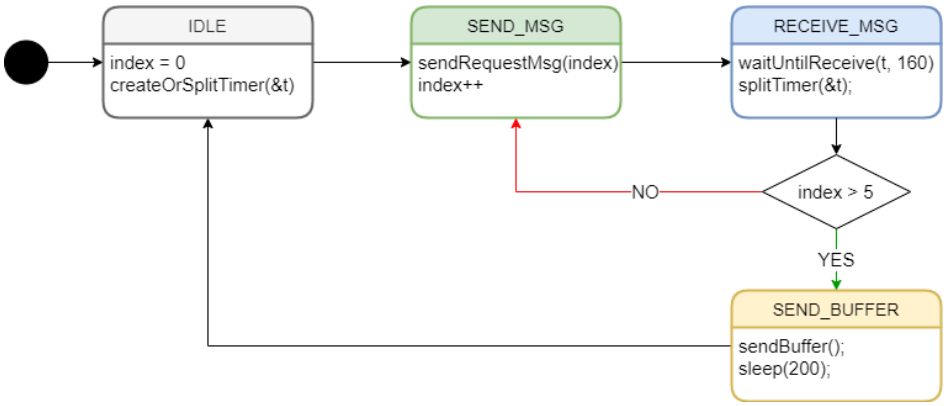


Figure 4.3: Representation of the state machine used to extract the parameters from CAN Simulator

After storing all the parameters in a buffer, the data is sent in a specific frame to the OBU with two bytes reserved for each parameter, separated with a comma.

For the sake of simplicity, the data are sent directly in hexadecimal to the IT2S platform, and the conversion is made on the platform side. Thus, all data processing is placed on this site, leaving Carloop focused only on extracting and transmitting data through the OBD port.

The simplest and fastest way to send is via USB. The RedBear Duo micro-controller has a native USB port, enabling it to dump data directly into the OBU using a USB cable.

However, this solution is not practical, so it is necessary to implement wireless communication between the two devices. The best solution is to use Bluetooth communication, in this case, BLE, because it is more connected to this type of IoT implementations. In BLE communications, it is necessary to define the roles and characteristics that each device performs in the Bluetooth network. It is mandatory to have at least one *Peripheral* device and a *Central* device. Based on the explanation given in section 2.4.1, it is decided to use the OBU as a *Central* to establishing the connection with Carloop that acts as a *Peripheral*.

In the configuration of the GAP level, it is also necessary to decide the devices 'role at the GATT level. Usually, the GATT server contains the data to be sent, and the GATT client only reads then. Since Carloop is the device that accesses the vehicle's data, it is assumed that it will have the server role.

The tecnique for sending data is identical to the one used for USB communication.

4.3 Collecting data from vehicles

The second part of this work consists of extracting proprietary weather-related parameters from the vehicles. The parameters required for the project are shown in table 4.2.

Table 4.2: Required weather-related parameters

Barometric Pressure	External Air Temperature
Brake Status and Boost	Vehicle and Engine Speed
Steering Wheel Angle	Steering Wheel Direction
Windshield Wiper State	Headlight Status
Engine Load	Wheels speed

All brands develop a different way of representing the content in CAN messages. In section 2.6, it is found in articles that the data may be in signed or unsigned and little-endian or big-endian configurations, plus the conversion to an integer, which may be different because of the scale and offset, increasing the difficulty to extract and interpret the data. However, this difficulty is more significant for the cases intended to identify parameters based on numerous messages captured during a trip, which led to thinking of a different way to acquire the desired parameters more easily.

The work developed is an iterative algorithm based on particular works presented in the state-of-the-art and other new concepts proposed to give more value to this work.

The selected parameters do not have the same size. The status parameters are only represented by a bit, unlike other parameters that can take values between one byte and two bytes. The analysis and the identification process vary depending on the parameters' size and behavior, and the proposed algorithm does not process the data equally for everyone. For that purpose, the algorithm's operation is divided into two categories: **binary parameters** and **non-binary parameters**.

4.3.1 Binary parameters

The identification of the binary parameters, such as the headlight status, windshield wiper state, and brake status, consists in three phases: **Reference**, **Monitoring**, and **Validation**. In each phase, the algorithm checks all the messages sent to the bus and counts all the transitions in each bit (bit-flips), if applicable. Figure 4.4 shows how bit-flips are counted. The data processing is then treated differently between each phase.

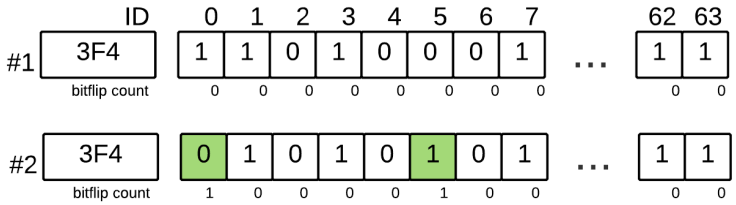


Figure 4.4: Demonstration of the counting process of bit-flips

Phase 0: Reference

In this phase, the user needs to turn ON the vehicle and not perform any action inside the vehicle for a bit of time. The algorithm will search for the bits that did not show any transition (0 to 1 or 1 to 0), saving the reference for the next phase. The rest of the bit-flip numbers greater than 0 are ignored in the next phase.

Phase 1: Monitoring

In phase 1, the user will have an order to make a specific number of transitions in the parameter he wants to identify. The algorithm then filters the bits with a bit-flip count twice the number of triggers and deletes those that present different results. For example, if the driver is required to press the brake pedal three times, the brake pedal is pressed and released (x3). This case means that the message containing the bit that represents this parameter will vary, in that position, from 0 to 1 when it is pressed and from 1 to 0 when it is not, so it has to be considered twice the number requested. The filtered parameters are the only ones that the algorithm will be aware of in the next phase.

Phase 2: Validation

The last phase serves as confirmation to ensure that identification is made correctly. The user makes a different number of transitions requested by the algorithm, finding the desired bit.

The flowchart presented in figure 4.5 demonstrates how the described process works.

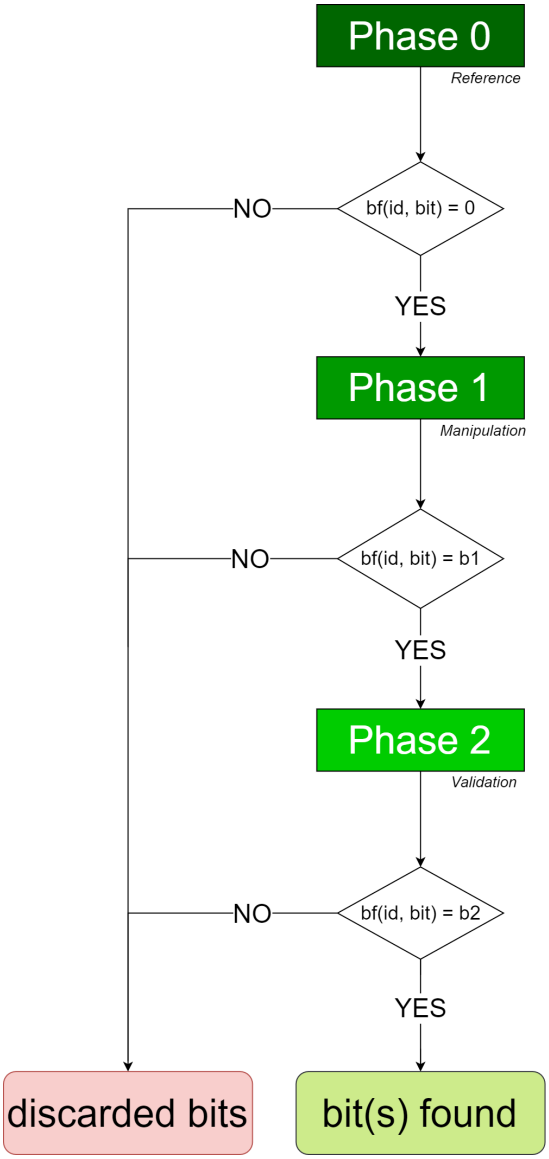


Figure 4.5: Flowchart of the method used to identify the binary parameters

4.3.2 Non-Binary parameters

The identification of non-binary parameters can be divided into two processes, depending on the type of parameter. This division occurs because some parameters listed in table 4.2 cannot be identified with the car stopped. Consequently, a new process had to be created to perform this part of the work.

The first process includes the *Steering Wheel Angle* and the *Throttle Pedal*. It is identical to the previous method, except that only the phase 0 and phase 1 are performed (Reference and Monitoring). There is no need for the confirmation phase because the parameters are already constituted by more bits, having a more particular behaviour.

Phase 0: Reference

The reference phase in identifying these parameters is more effective than in the identification of the previous parameters. In this process, the algorithm is not scanning the messages bit-by-bit, but byte-by-byte, so the number of candidates reduces. Any transition of a bit in a determined byte excludes the candidate, contrary to the analysis in the binary parameters in which only that bit is excluded.

Phase 1: Monitoring

After a particular time, the second phase begins with the driver being subjected to several instructions to execute inside the car. The type of instructions depends on the type of parameter to be identified. However, the processing and analysis of the data are identical. As the controller detects the messages via the OBD-II port, the driver carries out the instructions, and the algorithm records the UPs and DOWNs of the respective bytes that are being processed. If the byte X of the ID Y in the message N increases concerning byte X of the previous message (N-1), the algorithm increment UPs value. The same process occurs if there is a decrease concerning the previous byte. The time for this processing must be short, as the instructions are also quick to execute. These counters permit us to identify if the parameter presents a positive or a negative derivative quickly. After this count, the data is processed and filters the bytes with a counter 90% greater than the opposite counter. There may be parameters that require this phase to be repeated, and the analysis differs between them. The following figures 4.6 and 4.7 shows the processes to decode the mentioned parameters.

Figure 4.6: Illustration of the process used to find the throttle parameter

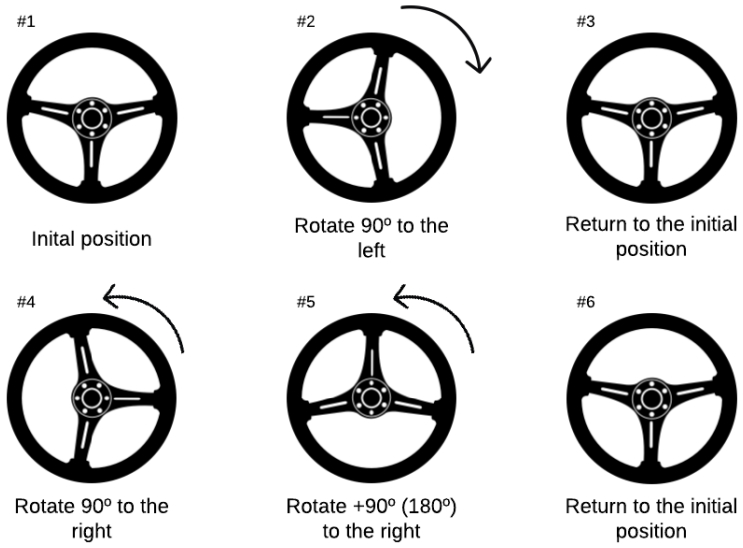


Figure 4.7: Illustration of the process used to find the steering wheel parameter

The second process includes the vehicle speed, the wheels' speed, and the engine speed (RPM). These parameters can only be identified when the vehicle is in motion.Therefore, the identification method cannot be the same as the process previously described, as there is no way to track the parameters with the car stopped.

Regarding the vehicle speed, the detection can follow two paths:

- Read the speed value through an exchange of messages with the ECU, sending a request to the controller with the respective PID and recording the response value directly in the buffer, as explained in the previous section with the simulator;
- Read the PID value and relate with all bytes of CAN messages, assigning a score to each byte - the byte(s) with the highest score are the candidates most likely to represent the speed of the car. This method is used in other parameters, and that is why it is explained later.

