FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Development of wireless telemetry system for FEUP's Formula Student vehicle

Tomás Azenha Oliveira Fernandes Costa

Master in Electrical and Computer Engineering

Supervisor: Paulo José Lopes Machado Portugal

November 1, 2024

Resumo

Esta dissertação apresenta o desenvolvimento de um sistema de telemetria com interface de utilizador (UI) desenhado para a equipa Formula Student (FS) da Faculdade de Engenharia da Universidade do Porto (FEUP). O projeto insere-se no contexto mais amplo da inovação dos veículos elétricos (VE), enfatizando a importância da redução das emissões de dióxido de carbono e do avanço da tecnologia EV. As competições FS servem como uma plataforma crítica para testar e refinar novas ideias para melhorar a eficiência dos VE, a velocidade de carregamento, a autonomia e a acessibilidade.

O sistema de telemetria desenvolvido neste estudo visa dar resposta a desafios específicos enfrentados pela equipa do FS FEUP, nomeadamente a impossibilidade de acesso a dados em tempo real durante as competições. O novo sistema integra um design modular para manutenção e escalabilidade, empregando diversas formas de visualização de dados para apresentar informação de forma eficaz. Suporta comunicação remota em tempo real, lidando com distâncias até 1000 metros e utiliza protocolos de comunicação robustos para garantir uma transmissão fiável de dados.

O sistema é composto por vários componentes principais. Um emissor baseado em LoRa trata da comunicação de longo alcance, transmitindo dados do veículo para uma gateway. A gateway processa estes dados e encaminha-os utilizando o protocolo MQTT, garantindo uma transferência de dados eficiente e fiável para uma máquina virtual na nuvem. Esta máquina virtual na nuvem serve como uma base de dados centralizada para armazenar e gerir todos os dados de telemetria de forma segura. A interface do utilizador fornece uma plataforma intuitiva para visualização e análise de dados em tempo real.

Ao integrar estes componentes de forma eficaz, o sistema de telemetria permite à equipa FS FEUP monitorizar o desempenho do veículo em tempo real, facilitando decisões baseadas em dados que melhoram a eficiência e o desempenho dos veículos.

Abstract

This dissertation presents the development of a telemetry system with a user interface (UI) designed for the Formula Student (FS) team at Faculdade de Engenharia da Universidade do Porto (FEUP). The project is set within the broader context of electric vehicle (EV) innovation, emphasizing the importance of reducing carbon dioxide emissions and advancing EV technology. FS competitions serve as a critical platform for testing and refining new ideas to enhance EV efficiency, charging speed, autonomy, and affordability.

The telemetry system developed in this study aims to address specific challenges faced by the FS FEUP team, particularly the inability to access real-time data during competitions. The new system integrates a modular design for maintainability and scalability, employing various forms of data visualization to present information effectively. It supports real-time remote communication, handling distances up to 1000 meters, and uses robust communication protocols to ensure reliable data transmission.

The system comprises several key components. A LoRa-based sender handles long-range communication by transmitting data from the vehicle to a gateway. The gateway processes this data and forwards it using the MQTT protocol, ensuring efficient and reliable data transfer to a cloud virtual machine. This cloud virtual machine serves as a centralized database for storing and managing all telemetry data securely. The user interface provides an intuitive platform for real-time data visualization and analysis.

By integrating these components effectively, the telemetry system enables the FS FEUP team to monitor vehicle performance in real-time, facilitating data-driven decisions that enhance vehicle efficiency and performance.

Sustainable Development Goals: United Nations

This dissertation on developing a telemetry system with a user interface (UI) for the Formula Student (FS) team at Faculdade de Engenharia da Universidade do Porto (FEUP) addresses several United Nations' Sustainable Development Goals (SDGs) directly and indirectly. By enhancing the efficiency and performance of electric vehicles (EVs), this work primarily aligns with Goals 7, 9, and 13.

Goal 7, *Affordable and Clean Energy*, is directly supported by the telemetry system as it aims to make EVs more efficient, promoting the use of clean and renewable energy sources. The dissertation's contribution to this goal includes optimizing energy consumption in EVs, which supports the broader objective of ensuring access to affordable, reliable, sustainable, and modern energy for all.