The first method is the simplest and most accurate way to obtain the vehicle speed. However, the second method can have the speed data without wasting time on the requesting process.

Regarding the wheels' speed, the second path is the only option for identification. During this work, other ways to identify these parameters, i.e., finding an ID with four values of two very identical bytes or finding these four equal values in different IDs. However, due to the microcontroller's processing and memory limitations of the micro-controller used, it is impossible to save that bunch of data and analyze it. A new solution had to be found that is fast and occupies the smallest possible memory space.

The inspiration to develop a new technique to relate CAN data with the ground-truth values of the PID speed comes from previous works [62] [63] described in section 2.6. During this part of the work, various methods have been attempted to relate these two variables.

Pearson's correlation coefficient between CAN data

The first method used to verify the relation is Pearson's correlation coefficient. Pearson's coefficient measures the relationship between two groups of data and can take values between -1 and 1. When the result is close to these boundaries, the correlation is strong, and when it is close to 0, the correlation is weak. Given a pair of random variables (x, y), the formula for the ρ is:

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y} \tag{4.1}$$

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 (y_i - \overline{y})^2}} \tag{4.2}$$

However, the raw CAN messages do not have the same sampling frequency, making it difficult to calculate the coefficient. The data can be interpolated to achieve the same number of samples [66] [67] [69], but this device does not have the memory to store all these messages.

```
id: 217; n: 2267
id: 211; n: 1966
id: 214; n: 1394
id:  90; n: 3791
id:  c6; n: 2842
id: 354; n: 291
id: 1f6; n: 5094
```

Figure 4.8: IDs and the corresponding number of messages

Pearson's correlation coefficient between bit-flips

After discarding the previous method, it is tried to implement the Pearson's correlation coefficient between the 8-bit bit-flips of two data groups. Now, the problem with the number of samples does not exist. However, after validating this method, the results show a low efficiency due to the strong correlation with almost all bytes. This strong correlation occurs because, usually, the number of bit-flips increases from the MSB to the LSB, resulting in a positive linear relationship. Figure 4.9 shows an example where this happens.

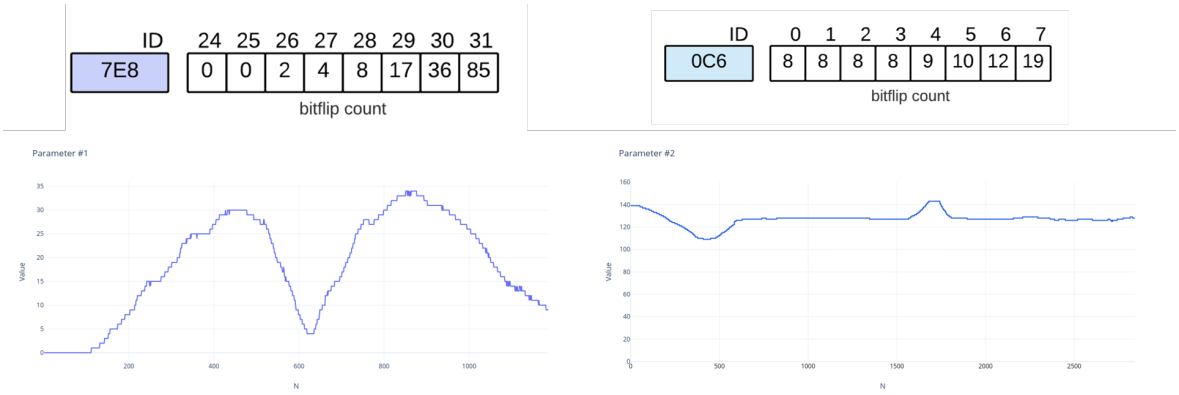


Figure 4.9: Comparison between the bit-flips and behaviour of two bytes

In this example, on the right, it represents the behavior of byte #3 of ID 7E8 and the bit-flips counters of that byte and, on the left, the behavior of the byte 0 of ID 0C6 and the respective bit-flips counters of that byte are represented. As can be seen in the figure above, there is no relationship between the graphics. However, it can be observed that the result between the ratio of the bit-flips values indicates a solid relationship between them.

$$\rho = \frac{(0-19)*(8-10,25)+...+(85-19)*(19-10,25))}{\sqrt{(0-19)^2*(8-10,25)^2+...+(85-19)^2*(19-10,25)^2}} = 0,997 \tag{4.3}$$

Implemented method

After the results were not promising in the previous methods, a new solution was chosen, based on our best knowledge. The solution found for this second process is a comparison between some features that classify the signals over time, without depending on the number of samples. The four features are average normalized, percentage of ascents, percentage of descents and position of the function's maximum value. The following table shows the explanation and the importance of each feature.

Table 4.3: Formula and description of the implemented method's features

Feature	Formula	Description
Average normalized	$\frac{average(x)-x(min)}{x(max)-x(min)}$	The sum of the values divided by the number of messages. In this case, it is used a min-max normalization to normalize the average between 0 and 1;
Percentage of ascents	$\frac{\text{number of ascents}}{\text{number of messages}}$	Estimate the derivative of the function described by the vehicle parameter during the trace
Percentage of descents	$\frac{\text{number of descents}}{\text{number of messages}}$	Estimate the derivative of the function described by the vehicle parameter during the trace
Normalized position of the maximum value	$\frac{index(max)}{\text{number of messages}}$	Find the position of the maximum number. This feature does not depend on the number of samples of the signal. However, when the sample numbers are too far, their comparison may not be accurate.

After all, a Root-Mean-Square Error (RMSE) is used to measure the differences between the features of the parameters. The formula for calculating the RMSE is given by:

$$RMSE = \sqrt{(\frac{1}{N})\sum_{i=1}^{N}(y_i - x_i)^2}, \tag{4.4}$$

where N is the number of features, yi is the predicted value (i.e. PID speed) and xi is the actual value (candidate).

The lower the RMSE value, the lower the distance between them. So, the probability of a given candidate is higher to be chosen.

CHAPTER 5

Tests and Validation

The previous chapter exposes the processes used to implement this work. After searching for methods to accomplish this dissertation's ambitions, it is time to test and validate them.

This chapter splits into two important topics for this work: the communication and the extraction of the parameters. Firstly, the two types of communication presented in 4.2 are tested with the CAN BUS ECU Simulator, following by the tests to the processes described in 4.3 on both vehicles. The results of the various methods will present figures and graphics to prove the veracity of the solutions.

In the end, it is considered a brief discussion about the algorithm's effectiveness and performance, based on the results of the tests.

5.1 Communication

Before beginning the tests to discover the CAN parameters, the validation of the communication between the Carloop and OBU is presented.

The previous chapter describes two methods to transmit these data: USB communication and BLE communication. This first part of the work demonstrates how Carloop can communicate the parameters with both standards.

The tests was performed at the Institute of Telecommunications in Aveiro. CAN messages are simulated through the simulator presented in 4.1.1. A PC and a Serial Bluetooth Terminal application for Android are used to validate data transmissions via USB and BLE, respectively.

5.1.1 USB communication

The first test proves that the two elements can communicate using a USB cable. The Carloop send request-messages and receives the respective response-messages from the simulator. The CAN parameters are broadcast to the IT2S platform after collecting all the values.

As shown in figure 4.3, Carloop requests the five parameters of table 4.2 and saves them in a buffer. When the last PID parameter is stored, the buffer is sent to the platform. This

platform has and embedded software to receive this data through a USB port, convert the values from HEX to Integer, and store it in a share-memory.

Figure 5.1 shows a picture with the setup used for this test.

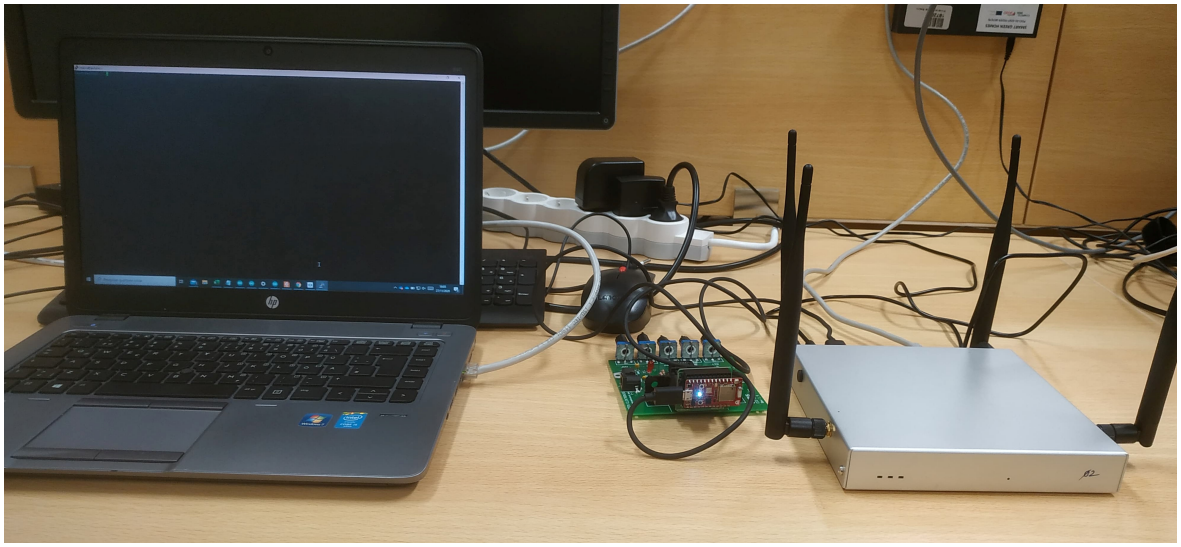


Figure 5.1: Setup used for the USB communication test

The IT2S platform receives data, and the integer values of the parameters are shown in an output terminal. Putty was the terminal chosen since it supports SSH protocol. Figure 5.2 shows the output terminal with the respective parameter values. The name of each parameter is abbreviated:

- *rpm* – Engine Speed (0x0C);
- *spd* – Vehicle Speed (0x0D);
- *ect* – Engine Coolant Temperature (0x05);
- *maf* – Mass Air Flow Sensor (0x10);
- *o_2* – Oxygen Sensor 1 (0x14);

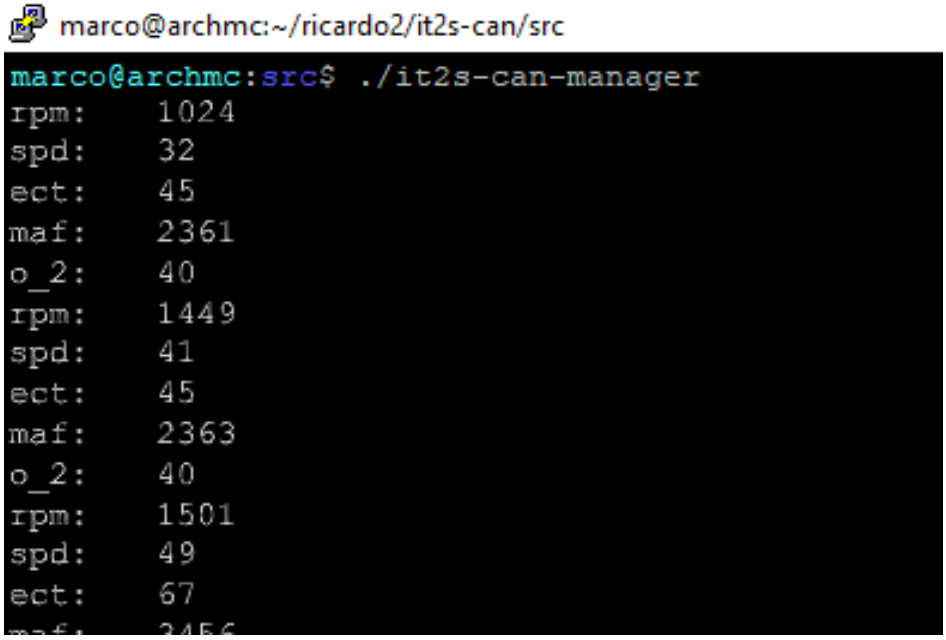


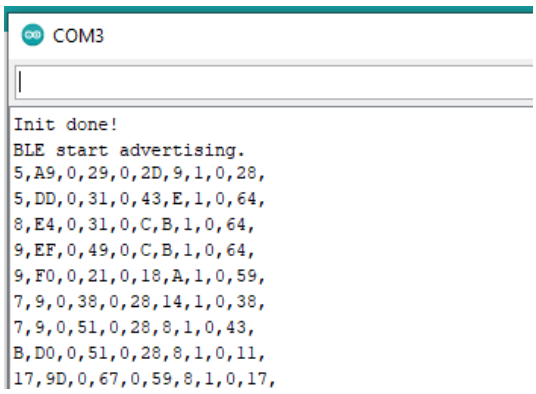
Figure 5.2: Data trasmitted from Carloop to platform via USB

After this test, the USB communication is ensured and proves that Carloop can transmit the CAN-bus data to the platform with this approach. The next step is to test the wireless communication, more precisely with the BLE.

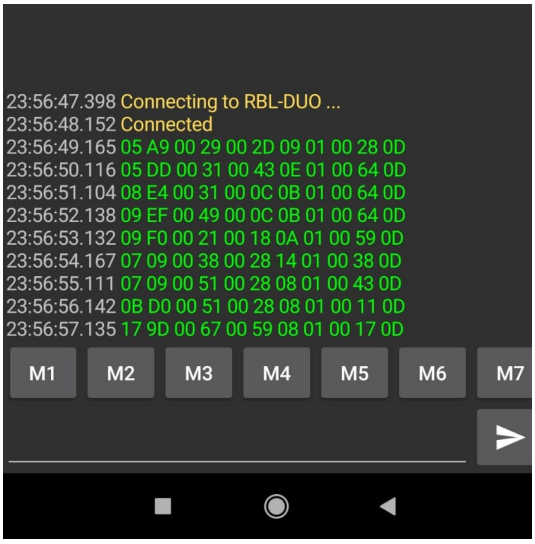
5.1.2 BLE

BLE communication implies a different setup. Due to the lack of time, it was only possible to develop the *peripheral* code to the Carloop, dismissing the Bluetooth part of the platform. For that reason, the setup includes a smartphone with a Bluetooth-Serial application to confirm the received data.

This application provides a simulation of a terminal to receive messages from every Bluetooth device that connects to the phone. The smartphone can detect which devices are available to pair and, as soon as Carloop is selected, the connection is complete, and it starts receiving the data, second by second.



(a) Debug of the transmitted data in the Arduino terminal



(b) Data on the Serial Bluetooth Terminal application

Figure 5.3: Result of the transmission by BLE

5.2 Data extraction from vehicles

These tests are executed in the vehicles and validate the algorithm's efficiency and the several processes proposed in section 4.3.

The present section presents the same structure as section 4.3 of the previous chapter. Firstly, the algorithm's performance in the detection of binary parameters is evaluated, followed by the non-binary parameters, and finishing with the presentation of two different routes to validate the identification of the position of the wheels.

In each part, the results are shown with a table containing the message ID and the respective position of the parameter in the data field identified by the algorithm. Since, in some cases, there is an iteration with the driver, graphics are also shown to validate that, in the position found by the algorithm, the iterations that the driver executes corresponds with the variations of the graphic. Other relevant figures are also shown for the demonstration of the test results.

5.2.1 Vehicles

The next two pictures in figure 5.4 show the vehicles used to test the methods implemented in this work: Nissan Micra 2012 and Renault Captur 2015.

(a) Nissan Micra 2012

(b) Renault Captur 2015

Figure 5.4: Pictures of the vehicles used to test the implementation methods

5.2.2 Binary parameters

Binary parameters may also be referred to as status parameters. These binary values are only intended to indicate whether the parameter is ON or OFF. When a parameter is in OFF, it is at '0' and when it is in ON state it is at '1'.

Identifying this type of parameters does not require any specific scenario since the tests are carried out with the vehicle stopped, only with the ignition triggered. These tests can be called *static*, because they do not need to set the vehicle in motion to extract the respective parameter.

Inside the vehicle, Carloop was installed, in the OBD-II port, with the detection software and a Computer to read the algorithm's requests and receive the output with the position of the parameter. Putty is used to reading the messages through the COM port where Carloop is connected to the PC. Due to the limited availability of the vehicles, the tests are performed during the night. Therefore the next figures, 5.5 and 5.6, may not be very perceptible with the arrangement of each vehicle's elements.

Figure 5.5: Setup with Carloop and PC in Nissan Micra 2012

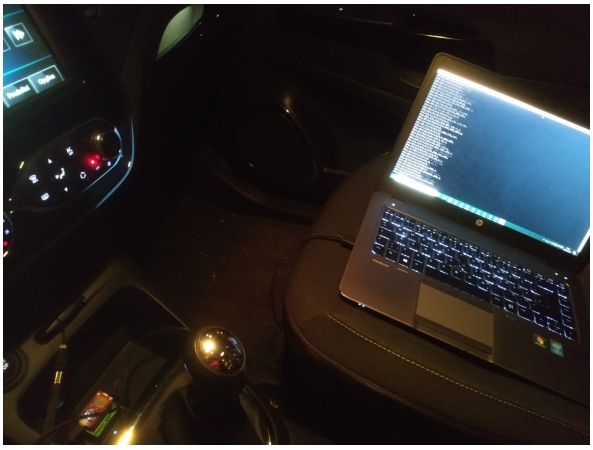


Figure 5.6: Setup with Carloop and PC in Renault Captur 2015

Figure 5.5 shows how Carloop interacts with the driver to identify the CAN message with the position lights' bit. After a set of messages, the algorithm asks the driver not to handle the lights control for 10 seconds, starting phase 0 of this process. The driver is then asked to turn the lights five times during 15 seconds, corresponding to phase 1. Three more transitions are requested for the validation phase (phase 2). In the end, the position of the parameter that the algorithm has identified is shown.

The result's veracity is shown in the graphic of figure 5.6, which shows the three phases and the requested variations within the required time spaces.

```
 --- NISSAN MICRA 2012 ---
 --- HEADLIGHT STATUS TEST: POSITION LIGHTS ---
 --- Starting Phase 0 in 3 ... 2 ... 1 ... (10s)

Turn on and off the lights for 5 times
Starting Phase 1 ... (15s)


Turn on and off the lights for 3 times
Phase 2 starting ... (15s)
id: 625 - bit#9
```

Figure 5.7: Identification of the position through the process of the three phases

Figure 5.8: Graphic with the transitions of the position lights in bit #9 of the CAN message with ID 625 over the time of the three phases

The other results obtained by the algorithm in this type of parameters are shown in the table 5.1. The figures from 5.9 to 5.13 show graphics of the parameters found to validate the position variations.

Table 5.1: Results of the binary parameters extracted in both vehicles

Parameters	**N Micra 2012**		**R Captur 2015**	
	CAN ID (HEX)	*Bit*	*CAN ID (HEX)*	*Bit*
Position Lights	625	9	3B7	16
Medium Lights	625	10	3B7	16
Maximum Lights	625	11	-	-
Windshield Wiper Speed #1	35D	16	-	-
Windshield Wiper Speed #2	35D	18	-	-

Table 5.1 consists of five binary parameters identified in each vehicle. The Nissan Micra 2012 (*N Micra 2012*) shows a positive response to the process presented, identifying all the parameters proposed for this part. Unlike Nissan, the tests on the Renault Captur 2015 (*R Captur 2015*) did not meet expectations, obtaining only two of the five parameters.

As regards the similarity of the results obtained on the Renault, it can be seen that there is no distinction, within the CAN-bus accessible via the OBD-II port, between the position lights and the medium lights. A more detailed analysis of these results is described later in the final considerations.

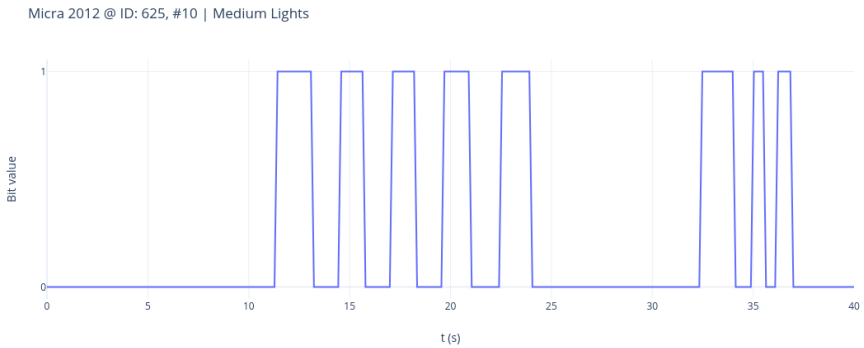


Figure 5.9: Graphic with the transitions of the medium lights of N Micra 2012 in bit #10 of the CAN message with ID 625 over the time of the three phases

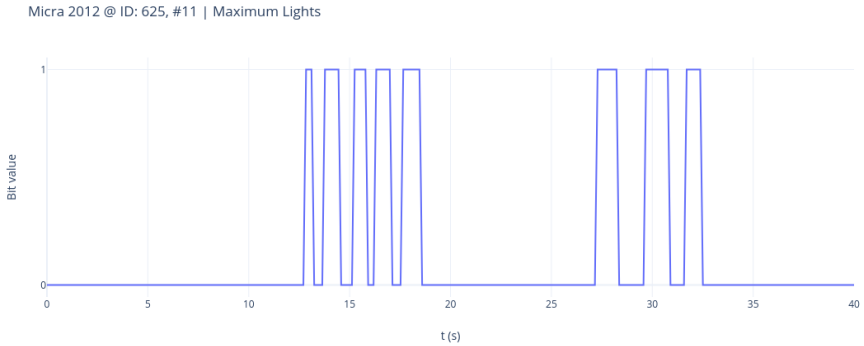


Figure 5.10: Graphic with the transitions of the maximum lights of Nissan Micra 2012 in bit #11 of the CAN message with ID 625 over the time of the three phases

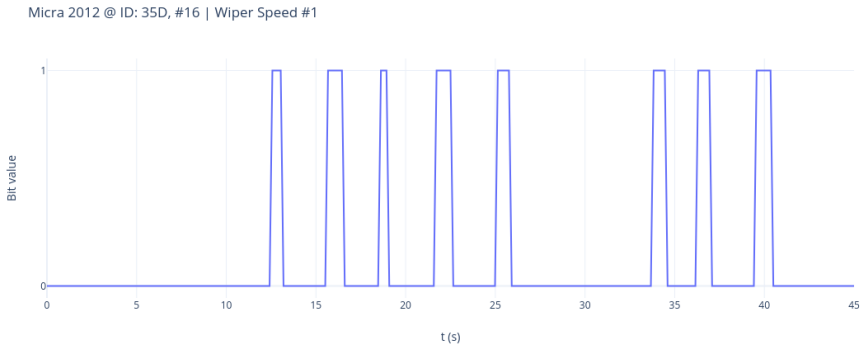


Figure 5.11: Graphic with the transitions of the windshield wiper speed #1 of Nissan Micra 2012 in bit #16 of the CAN message with ID 35D over the time of the three phases