In the context of Goal 9, *Industry, Innovation, and Infrastructure*, the telemetry system represents a significant technological advancement. By developing a system that monitors and enhances the performance of EVs, the dissertation fosters innovation and supports the creation of resilient infrastructure. This aligns with specific targets such as upgrading technological capabilities and fostering sustainable industrialization.

Lastly, Goal 13, *Climate Action*, is significantly impacted by the dissertation. By improving the efficiency and performance of EVs, the telemetry system helps in reducing greenhouse gas emissions from the transportation sector. This directly contributes to efforts to combat climate change and mitigate its impacts, supporting global climate action initiatives.

The telemetry system dissertation significantly contributes to improving the quality of life, environmental protection, and social equality. By enhancing the efficiency of EVs, it promotes cleaner air and reduces dependence on fossil fuels, contributing to environmental protection. The dissertation also supports social equality by fostering innovation in sustainable transport, making it more accessible and affordable. The positive changes generated by the dissertation include reduced greenhouse gas emissions, optimized resource use, and the promotion of sustainable urban transport systems. These outcomes not only support the identified SDGs but also contribute to the broader agenda of sustainable development and climate resilience.

Agradecimentos

Gostaria de expressar a minha sincera gratidão a todas as pessoas e instituições que contribuíram para a realização desta dissertação.

Primeiramente, ao meu orientador, professor Paulo José Lopes Machado Portugal, pela orientação e auxílio nos problemas encarados e por toda a honestidade ao longo de todo este processo.

À Xelerate.tech, pela oportunidade de realizar a minha dissertação na empresa e por toda a ajuda que me deram nos diversos desafios que encarei.

Ao meu orientador na empresa, João Victor Peroni de Almeida, por estar sempre disponível para esclarecer as minhas dúvidas e por todo o conhecimento que me transmitiu.

Aos meus familiares: ao meu Pai, à minha Mãe, à minha Irmã, ao meu Avô Miguel e à minha Avó Odete, por estarem sempre ao meu lado, não apenas durante a realização desta dissertação, mas ao longo de todo o meu percurso académico, apoiando-me nos bons e nos maus momentos.

Aos meus amigos, pelo apoio e pela amizade. Em especial, ao Tomás Dias, pela paciência e pelo tempo dedicado à ajuda na realização dos testes ao sistema.

A todos, o meu muito obrigado.

Tomás Azenha Oliveira Fernandes Costa

"The greatest glory in living lies not in never falling,
but in rising every time we fall."

Nelson Mandela

Contents

List of Figures

List of Tables

Abbreviations and Symbols

UI	User Interface
FS	Formula Student
FEUP	Faculty of Engineering of the University of Porto
EV	Electric Vehicle
IoT	Internet of Things
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
CAN	Controller Area Network
ECU	Electronic Control Unit
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
SD	Secure Digital
LPWAN	Low Power Wide Area Network
BLE	Bluetooth Low Energy
ISM	Industrial, Scientific, and Medical
FFD	Full Function Device
RFD	Reduced Function Device
MAC	Medium Access Control
APS	Application Support Sub-layer
ZDO	Zigbee Device Object
HVAC	Heating, Ventilation, and Air Conditioning
DBPSK	Differential Binary Phase Shift Keying
GFSK	Gaussian Frequency Shift Keying
CSS	Chirp Spread Spectrum
SF	Spreading Factor
BW	Bandwidth
MQTT	Message Queuing Telemetry Transport
CoAP	Constrained Application Protocol
XMPP	Extensible Messaging and Presence Protocol
AMQP	Advanced Message Queuing Protocol
QoS	Quality of Service
M2M	Machine to Machine
UDP	User Datagram Protocol
XML	Extensible Markup Language

UX	User Experience
DOM	Document Object Model
SVG	Scalable Vector Graphics
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
CRC	Cyclic Redundancy Check
ToA	Time on Air
VM	Virtual Machine
HTTP	HyperText Transfer Protocol
API	Application Programming Interface
CSV	Comma-Separated Values
JSON	JavaScript Object Notation
RSSI	Received Signal Strength Indicator
GCP	Google Cloud Platform
SQL	Structured Query Language
AJAX	Asynchronous JavaScript and XML

Chapter 1

Introduction

This chapter introduces the development of a telemetry system with a user interface (UI) for the Formula Student (FS) team from Faculdade de Engenharia da Universidade do Porto (FEUP). It contextualizes this dissertation in the innovation scenario in electric vehicles (EVs), highlighting its relevance and significance. The motivations behind this dissertation are then explained, highlighting the specific needs and challenges the FS FEUP team faces. The chapter also outlines the objectives of the work, detailing the intended outcomes and goals. Finally, it provides an overview of the document structure.

1.1 Context

In recent years, EVs have become more frequent in our daily lives. This transition is a main component of global efforts to reduce carbon dioxide emissions and combat pollution [1]. Countries are embracing EVs to address environmental concerns and as a proactive approach to finding alternatives to fossil fuels [2]. Significant advances in technology and supportive government policies help the rise in the adoption of EVs.

According to the International Energy Agency, EVs sales in major markets like the United States, Europe, and China have seen substantial growth [3]. In 2023, the United States recorded 1.4 million electric car sales, a 40% increase from the previous year, and Europe had 3.2 million sales, a 20% rise from the previous year. China registered 8 million electric car sales (Figure 1.1). These trends indicate a growing demand for EVs, which are becoming more accessible and practical for everyday use.

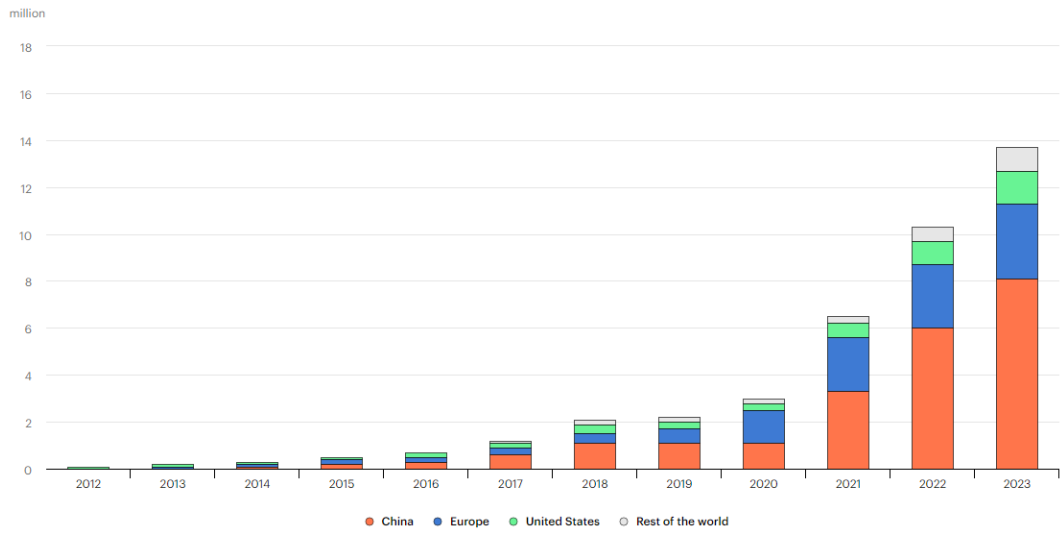

Figure 1.1: Electric car sales, 2012-2023 [3]

As the adoption of EVs continues to rise, the need for innovation and skilled engineers in this field becomes increasingly important. FS competitions are international engineering design contests where university teams design, build, and race small formula-style cars. These competitions are educational platforms that challenge students to apply theoretical knowledge to real-world problems, fostering innovation and practical skills. Teams are judged on various aspects, including design, performance, cost, and presentation [4]. FS competitions play an important role in advancing EV technology by providing engineering students with a platform to apply their knowledge and creativity in a practical environment, pushing the boundaries of what EVs can achieve.