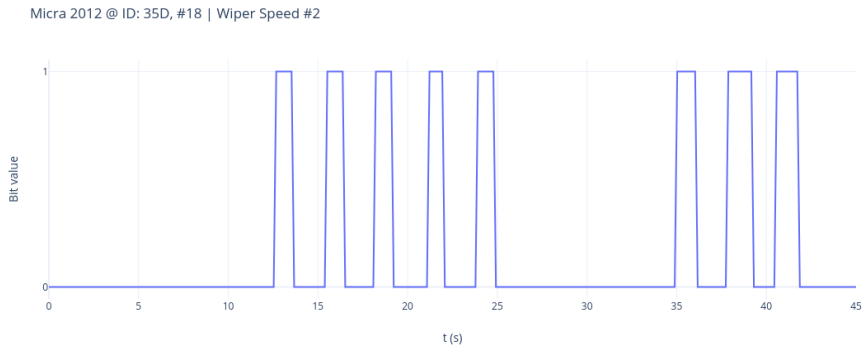


Figure 5.12: Graphic with the transitions of the windshield wiper speed #2 of Nissan Micra 2012 in bit #18 of the CAN message with ID 35D over the time of the three phases

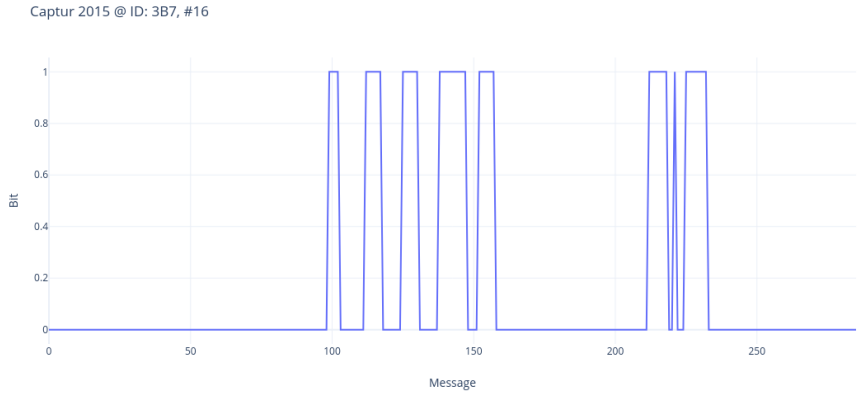


Figure 5.13: Graphic with the transitions of the position lights of Renault Captur 2015 in bit #16 of the CAN message with ID 3B7 over the time of the three phases

5.2.3 Non-Binary parameters

The non-binary parameters are represented by one or more bytes, and therefore, the algorithm needs to perform other processes to find them in CAN messages.

In section 4.3.2 shows that different processes are implemented within this type of parameters. In the *Throttle Pedal Position* and the *Steering Wheel Direction*, the tests are also *static* as in the binary parameters, not requiring any specific place to perform them. In the other parameters, the locations and specifications for the tests are mentioned later. The conditions where the tests occurred are the same as those described in section 5.2.1.

Regarding the *static* tests, figure 5.14 and 5.16 shows how Carloop interacts with the driver to identify *Steering Wheel Direction* and the *Throttle Pedal Position*.

The procedure of both parameters, which is explained in section 4.3.2 of the previous chapter, is similar. However, *Steering Wheel Direction* requires a much longer phase 1. After the reference period corresponding to phase 0, the driver starts phase 1 by using the steering wheel to each side as required. After that, the results are displayed with the CAN ID, and the corresponded byte number.

The same process also identifies the *Throttle Pedal* position; however, in phase 1, it is only requested to press the pedal twice. The graphics on figure 5.15 and 5.17 represent the variations related to the actions caused by the driver in the found parameters, validating the test result. Both figures represent the tests performed at Renault.

```
 --- RENAULT CAPTUR 2015 ---
 --- STEERING WHEEL ANGLE & DIRECTION TEST ---

Rotate 90° to the right ... (5s)
Return to the start position ... (5s)
Rotate 90° to the left ... (5s)
Rotate +90° to the left ... (5s)
Return to the start position ... (5s)

id: c6 - byte#0
RIGHT when the value goes down!
LEFT when the value goes up!
```

Figure 5.14: Identification of the steering wheel position through the process described

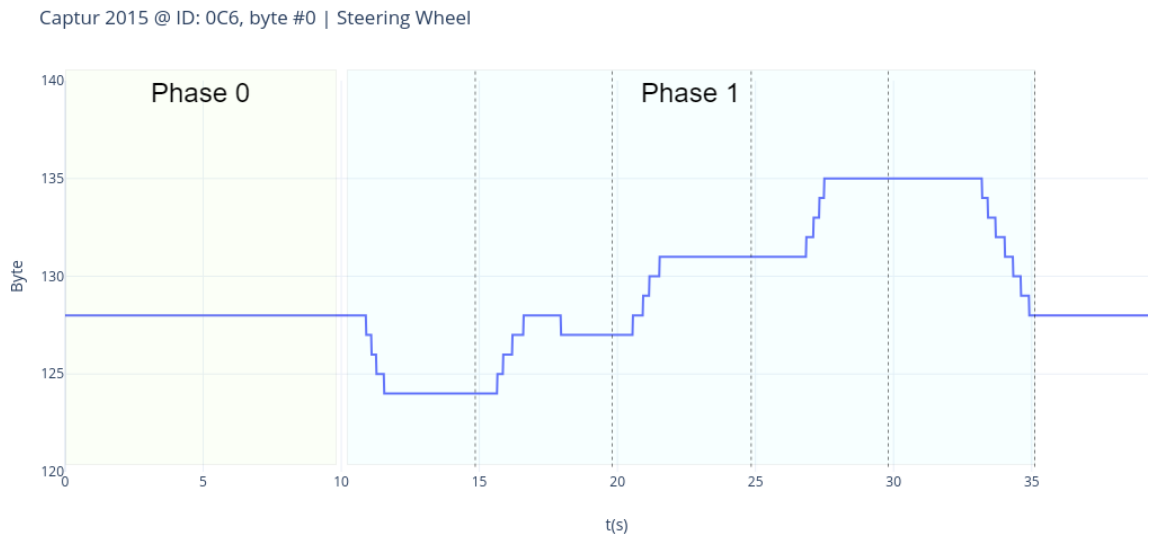


Figure 5.15: Graphic with the variations of the steering wheel in byte #0 of the CAN message with ID 0C6 over the time

```
 --- RENAULT CAPTUR 2015 ---
 --- THROTTLE PEDAL TEST ---

Press the pedal ... (5s)
Remove and wait  7s
Press again the pedal ... (5s)

id: 186 - byte#5
id: 18a - byte#2
```

Figure 5.16: Identification of the throttle position through the process described

Figure 5.17: Graphic with the variations of the throttle pedal in byte #5 of the CAN message with ID 186 over the time

There is no way to test without the car moving regarding the *Vehicle Speed*, *Wheels Speed* and *Engine Speed* parameters. Therefore the implemented method implies a different process of processing and analysis for this type of data.

The scenario chosen to evaluate the new method's performance is a road located in an area with little affluence to avoid disturbances during the tests, which would be impossible in environments with a greater vehicle flow. These validations were made at Rua das Longas, in Ovar, as shown in figure 5.18, during the night. The route chosen is straight to make the detection of wheel speed more effective and is equivalent to approximately 30 seconds of driving, being the time necessary for the method to identify the parameters in CAN messages.

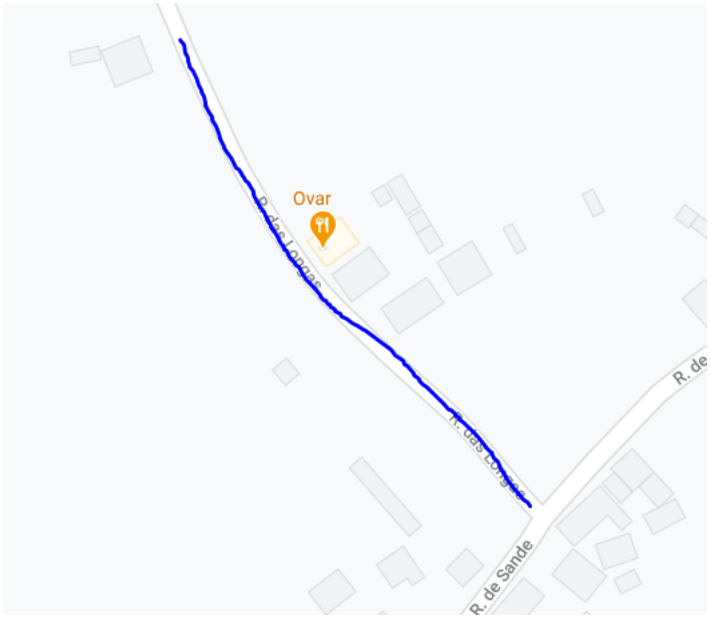


Figure 5.18: Trace in Rua das Longas, Ovar, used to test the method for speeds

The following figures show the results obtained for *Vehicle Speed* and *Wheels Speed* in the

Renault. Figure 5.19 shows the result with the features of the PID 0x0D, corresponding to the vehicle speed. The calculation performed by Carloop is shown as follows: ID, the normalized value of the average, % of ascents divided by 100, % of descents divided by 100 and the normalized position of the maximum value. In the previous chapter, table 4.3 describes these features in more detail.

```
 --- RENAULT CAPTUR 2015 ---
 --- VEHICLE AND WHEEL SPEED TEST ---
Start the 30s trace in 3... 2... 1...
Vehicle Speed PID status (ID, norm_mean, (ascents)/100, (descents)/100, norm_max
_idx):
7e8,0.60,0.52,0.48,0.55
```

Figure 5.19: Result of the features calculations from the values extracted from the PID 0x0D corresponding to the vehicle speed

The algorithm initially filters the CAN messages bytes, which contain a minimum value equal to 0, due to the vehicle starting stopped) and a maximum value between 1 and 254. This filter eliminates data that have not been changed or which are bytes with less significant bits of a certain variable. Then, as shown in figure 5.20, the first candidates with the same values calculated in the PID plus the RMSE value between the candidate features and the PID features are chosen. The candidates with RMSE values below 0.1 are considered the primary candidates for the vehicle speed but proceed to a final analysis to distinguish which of them correspond to the wheels speed.

```
POSSIBLE CANDIDATES
(ID #byte, norm_mean, (ascents)/100, (descents)/100, norm_max_idx, rmse, y/n)
186 #1,0.49,0.55,0.45,0.09,0.2370
186 #5,0.24,0.72,0.28,0.43,0.2349
18a #1,0.24,0.40,0.60,0.08,0.3060
18a #2,0.24,0.69,0.31,0.44,0.2209
217 #1,0.86,0.64,0.36,0.26,0.2071
217 #3,0.53,0.53,0.47,0.48,0.0507, CANDIDATE!
217 #4,0.47,0.48,0.52,0.43,0.0956, CANDIDATE!
217 #5,0.38,0.50,0.50,0.42,0.1287
 90 #0,0.17,0.47,0.53,0.00,0.3496
 c6 #3,0.41,0.50,0.50,0.10,0.2453
12e #4,0.57,0.49,0.51,0.00,0.2755
29a #0,0.51,0.51,0.49,0.43,0.0751, CANDIDATE!
29a #2,0.52,0.51,0.49,0.43,0.0714, CANDIDATE!
29a #4,0.51,0.52,0.48,0.43,0.0765, CANDIDATE!
29a #6,0.50,0.89,0.11,0.01,0.3791
29c #0,0.52,0.51,0.49,0.41,0.0828, CANDIDATE!
29c #2,0.52,0.51,0.49,0.40,0.0860, CANDIDATE!
354 #0,0.53,0.50,0.50,0.47,0.0563, CANDIDATE!
354 #1,0.48,0.44,0.56,0.41,0.1089
211 #3,0.74,0.50,0.50,0.01,0.2761
5e9 #7,0.87,1.00,0.00,0.13,0.4174
7e8 #3,0.60,0.52,0.48,0.55,0.0000, CANDIDATE!
653 #5,0.23,0.49,0.51,0.06,0.3081
500 #1,0.50,0.53,0.47,0.14,0.2121
500 #3,0.50,0.47,0.53,0.86,0.1704
563 #0,0.51,0.51,0.49,0.20,0.1810
563 #1,0.46,0.50,0.50,0.04,0.2625
564 #0,0.46,0.52,0.48,0.56,0.0696, CANDIDATE!
564 #1,0.49,0.47,0.53,0.05,0.2559
352 #3,0.18,0.45,0.55,0.00,0.3503
5d7 #0,0.51,0.53,0.47,0.55,0.0484, CANDIDATE!
5d7 #1,0.48,0.49,0.51,0.40,0.0973, CANDIDATE!
5d7 #5,0.41,0.95,0.05,0.57,0.3177
511 #5,0.50,0.76,0.24,0.14,0.2695
511 #6,0.54,0.46,0.54,0.11,0.2250
```