1.2 Motivation

The FS FEUP team currently faces significant challenges due to the lack of a telemetry system. Without this system, accessing real-time data from the vehicle is impossible, as the data is stored on an SD card inside the car. This setup prevents the team from accessing the data while the car is competing, limiting their ability to make timely adjustments. Although the team can still fine-tune the vehicle's parameters between different competitions, it takes more time due to the difficult access to the SD card.

Competing against teams with telemetry systems places the FS FEUP team at a disadvantage. Other teams can use their telemetry systems to gain insights and make rapid adjustments, giving them a competitive edge. The absence of a telemetry system hinders the team's ability to optimize the vehicle's performance and efficiency.

Integrating a telemetry system with a UI into their vehicle would allow the FS FEUP team to collect and analyze vast amounts of real-time data, enabling adjustments to vehicle parameters during tests and races. The UI would serve as a tool for monitoring and analyzing data, providing

insights into various aspects of the vehicle's operation, such as power consumption, component temperature, and engine performance metrics. This real-time monitoring enables teams to make informed decisions, optimizing the efficiency and performance of the car [5].

This dissertation addresses the FS FEUP team's need for a system capable of real-time data visualization. Developing such a system will allow the team to make more efficient data-driven decisions and automate data collection and analysis, reducing the time and effort required for these tasks.

1.3 Objectives

This dissertation aims to provide a comprehensive telemetry system that supports the FS FEUP team in monitoring and analyzing real-time data from their vehicle. This system will help in the decision-making process and enhance the overall performance and efficiency of their operations.

To achieve this goal, specific objectives have been defined together with the FS FEUP team.

Establishing remote communication between the vehicle and the UI to support real-time data visualization is a key objective. The communication system must handle distances up to 1000 meters to ensure it covers the distance of the tracks where the vehicle will compete. It is mandatory to implement one-way communication from the vehicle to the UI to comply with competition rules. It is also indispensable to use a robust communication protocol to ensure information is reliably transmitted and available for analysis without interruptions or inconsistencies.

Another objective is to utilize various forms of data visualization to present information effectively. The UI must show real-time charts of either a single variable or a comparison of two variables. It must also provide the last value sent by the car. This approach is intended to facilitate the understanding of data and relationships between variables, enabling users to understand the evolution and current state of the data.

Additionally, the telemetry system will be designed to be maintainable, scalable, and extensible. This involves decomposing the system into distinct, self-contained modules, each responsible for specific functionalities. This modular design ensures that updates or changes to one module do not impact the entire system, making it easier to maintain and upgrade over time.

By meeting these objectives, the telemetry system will enable the FS FEUP team to make more efficient data-driven decisions and automate data collection and analysis, ultimately reducing the time and effort required for these tasks and enhancing the team's overall performance and efficiency in competitions.

1.4 Document Structure

This document consists of 6 chapters. In addition to chapter 1, the others are organized as follows:

- Chapter 2: Presentation of the fundamental concepts that allow the understanding of the dissertation.
- Chapter 3: Discussion of wireless technologies, Internet of Things (IoT) communication protocols at the application layer, and user interface design, including comparisons between different technologies and their applications.
- Chapter 4: Description of the requirements and architecture of the proposed system, detailing the system components.
- Chapter 5: Detail of the system implementation, including the components of the different subsystems, and the tests carried out and discussion of the results.
- Chapter 6: Presentation of the conclusions of the dissertation and suggestions for future work.

Chapter 2

Background

In order to have a clear understanding of what FS is and its relevance in the current automotive industry, this chapter begins by presenting the advances of the automotive industry, particularly the integration of IoT and the growth of EVs. It then discusses FS in general, followed by an overview of the competitions. Finally, it introduces the FS FEUP team and describes the current state of their data acquisition and processing system.

2.1 Automotive Industry, IoT and Electric Vehicles

A significant transformation is taking place in the automotive industry mostly due to the integration of IoT in vehicles and the growth of EVs. This is turning the industry towards a more efficient and sustainable future by improving vehicle features, safety, and user experience [6, 7].