Figure 5.20: Indication of the candidates to the vehicle speed from the RMSE values

Since the vehicle travels on a straight road, the *Wheels Speed* parameters have to show the same scale and offset in CAN messages, and therefore, the maximum value reached has to be very similar in all four. Figure 5.21 shows the identification of the four-wheel speed parameters. The interval [34, 35] indicates that four candidates contain a maximum value between 34 and 35, concluding that they are the parameters corresponding to the wheels. In some instances, a slight curve on the street coincides when the vehicle reaches its maximum speed during the test. The route represented in figure 5.18 shows that exists a small curve during the test, which explains that three of the four parameters have a maximum value of 34 and another parameter has a maximum value of 35. Through this technique, by searching the parameters in intervals, it is possible to fix these small coincidences.

```
[26 27] = 0
[27 28] = 0
[28 29] = 1
[29 30] = 1
[30 31] = 0
[31 32] = 0
[32 33] = 0
[33 34] = 3
[34 35] = 4
WHEEL SPEED FOUND!
id: 29a - byte #0
id: 29a - byte #2
id: 29c - byte #0
id: 29c - byte #2

[35 36] = 1
[36 37] = 0
[37 38] = 0
[38 39] = 0
[39 40] = 0
[40 41] = 0
[41 42] = 0
[42 43] = 0
[43 44] = 1
[44 45] = 1
[45 46] = 0
```

Figure 5.21: Result of the method of detection of the wheel speed parameters, with the message ID and the byte corresponding to the position found

The graphics of figures 5.22 and 5.23 show the values of the PID values and four bytes' values over the time to validate the similarity between them.

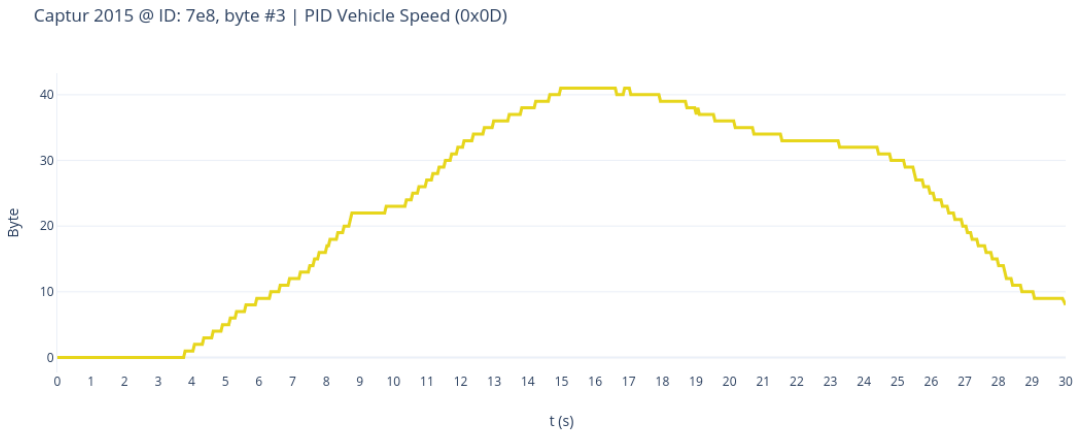
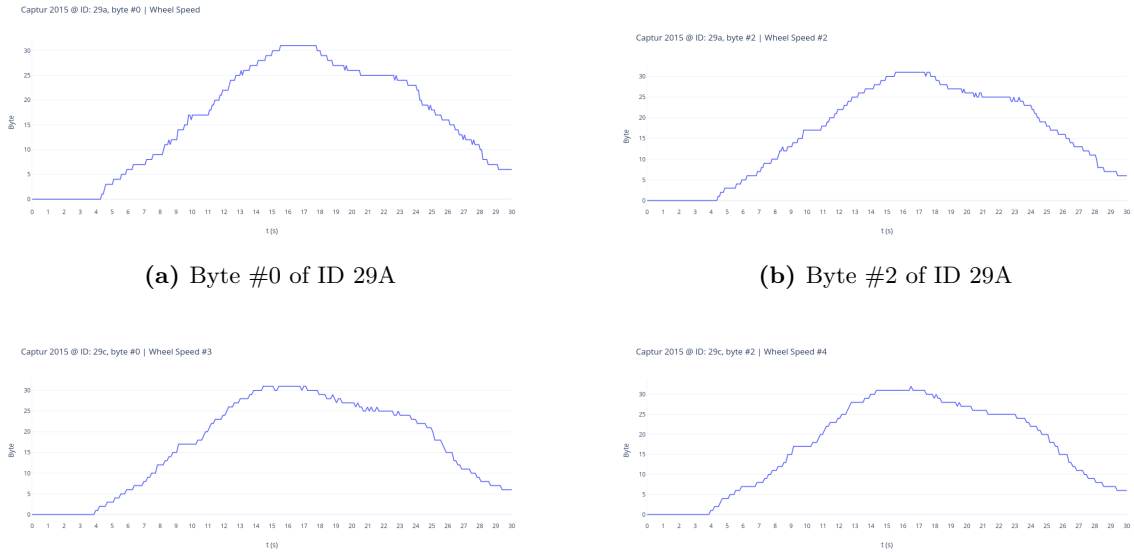


Figure 5.22: Graphic with the variation of the vehicle speed in byte #3 of the CAN message with ID 7E8 and PID 0x0D over the time



(a) Byte #0 of ID 29A

(b) Byte #2 of ID 29A

(c) Byte #0 of ID 29C

(d) Byte #2 of ID 29C

Figure 5.23: Graphics of the variation of the wheel speed parameters found over time

This method is also used to find the *Engine Speed* parameter. The only difference is that the primary candidates are only considered if they have an RSME value below 0,05. The results show that byte #0 of ID 186 match the features of byte #3 of PID 0x0D. The graphics of these bytes are presented in figures 5.24 and 5.25 to show the similarity between them.

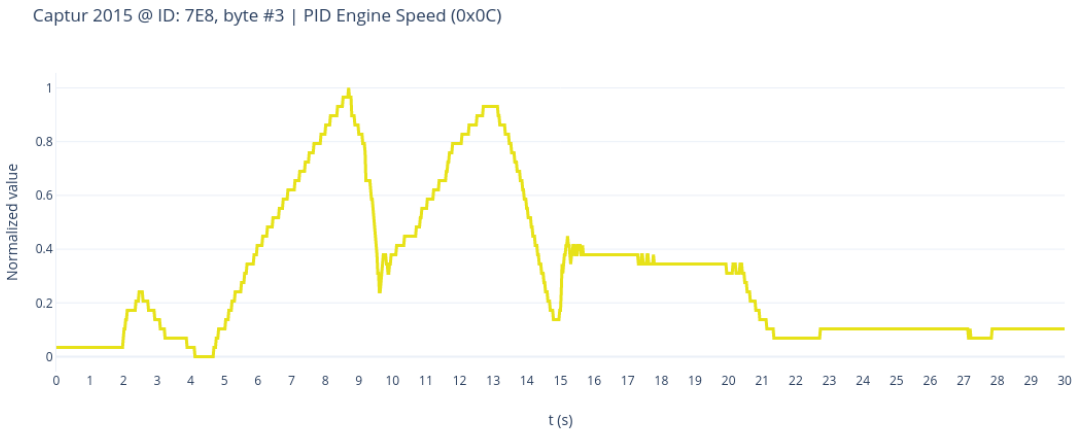


Figure 5.24: Graphic with the variation of the engine speed in byte #3 of the CAN message with ID 7E8 and PID 0x0C over the time

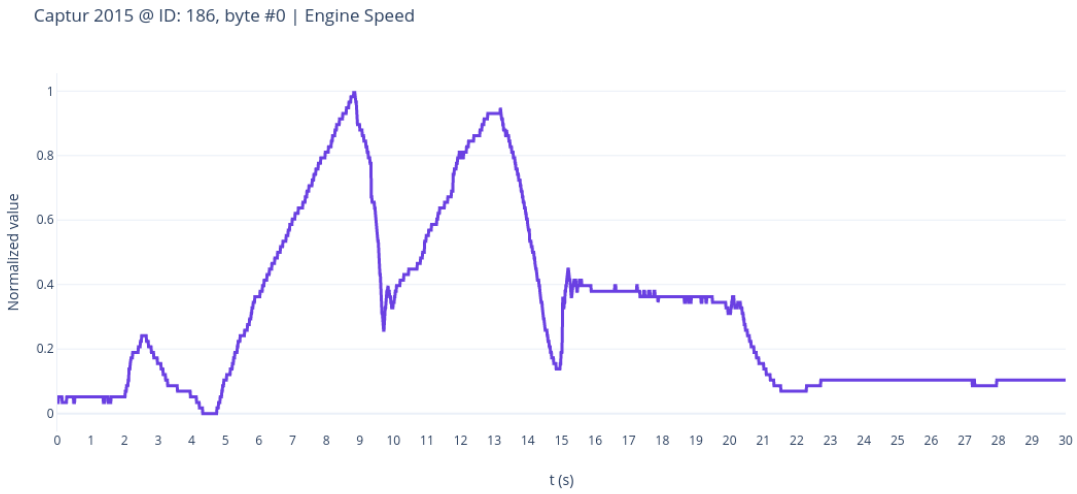


Figure 5.25: Graphic with the variation of the engine speed in byte #0 of the CAN message with ID 186 over the time

Notice that all the proposed non-binary parameters are identified by only one byte. The algorithm developed in this work allows finding a byte that can identify the parameters. However, some parameters have two bytes of information, such as the steering wheel direction parameter. Figure 5.26 shows all the bit-flips of the message with the ID 0C6 that occurred during the test. If we analyze this figure more in detail, as shown in figure 5.27, we notice a growth in the number of bit-flips from the Most Significant Bit (MSB) (0) to the Least Significant Bit (LSB) (15), indicating a parameter of two bytes.