IoT in the automotive industry consists of a network of devices that communicate and exchange data, improving vehicle operation and management. This data exchange can occur between vehicles, Vehicle-to-Vehicle (V2V), or between vehicles and infrastructures, Vehicle-to-Everything (V2X). These enhancements improve navigation, traffic management, and predictive maintenance [7]. Additionally, IoT solutions enable fleet management by providing data on vehicle location, performance, and maintenance needs, leading to improved efficiency [8]. Vehicles equipped with IoT solutions can also monitor their health and predict potential failures before they occur, reducing downtime and repair costs [6].

The synergy between IoT and EVs creates a new paradigm in the automotive industry where both technologies complement each other. IoT enables smart charging solutions that optimize charging times based on grid demand, ensuring efficient energy use and reducing charging costs. IoT solutions allow for remote monitoring and diagnostics, enabling manufacturers to push software updates and perform maintenance checks remotely. This minimizes the need for service center visits and ensures optimal vehicle performance [9].

2.2 Formula Student

Formula Student is an international engineering competition that challenges students to design, build, and race a single-seat racing car. The competition is structured to evaluate various aspects of the team's capabilities through both dynamic and static events.

The primary objectives of FS are to encourage innovation, hands-on learning, and teamwork among students. Participants gain practical experience in engineering, project management, and problem-solving, preparing them for future careers in the automotive industry and beyond. The competition fosters an environment where students can apply theoretical knowledge in real-world scenarios, promoting a deeper understanding of engineering principles and their applications. [4]

2.2.1 Formula Student Competitions

FS competitions are divided into dynamic and static events, each designed to evaluate different aspects of the car and the team. The points distribution for these events is shown in Figure 2.1.

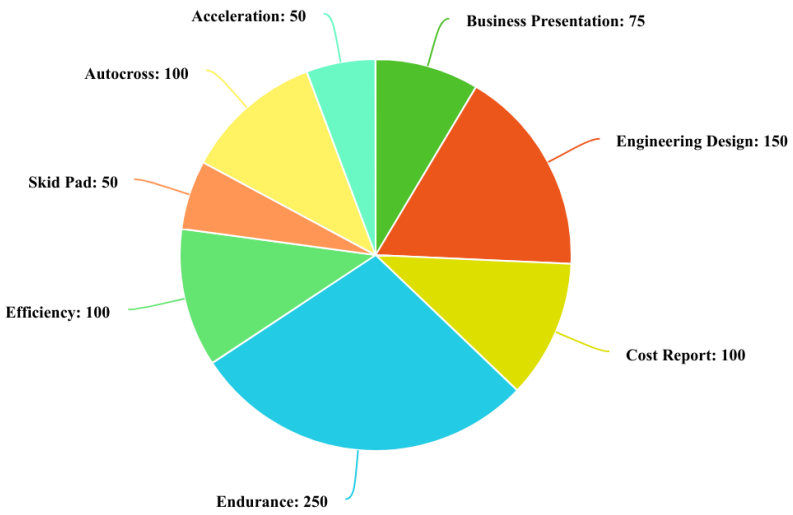

Figure 2.1: Points distribution of events in FS competitions

Dynamic events evaluate the car's performance, while static events assess the team's capabilities. There are five dynamic events, totaling 675 points [10]:

- **Acceleration (75 points):** The car's acceleration in a straight line for 75 meters is evaluated, starting from a standstill.
- **Autocross (100 points):** Tests the car's dynamic ability in a one-lap sprint.
- **Skid Pad (75 points):** Assess the lateral grip of the car on a flat surface while making a constant radius turn.
- **Efficiency (100 points):** During the endurance race, energy consumption is recorded. Therefore, the energy used and lap times determine the efficiency of the car.

- **Endurance (325 points):** covers a distance of 22 kilometers and takes place on a track similar to Autocross. Consistency and reliability are the factors to be evaluated in this event.

There are three static events, totaling 325 points [10]:

- **Business Presentation (75 points):** In this event, every team showcases their business plan for the prototype they have built to a panel of judges who represent a fictional company.
- **Engineering Design (150 points):** In this event, students are required to demonstrate their understanding of the car and engineering principles to a panel of judges.
- **Cost Report (100 points):** Teams produce a report outlining the expenses linked to materials, processes, and the assembly of the vehicle. Evaluation criteria include the overall cost of the vehicle and the quality of the submitted report.