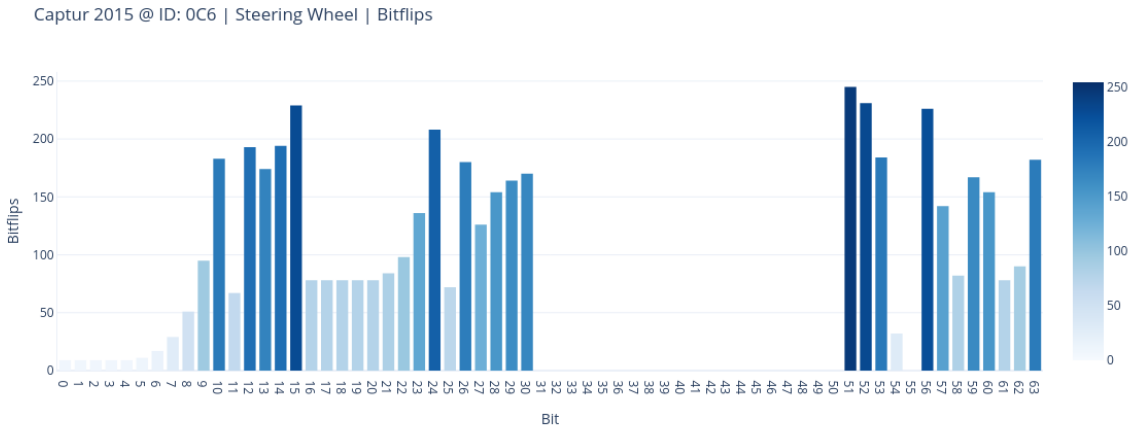


Figure 5.26: Number of bit-flips of the CAN ID 0C6 message during the test

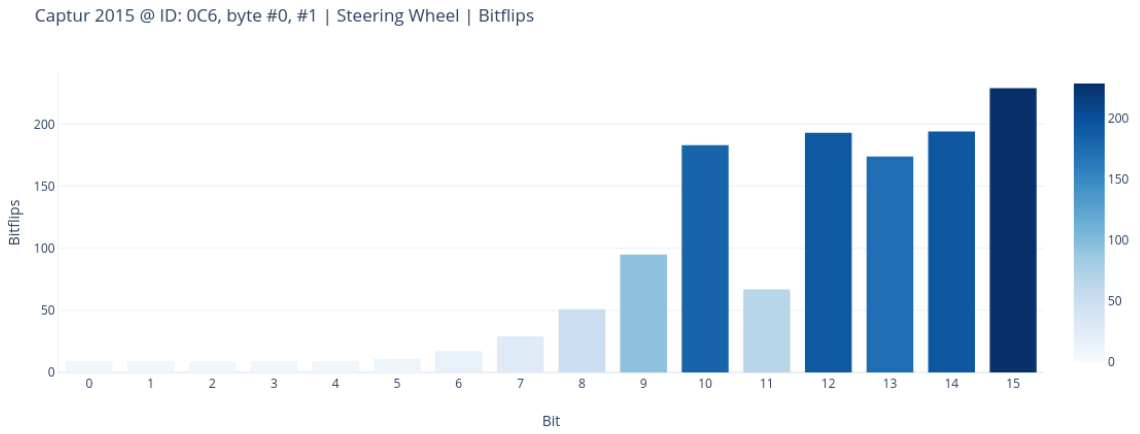


Figure 5.27: First two bytes' bit-flips of the CAN ID 0C6

Based on this analysis, it is affirmative to say that the CAN message with the ID 0C6 contains the steering wheel angle values in bytes #0 and #1 of the data field.

The results obtained by the algorithm in this type of parameters on Renault Captur are resumed in Table 5.2. This table also contains the offsets, scales and units of the physical values. The next formula shows how to obtain the values from the decimal value of the parameters.

$$physical\_value = offset + scale * decimal\_value, \tag{5.1}$$

where the *offset* is the value to offset the physical value, the *scale* is the value to multiply the physical value.

Table 5.2: Non-Binary parameters of Renault Captur with the respective CAN ID, Byte(s), Unit, Offset and Scale

R Captur 2015	CAN ID (HEX)	Byte(s)	Unit	Offset	Scale
Steering Wheel Angle	0C6	0-1	º	8000	0.1
Steering Wheel Direction	0C6	0	bool	-	-
Throttle Pedal	186	5	%	0	0.4
Vehicle Speed	217	3 - 4(7-4)	km/h	0	0.1
Engine Speed	186	0-1	rpm	0	0.125
Wheels Speed	29A	0-1	km/h	0	0.005
	29A	2-3	km/h	0	0.005
	29C	0-1	km/h	0	0.005
	29C	2-3	km/h	0	0.005

The tests described in this section were performed in the same way for the Nissan Micra 2012. The results of the tests are synthesized in Table 5.3. This table also contains the offsets, scales and units of the physical values converted from hexadecimal to an integer.

Table 5.3: Non-Binary parameters of Nissan Micra with the respective CAN ID, Byte(s), Unit, Offset and Scale

N Micra 2012	CAN ID (HEX)	Byte(s)	Unit	Offset	Scale
Steering Wheel Angle	-	-	-	-	-
Steering Wheel Direction	-	-	-	-	-
Throttle Pedal	182	4	%	0	1
Vehicle Speed	280	4-5	km/h	0	0.01
Engine Speed	180	0-1	rpm	0	0.125
Wheels Speed	284	0-1	km/h	0	0.005
	284	2-3	km/h	0	0.005
	285	0-1	km/h	0	0.005
	285	2-3	km/h	0	0.005

Based on the two tables, it can be seen that the results obtained for these parameters were quite positive. In Renault they were 100% achieved. However, in Nissan it was not possible to find the message regarding the steering wheel of the vehicle. The reason why the algorithm failed in this parameter is explained in detail below.

5.2.4 Identification of Wheels Position in specific traces

Once the four wheels' parameters are identified, it is easier to identify the corresponded position of every wheel. Firstly, it is needed to collect CAN data again, describing one or more curves to verify the wheels' speed differences.

When the vehicle moves on a straight street, the wheels follow the same line and, for that reason, they rotate at the same speed. However, when the vehicle rotates, they describe a different circumference line, drawing a smaller circumference on the internal wheels and larger circumferences on the external wheels. As a result, the externals wheels ride a longer path and need to speed up to follow the other wheels, as shown in figure 5.28.

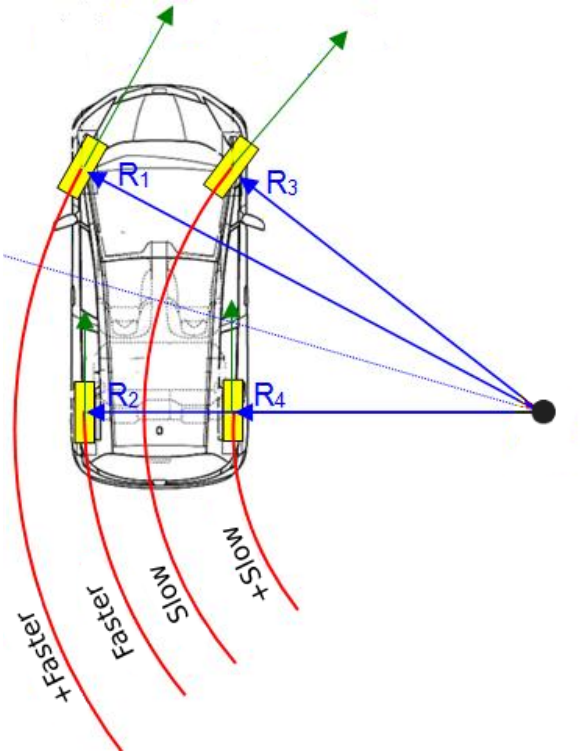


Figure 5.28: Illustration of the different circumferences of the wheels in a curve. Adapted from [76]

This control is made by a differential. The differential is a fundamental element placed in the wheel axle and can adjust the wheels speed in a curve. This mechanism allows the semi-axles (which connect the differential to the wheel) to have different rotation speeds with the same torque.

The scenarios chosen to check these differences in the wheel's speed are two roads with an associated curve. The test for the right-side curve is made from Rua Cimo de Vila to Rua Montes de Sandes, located in Ovar and the test for the left-side curve is performed from Rua Montes de Sandes to Rua do Beira Monte, as shown in figure 5.29 and 5.30.

Figure 5.29: Right curve between Rua Cimo de Vila and Rua Montes de Sandes

Figure 5.30: Left curve between Rua Montes de Sandes and Rua do Beira Monte

During the tests, it was raining and dark. The photos of figures 5.31 and 5.32, taken at the moment of the test in Renault and Nissan prove this conditions.

Figure 5.31: Picture taken at the moment of the test with Nissan Micra 2012

Figure 5.32: Picture taken at the moment of the test with Renault Captur 2015

The graphic in figure represents the *Wheels Speed* parameters' variation presented in the table 5.2 of Renault, when the car is describing the right curve.

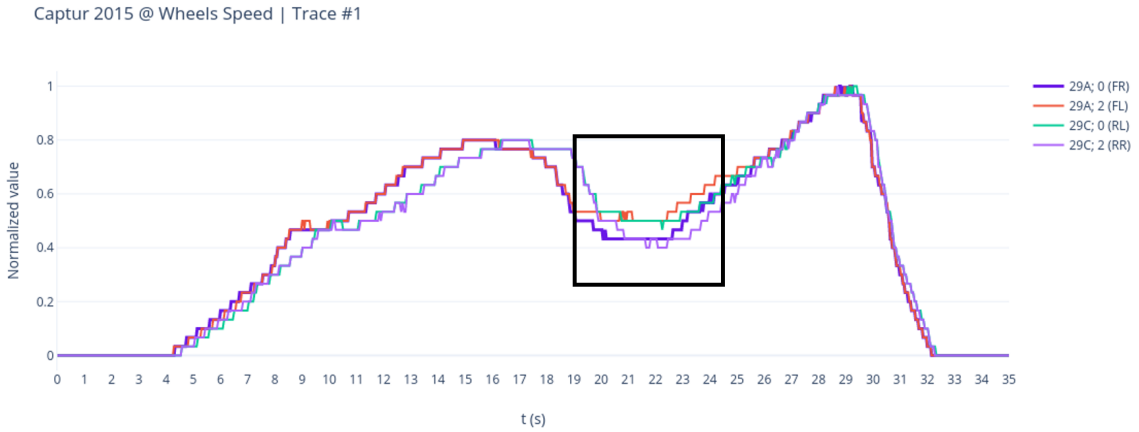


Figure 5.33: Graphic of the variation of the wheel speed parameters and the identification of the moment of the right curve

First of all, it is revealing to note that the number of messages of the two messages 29A and 29C are not the same, and therefore, even if the graphics are normalized, it can be visible a small delay in one of the messages. However, the curve identified by the square drawn in the figure is important to take from this test. There are small differences in the wheel's speed in the curve area, thus proving the theory explained above. The light purple line should represent the right rear wheel, and the orange line describes the speed of the left front wheel, the one that describes a larger circumference in a right curve. Table 5.4 demonstrates the results obtained.

Table 5.4: Identification of the values for each wheel in Renault Captur

R Captur 2015	Position	CAN ID (HEX)	Byte(s)
Wheels Speed	*Front Right (FR)*	29A	0-1
	Rear Right (RR)	29C	2-3
	Front Left (FL)	29A	2-3
	Rear Left (RL)	29C	0-1

The second test is performed with the vehicle rotating to the left to validate the previous identification. The right wheels will have to move faster than the left wheels, thus, it is expected the opposite result regarding the first trace. The graphic of figure 5.34 shows the result of the route taken.

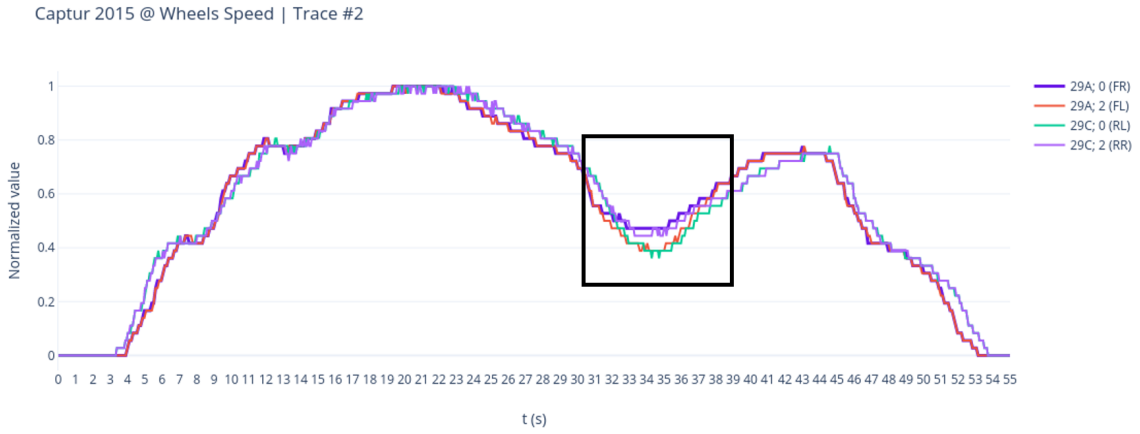


Figure 5.34: Graphic of the variation of the wheel speed parameters and the identification of the moment of the left curve

Based on this graphic, it is possible to validate the results presented in table 5.4. The FR wheel is represented by the dark purple color, the RR wheel by the light purple color, the FL wheel by the orange color and the RL wheel by the green color.

Table 5.5 represents the results for Nissan Micra 2012.

Table 5.5: Identification of the values for each wheel in Nissan Micra

N Micra 2012	Position	CAN ID (HEX)	Byte(s)
Wheels Speed	*Front Right (FR)*	284	0-1
	Rear Right (RR))	285	0-1
	Front Left (FL)	284	2-3
	Rear Left (RL)	285	2-3

5.2.5 Final considerations

As mentioned throughout this chapter, communication tests were performed to validate Carloop's communication with the IT2S platform in two ways: cable and wireless. The results of the tests ensured that Carloop could transmit the parameters of CAN messages in both configurations.

Subsequently, the extraction tests of the parameters mentioned in table 4.2 begin. These tests have validated the solutions implemented to extract binary and non-binary parameters through *static* and *dynamic* methods. The data extracted from the vehicles was positive for both cars but showed some differences with better results in the Nissan Micra 2012 for binary parameters and better results in the Renault Captur 2015 for non-binary parameters.

Regarding the results of Renault's binary parameters, the algorithm could not detect the windshields' movement and the maximum lights. In newer cars, there is a type of bus supplementary to the CAN bus. The Local Interconnect Network (LIN) is a protocol with low performance and low costs due to the use of only one wire for communication and not

requires a high communication speed [77]. Typically, the LIN Master serves as a gateway to the CAN bus and is used for messages regarding windows, air conditioning and windshields [78]. Due to the lack of documentation of the in-vehicle network, it is only estimated that the problem comes from here.

In the non-binary parameters, Renault's results exceeded expectations, unlike the Nissan results that were not fully achieved. However, after manual analysis of the messages captured from the bus, it is concluded that it is not detected by the method used being weak but by the fact that there is no reference to the steering wheel angle in CAN messages. During this analysis, it was found that there is only one value that represents the acceleration of the steering wheel and that it is located in the message with ID 2 in byte #2, as shown in the graphic of figure 5.35 concerning the trip represented in figure 5.30.

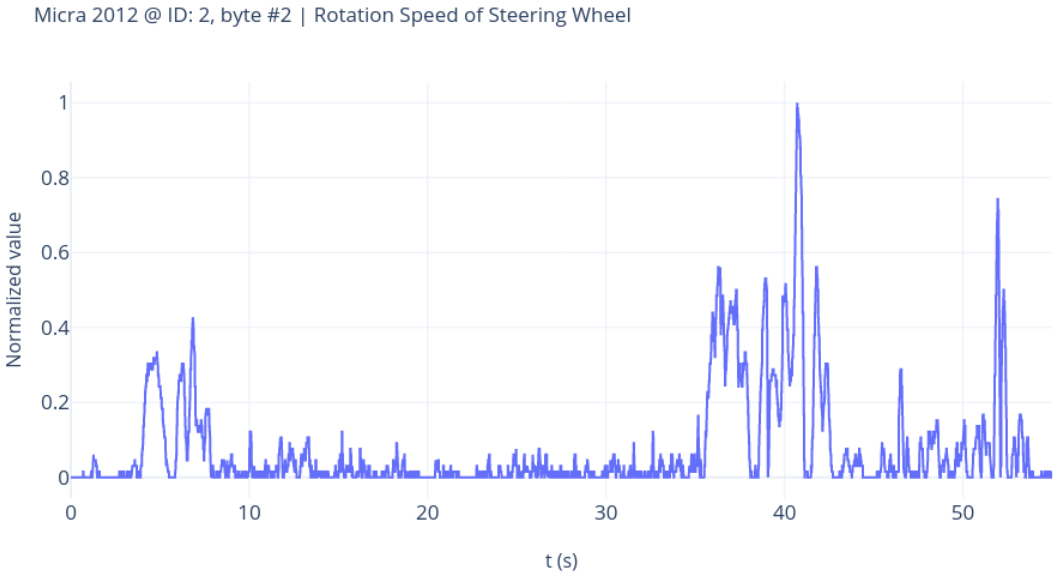


Figure 5.35: Graphic with the values of byte #2 of ID 2 representing the rotation speed of the steering wheel

The graphic peaks refer to the steering wheel's turning when the car leaves the parking lot, drives the curve for the test and leans against the roadside at the end of the test.

At the end, an analysis is added to the wheels speed to assign the corresponding positions to each wheel. This test also shows how the data extracted from the vehicle can inform about the state of the weather during a drive through the state of the windshield and the state of the lights. The remaining parameters also help in road safety, such as detecting of sudden braking, which can be seen at the end of the graphic in figure 5.32.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

This work studies how to extract and decode raw CAN messages transmitted within vehicle communication networks. A solution to find specific parameters for a weather-related project was structured, implemented and evaluated by a device connected to the OBD-II port of the vehicles with an algorithm to apply reverse engineering on the CAN bus.

This work was developed due to the lack of results in online parameter detection, with the primary objective of developing an iteration algorithm with the driver to contribute to the evolution of ITS. The speed and efficiency of the parameter detection and the transmission of this data to the OBU are other important objectives of this dissertation. It was stepped several tasks to achieve the proposed objectives. Starting from the study of concepts involved with this dissertation and the review of the state-of-the-art to the design of the system architecture, development of the implementation and the validation of the planned implementation.

During this work, many tests were made to communicate with OBU and extract parameters from raw CAN signals. The results of the tested communications were both positive with some advantages and disadvantages between them. The main advantage of cable communication is data transmission speed, not compromising delays or failures in the transmission of parameters to the OBU. However it is not easy to have cables inside the vehicle without them being able to disturb or obstruct the conductor. In an era when wired communication is falling into disuse, wireless communication becomes an inherent option each time, and that is why BLE technology is chosen. Although the latency levels are higher than the USB communication, it does not prevent this technology from being the one that will be used in the TRUST project.

At the level of the work performed inside the vehicles, tests were made to the lights, windshields, pedals, steering wheel, engine speed, vehicle speed, and wheel speeds. Other parameters are not tested, such as External Air Temperature, Barometric Pressure, and Engine Load. However, they can be obtained through the implemented method to relate the messages with PIDs or extract the value directly by transmitting a request-message to the bus with the intended PID. The results of the tested parameters were positive, as mentioned

in the previous chapter. Still, not all parameters were found, which concludes that there may be gateways and other communication protocols within the in-vehicle network that make it impossible to find them from reading on the OBD-II port. Another aspect that interfered in the results was the specificity of the iterative methods with the driver: the case of the parameter referring to the steering wheel angle happens because the test did not correspond to how Nissan configured the parameter's behaviour in the system. Nevertheless, this event permits insight know-how into how OEMs develop their systems. The results also showed that they could have different offset and scale for the parameters represented by one or more bytes.

The method implemented to relate CAN messages to diagnostic messages shows that it is unnecessary to have databases with millions of CAN messages to obtain the desired results. This work has proven that only a 30-second collection is needed to achieve this goal.

It was concluded that the results are very exciting and promising and bring a contribution to this area.

6.2 Future work

During the development of this work, which culminated in the individual evaluation, it was perceptible that certain aspects of the implementation could be improved. Thus, here are some study topics that stand out for future developments:

- Check the effectiveness of this algorithm on a larger number of vehicles, from the oldest to the most modern vehicle;
- Explore how OEMs represent the parameters in intra-vehicle messages in order to develop new methods to detect them;
- Develop a device with a system dedicated to this function, preferably with more memory to implement new methods for more accurate estimations such as the coefficient of variation;
- Develop other methods for the most modern vehicles with new sensors that can be added to the list of parameters to evaluate the weather conditions, such as a sun sensor and rain sensor.
- Test the two forms of communication with the OBU in a real environment.

References

[1] *World vehicle population tops 1 billion units*, Accessed: October 2020. [Online]. Available: https://www.wardsauto.com/news-analysis/world-vehicle-population-tops-1-billion-units.

[2] *Automotive industry worldwide - statistics & facts*, Accessed: November 2020. [Online]. Available: https://www.statista.com/topics/1487/automotive-industry/#:~:text=Global%5C%20sales%5C%20of%5C%20automobiles%5C%20are,commercial%5C%20vehicles%5C%20and%5C%20passenger%5C%20cars.

[3] *Number of cars sold worldwide between 2010 and 2021*, Accessed: November 2020. [Online]. Available: https://www.statista.com/statistics/200002/international-car-sales-since-1990/.

[4] *European comission, statistics - accident data (road fatalities in the eu since 2001)*, Accessed: November 2020. [Online]. Available: https://ec.europa.eu/transport/road_safety/specialist/statistics_en.

[5] *Obd-ii background*, Accessed: January 2020. [Online]. Available: http://www.obdii.com/background.html.

[6] *Advantages of vehicle to vehicle communication*, Accessed: October 2020. [Online]. Available: https://www.azuga.com/blog/vehicle-to-vehicle-communication-benefits.

[7] *We hacked a ford focus and a volkswagen polo*, Accessed: April 2020. [Online]. Available: https://www.which.co.uk/news/2020/04/we-hacked-a-ford-focus-and-a-volkswagen-polo/.

[8] *Tesla model x hacked with $195 raspberry pi based board*, Accessed: November 2020. [Online]. Available: https://www.embedded.com/tesla-model-x-hacked-with-195-raspberry-pi-based-board/.

[9] D. Enriquez, A. Bautista, P. Field, S.-i. Kim, S. Jensen, M. Ali, and J. Miller, "Canopnr: Can-obd programmable-expandable network-enabled reader for real-time tracking of slippery road conditions using vehicular parameters", in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, IEEE, 2012, pp. 260–264.

[10] C. Van Geem, M. Bellen, B. Bogaerts, B. Beusen, B. Berlémont, T. Denys, P. De Meulenaere, L. Mertens, and P. Hellinckx, "Sensors on vehicles (sensovo)–proof-of-concept for road surface distress detection with wheel accelerations and tof camera data collected by a fleet of ordinary vehicles", *Transportation Research Procedia*, vol. 14, pp. 2966–2975, 2016.

[11] M. N. Iqbal, L. Y. Xin, W. U. Rehman, A. Rakhio, S. Siddique, D. Zahid, W. Yasin, and A. B. Waqar, "Diagnostic tool and remote online diagnostic system for euro standard vehicles", in *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*, IEEE, 2017, pp. 415–419.

[12] I. Galanis, I. Anagnostopoulos, P. Gurunathan, and D. Burkard, "Environmental-based speed recommendation for future smart cars", *Future Internet*, vol. 11, no. 3, p. 78, 2019.