2.3 Formula Student FEUP

Formula Student FEUP is a team of approximately 60 students from FEUP. Their mission is to facilitate students' self-development by allowing them to collaborate on designing a single-seat electric racing car prototype.

The team is composed of students from diverse engineering areas, such as mechanical, electronic, computer science, and other related fields [10].

The FS FEUP team is divided into 8 departments, each responsible for specific aspects of the car or the team's operation [11]:

- **Powertrain:** Manages the Tractive System Accumulator, Transmission, and Cooling subsystems.
- **Chassis & Aerodynamics:** Oversees the Chassis, Aerodynamics, and Driver's Interface subsystems.
- **Vehicle Dynamics:** Handles Suspension Geometry and Kinematics, Control, Lap Time Simulation, and Testing subsystems.
- **Suspension:** Covers Wheel Assembly, Vibrations, Wishbones, Braking, and Steering subsystems.
- **Autonomous System:** Focuses on State Estimation, Planning and Control, Perception, Simulation, and Infrastructure subsystems.
- **Electronics & Software:** Takes care of Low Voltage, Software, Tractive System, and Tractive System Accumulator subsystems.
- **Research & Development:** Develops the infrastructure to link students, academia, and companies by supporting Master's and Bachelor's Thesis projects.

- **Operations:** Manages Sponsors, Marketing, and Logistics.

FSFEUP01 (Figure 2.2) was the team's first-ever prototype and was used between 2021 and 2023. During this period they participated in renowned events related to Formula Student and were able to collect 6 awards. Building on their initial experiences and success with FSFEUP01, the team is now in the 2023-2025 cycle focusing on the development of FSFEUP02. The objective will be to develop an autonomous driving prototype.

Figure 2.2: FSFEUP01

2.4 Integration of IoT in FS FEUP Prototype

Advances in the automotive industry, particularly in the area of EVs and the integration of IoT in vehicles, are not limited to just commercial vehicle manufacturers. These advances are also present in academic and competitive environments, such as FS competitions.

The FS FEUP team is an example of a team that wants to integrate these technological advances into their prototype. The integration will enable a more advanced and efficient data acquisition and processing system.

The current data acquisition and processing system used by the FS FEUP is not optimized. The car is equipped with various sensors that monitor parameters such as temperatures, voltages, and currents. These sensors are connected to the Controller Area Network (CAN) bus to gather the data. The CAN bus is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other.

One Electronic Control Unit (ECU) of the car communicates with the CAN bus via a CAN to Universal Asynchronous Receiver / Transmitter (UART) transceiver, allowing it to read the values from the sensors present on the CAN bus. Through Serial Peripheral Interface (SPI) communication, the ECU writes the collected data to a Secure Digital (SD) card in a comma-separated values (CSV) file. This ensures that all data is captured and preserved for post-session analysis. After a practice session or competition event, the SD card is removed from the car, and the raw data is processed using a Python script.

To solve the issues of manual data processing and lack of real-time data visualization, the proposed solution will involve the development of an improved telemetry system that allows real-time data processing and visualization. The main features of the system will include wireless data transmission, real-time data processing, and a user interface for real-time data visualization.

2.5 Conclusions

In summary, the integration of IoT and advancements in EV technology are not confined to commercial automotive manufacturers but are also making significant impacts in academic and competitive environments such as FS competitions. The FS FEUP team exemplifies the application of these cutting-edge technologies in their efforts to develop an efficient data acquisition and processing system for their prototype.

Currently, the FS FEUP team's data acquisition and processing system is not optimized for real-time data visualization and analysis, limiting the team's ability to make immediate data-driven decisions. To address these limitations, the proposed solution is an improved telemetry system that enables wireless data transmission, real-time data processing, and a user interface for real-time visualization. These enhancements are essential for improving the team's efficiency and performance during practice sessions and competitions.