[13] N. AbuAli, "Advanced vehicular sensing of road artifacts and driver behavior", in *2015 IEEE Symposium on Computers and Communication (ISCC)*, IEEE, 2015, pp. 45–49.

[14] M. Bartos, H. Park, T. Zhou, B. Kerkez, and R. Vasudevan, "Windshield wipers on connected vehicles produce high-accuracy rainfall maps", *Scientific reports*, vol. 9, no. 1, pp. 1–9, 2019.

[15] M. Chapman, S. Drobot, T. Jensen, C. Johansen, W. Mahoney III, P. Pisano, and B. McKeever, "Using vehicle probe data to diagnose road weather conditions—results from the detroit intellidrive (sm) field study", *Transportation Research Record*, vol. 2169, pp. 116–127, 2010.

[16] W. P. Mahoney III and J. M. O'Sullivan, "Realizing the potential of vehicle-based observations", *Bulletin of the American Meteorological Society*, vol. 94, no. 7, pp. 1007–1018, 2013.

[17] R. B. GmbH, "Can specification, version 2.0", 1991.

[18] *Canbus: The central networking system of vehicles*, Accessed: December 2019, Jun. 2019. [Online]. Available: https://premioinc.com/blogs/blog/can-bus-the-central-networking-system-of-vehicles.

[19] "Road vehicles — Controller area network (CAN) — Part 2: High-speed medium access unit", International Organization for Standardization, Standard, Dec. 2016.

[20] *Can physical layer and termination guide*, Accessed: November 2020, 2020. [Online]. Available: https://www.ni.com/pt-pt/innovations/white-papers/09/can-physical-layer-and-termination-guide.html.

[21] *Why are termination networks in can transceivers so important?*, Accessed: October 2020, 2016. [Online]. Available: https://e2e.ti.com/blogs_/b/industrial_strength/archive/2016/07/14/the-importance-of-termination-networks-in-can-transceivers.

[22] J. A. Cook and J. S. Freudenberg, "Controller area network (can)", 2008.

[23] "Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services", International Organization for Standardization, Standard, Apr. 2016.

[24] *All about can bus*, Accessed: December 2019, 2012. [Online]. Available: https://www.kanda.com/blog/microcontrollers/bus/.

[25] "Road vehicles — Communication between vehicle and external equipment for emissions-related diagnostics — Part 5: Emissions-related diagnostic services", International Organization for Standardization, Standard, 2015.

[26] "Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling", International Organization for Standardization, Standard, Dec. 2015.

[27] *Can fd explained - a simple intro (2020)*, Accessed: November 2020. [Online]. Available: https://www.csselectronics.com/screen/page/can-fd-flexible-data-rate-intro/language/en.

[28] *Can fd - the basic idea*, Accessed: November 2020. [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-fd/.

[29] *Controller area network (can) and flexible data-rate (can fd)*, Accessed: November 2020. [Online]. Available: https://www.kvaser.com/about-can/can-fd/.

[30] *Can xl is knocking on the door*, Accessed: November 2020. [Online]. Available: https://www.can-cia.org/news/cia-in-action/view/can-xl-is-knocking-on-the-door.

[31] *Can xl and can fd light*, Accessed: November 2020. [Online]. Available: https://www.can-cia.org/news/cia-in-action/view/can-xl-and-can-fd-light/2020/5/28/.

[32] C. Senger, "Can xl error detection capabilities", *CAN Newsletter*, Jun. 22, 2020.

[33] *Can xl: Next step in can evolution*, Accessed: November 2020. [Online]. Available: https://www.bosch-semiconductors.com/news/t-newsdetailpage-4.html.

[34] *Can xl: Bridging the bitrate gap between can fd and ethernet*, Accessed: November 2020. [Online]. Available: https://www.kvaser.com/can-xl-bridging-the-bitrate-gap-between-can-fd-and-ethernet.

[35] C. Pijolat, C. Pupier, M. Sauvan, G. Tournier, and R. Lalauze, "Gas detection for automotive pollution control", *Sensors and Actuators B: Chemical*, vol. 59, no. 2, pp. 195–202, 1999.

[36] M. A. K. Niazi, A. Nayyar, A. Raza, A. U. Awan, M. Ali, N. Rashid, and J. Iqbal, "Development of an on-board diagnostic (obd) kit for troubleshooting of compliant vehicles", 2013.

[37] *Obd2 explained - a simple intro (2020)*, Accessed: October 2020, 2020. [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-obd2-explained/.

[38] J. Laukkonen, *Elm327 programmed microcontroller car diagnostics*, Accessed: June 2020, Feb. 2020. [Online]. Available: https://www.lifewire.com/elm327-microcontroller-car-diagnostics-534688.

[39] *What's the difference between bluetooth le and bluetooth spp (ble vs spp)?*, Accessed: June 2020. [Online]. Available: https://www.serialio.com/faqs/whats-difference-between-bluetooth-le-and-bluetooth-spp-ble-vs-spp.

[40] *Bluetooth vs. bluetooth low energy: What's the difference?*, Accessed: June 2020, Nov. 2015. [Online]. Available: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy.

[41] *What is bluetooth low energy (ble) and how does it work?*, Accessed: June 2020, Mar. 2019. [Online]. Available: https://www.centare.com/blog/what_is_bluetooth_low_energy/.

[42] "Bluetooth Core Specification Version 5.1 ", SIG Bluetooth, Tech. Rep., Jan. 2019, Accessed: July 2020.

[43] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology", *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[44] *Bluetooth® low energy packet types*, Accessed: July 2020. [Online]. Available: https://microchipdeveloper.com/wireless:ble-link-layer-packet-types.

[45] *Ble protocol stack — host controller interface (hci)*, Accessed: June 2020, Sep. 2019. [Online]. Available: https://medium.com/@pcng/ble-protocol-stack-host-controller-interface-hci-44dd5697bd8.

[46] *Gap | introduction to bluetooth low energy*, Accessed: July 2020. [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap.

[47] *Gatt | introduction to bluetooth low energy*, Accessed: July 2020. [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt.

[48] W. S. Jeon, M. H. Dwijaksara, and D. G. Jeong, "Performance analysis of neighbor discovery process in bluetooth low-energy networks", *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 1865–1871, 2017.

[49] W. Bronzi, R. Frank, G. Castignani, and T. Engel, "Bluetooth low energy performance and robustness analysis for inter-vehicular communications", *Ad Hoc Networks*, vol. 37, pp. 76–86, 2016.

[50] M. Ryan, "Bluetooth: With low energy comes low security", in *7th {USENIX} Workshop on Offensive Technologies ({WOOT} 13)*, 2013.

[51] S. Chumkamon, P. Tuvaphanthaphiphat, and P. Keeratiwintakorn, "The vertical handoff between gsm and zigbee networks for vehicular communication", in *ECTI-CON2010: The 2010 ECTI International Confernce on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, IEEE, 2010, pp. 603–606.

[52] F. A. Teixeira, V. F. e Silva, J. L. Leoni, D. F. Macedo, and J. M. Nogueira, "Vehicular networks using the ieee 802.11 p standard: An experimental analysis", *Vehicular Communications*, vol. 1, no. 2, pp. 91–96, 2014.

[53] W. Bronzi, R. Frank, G. Castignani, and T. Engel, "Bluetooth low energy for inter-vehicular communications", in *2014 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2014, pp. 215–221.

[54] M. Uysal, Z. Ghassemlooy, A. Bekkali, A. Kadri, and H. Menouar, "Visible light communication for vehicular networking: Performance study of a v2v system using a measured headlamp beam pattern model", *IEEE Vehicular Technology Magazine*, vol. 10, no. 4, pp. 45–53, 2015.

[55] A. Festag, "Standards for vehicular communication—from ieee 802.11 p to 5g", *e & i Elektrotechnik und Informationstechnik*, vol. 132, no. 7, pp. 409–416, 2015.

[56] R. Molina-Masegosa and J. Gozalvez, "Lte-v for sidelink 5g v2x vehicular communications: A new 5g technology for short-range vehicle-to-everything communications", *IEEE Vehicular Technology Magazine*, vol. 12, no. 4, pp. 30–39, 2017.

[57] J. Choi, V. Va, N. Gonzalez-Prelcic, R. Daniels, C. R. Bhat, and R. W. Heath, "Millimeter-wave vehicular communication to support massive automotive sensing", *IEEE Communications Magazine*, vol. 54, no. 12, pp. 160–167, 2016.

[58] "Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service", European Telecommunications Standards Institute, Standard, Jan. 2019.

[59] "Intelligent transport systems (its); testing; part 1: Conformance test specifications for co-operative awareness messages (cam); cam validation report", European Telecommunications Standards Institute, Standard, Apr. 2014.

[60] J. Santa, F. Pereñiguez, A. Moragón, and A. F. Skarmeta, "Experimental evaluation of cam and denm messaging services in vehicular communications", *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 98–120, 2014.

[61] "Intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service", European Telecommunications Standards Institute, Standard, Apr. 2019.

[62] H. M. Song and H. K. Kim, "Discovering can specification using on-board diagnostics", *IEEE Design & Test*, 2020.

[63] T. Huybrechts, Y. Vanommeslaeghe, D. Blontrock, G. Van Barel, and P. Hellinckx, "Automatic reverse engineering of can bus data using machine learning techniques", in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Springer, 2017, pp. 751–761.

[64] M. Marchetti and D. Stabili, "Read: Reverse engineering of automotive data frames", *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2018.

[65] M. Markovitz and A. Wool, "Field classification, modeling and anomaly detection in unknown can bus networks", *Vehicular Communications*, vol. 9, pp. 43–52, 2017.

[66] M. Verma, R. Bridges, and S. Hollifield, "Actt: Automotive can tokenization and translation", in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 2018, pp. 278–283.

[67] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "Librecan: Automated can message translator", in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2283–2300.

[68] C. Young, J. Svoboda, and J. Zambreno, "Towards reverse engineering controller area network messages using machine learning", *IEEE WF-IoT IEEE*, 2020.

[69] M. E. Verma, R. A. Bridges, J. J. Sosnowski, S. C. Hollifield, and M. D. Iannacone, "Can-d: A modular four-step pipeline for comprehensively decoding controller area network data", *arXiv preprint arXiv:2006.05993*, 2020.

[70] D. Frassinelli, S. Park, and S. Nürnberger, "I know where you parked last summer: Automated reverse engineering and privacy analysis of modern cars", in *2020 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2020, pp. 1401–1415.

[71] *Pc engines - apu3 series system board*, Accessed: December 2020, PC Engines GmbH, 2017. [Online]. Available: `https://www.pcengines.ch/pdf/apu3.pdf`.

[72] *Canbus obd ecu simulator mobydic1610*, Accessed: October 2019. [Online]. Available: `https://www.ozenelektronik.com/canbus-obd-ecu-simulator-p.html`.

[73] *Oe91c1610 datasheet*, Ozen Elektronik.

[74] *Carloop*, Accessed: October 2019. [Online]. Available: `https://www.carloop.io/`.

[75] *Redbear duo*, Accessed: October 2019. [Online]. Available: `https://github.com/redbear/Duo`.

[76] *Carro dentro de uma curva*, Accessed: November 2020. [Online]. Available: `http://www.ebanataw.com.br/trafegando/concdc.htm`.

[77] *Lin bus explained - a simple intro (2021)*, Accessed: December 2020. [Online]. Available: `https://www.csselectronics.com/screen/page/lin-bus-protocol-intro-basics/language/en`.

[78] *The lin interface and automotive interconnects — a perfect match*, Accessed: December 2020. [Online]. Available: https://www.electronicdesign.com/markets/automotive/article/21806595/the-lin-interface-and-automotive-interconnectsa-perfect-match.