These characteristics will be discussed in the next chapters, where different technologies will be presented and a decision will be made about the most appropriate technologies.

Chapter 3

State of the art

This chapter will start with an overview of different wireless technologies, which are potential solutions to be used in resolving the problem. It will also present the communication protocols used in telemetry systems. Finally, there will be an explanation of the guiding principles to be followed when developing a UI and the technologies that can be utilized.

3.1 Wireless Technologies

Wireless technologies facilitate data transfer across various environments and distances. Some wireless technologies, like Low Power Wide Area Networks (LPWAN), are optimized for long-range communication over several kilometers. In contrast, other technologies like Zigbee and Bluetooth Low Energy (BLE) are designed to provide efficient data transfer with low power consumption within shorter ranges. However, this focus on short-range capabilities can be limited unless a mesh topology is employed. Mesh networks can extend the effective range of these technologies by enabling devices to relay data to one another, thereby overcoming distance limitations.[12]

3.1.1 Zigbee

Zigbee is a wireless communication technology designed for low-power and low-cost devices based on the IEEE 802.15.4 standard for low-rate wireless personal area networks (LR-WPAN). It operates in the 2.4 GHz, 868 MHz, and 915 MHz industrial, scientific, and medical (ISM) frequency bands, offering data transmission rates up to 250 kbps, depending on the frequency band being used and environmental conditions [13, 14].

3.1.1.1 Device Types

Zigbee devices are categorized into three types: application devices, logical devices, and physical devices [15]. Application devices include various sensors and actuators that collect information from the surrounding environment and supply data to the network coordinator [15]. These devices

are used in applications such as home automation [16], industrial control [17], and smart energy management [18].

Logical Devices consist of three types: coordinators, routers, and end devices. Coordinators are the main devices responsible for network formation, security key distribution, and overall network management. There is exactly one coordinator in each Zigbee network. Routers extend the network range by passing data between devices and help expand the network coverage while maintaining robust communication paths. End devices are typically low-power nodes that communicate with a parent router or coordinator. They do not route data for other devices and are often battery-powered, making them ideal for sensors and actuators in various applications. This classification is based on the functional roles that devices play within the network. It is important to note that a single device may fulfill multiple roles or belong to multiple categories depending on its specific configuration and functionality [15].

Physical Devices come in two types: Full Function Devices (FFD) and Reduced Function Devices (RFD). FFDs are capable of performing all network-related functions and can be used as a coordinator, router, or end device. RFDs are limited in function and are typically used as end devices. They are often battery-operated sensors that collect data and communicate with routers or coordinators [15].

3.1.1.2 Network Topology

Zigbee supports multiple network topologies, including star, tree, and mesh (Figure 3.1). In a star topology, all nodes are connected directly to a central coordinator. This setup simplifies the management, making it suitable for smaller networks where centralized control is preferred. However, the failure of the central coordinator can disrupt the entire network. A tree topology features a central coordinator at the root, with routers and end devices forming branches and leaves. It balances simplicity and robustness and is useful for extending network coverage. In a mesh topology, each node can communicate with multiple other nodes, creating a self-healing network that reroutes data if a node fails or becomes unreliable. This is particularly useful in environments with common signal obstructions and enhances the network's reliability and range. [15]

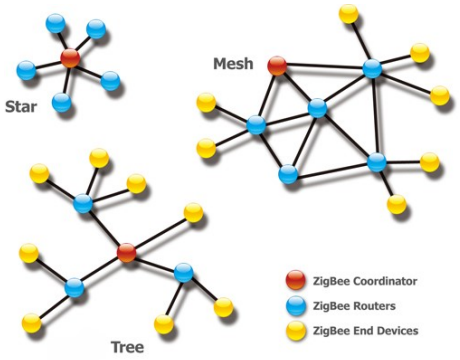

Figure 3.1: Zigbee Topologies [19]

3.1.1.3 Protocol Stack

The Zigbee protocol stack (Figure 3.2) is built on top of the IEEE 802.15.4 standard, defining the physical and MAC layers. The Zigbee Alliance extends this with the network and application layers, adding features such as network management, security, and application support.

The Physical layer handles hardware initialization, channel selection, and energy estimation. Zigbee supports three frequency bands: 2.4 GHz (worldwide), at 250 kbps, 915 MHz (America) at 40 kbps, and 868 MHz (Europe) at 20 kbps. The MAC layer offers data and management services, supporting beacon and non-beacon enabled networks. It controls frame structure for data transmission, acknowledgment, and command frames. The Network layer performs several network-related tasks, such as starting a new network, joining and leaving a network, and addressing new devices. [15, 20]

The Application layer includes the Application Support (APS) sub-layer, which handles cryptography keys and ensures secure communication. It manages key establishment, device addition and removal, and security updates. Another component is the Application Framework, which provides the environment in which Zigbee applications operate. It defines the structure and behavior of application objects, allowing for the creation of diverse applications. The Zigbee Device Object (ZDO) is another part of the application layer. The ZDO is responsible for device management functions, including starting and joining networks, discovering other devices, and configuring security. It provides a standardized interface for interacting with Zigbee devices and ensures that devices can communicate and work together within the network [15, 20].

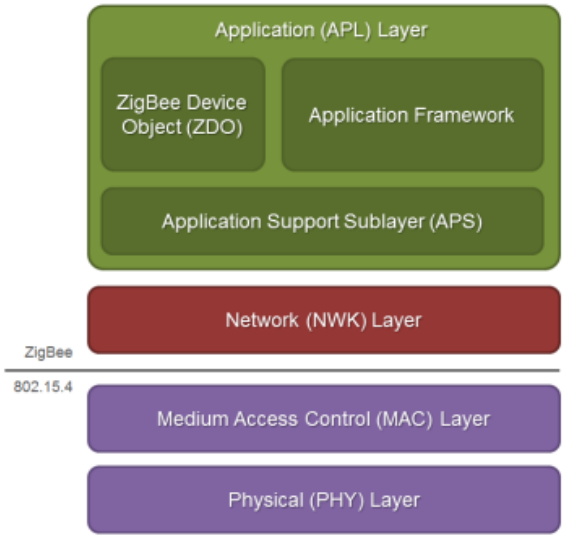

Figure 3.2: Zigbee Protocol Stack [21]

3.1.1.4 Applications

Zigbee technology is widely used in home automation, industrial control, and smart energy management. In home automation, Zigbee enables wireless control and monitoring of lighting, heat-

ing, ventilation and air conditioning (HVAC), security systems, and other household devices. The technology's low power consumption and adequate data rates make it ideal for battery-operated devices in a smart home environment [16].

In industrial environments, Zigbee is used for monitoring and control systems. Its robust mesh network capabilities ensure reliable communication in complex industrial environments, making it suitable for applications like process monitoring, asset tracking, and predictive maintenance [17].

For smart energy management, Zigbee networks manage and optimize energy consumption in homes and commercial buildings, supporting applications like smart meters, in-home displays, and load control devices, contributing to energy efficiency and cost savings [18].

3.1.1.5 Limitations and Challenges

Despite its advantages, Zigbee has some limitations. The range is limited to 100 meters, which may not be sufficient for applications requiring wider coverage. However, this limitation can be mitigated by employing a mesh topology combined with routers, which can significantly extend the effective network range by allowing devices to relay data through multiple nodes. [16]. Additionally, operating in the 2.4 GHz band, Zigbee can face interference from other devices using the same frequency, such as Wi-Fi and Bluetooth, which can affect performance [20].

3.1.2 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a wireless communication technology designed for devices with low power consumption. BLE has been part of the Bluetooth specification since version 4.0, released in December 2009, focusing on energy efficiency [22]. Like Bluetooth, it operates in the 2.4 GHz ISM band, but with significant differences in data handling and power consumption [23].

3.1.2.1 Device Types

BLE devices are categorized into different roles based on their function in the network: central, peripheral, broadcaster, and observer. The central device scans for peripheral devices, and initiates and manages multiple connections. Peripheral devices advertise their presence and accept connections from central devices. They are often battery-operated and designed to be energy-efficient. A broadcaster is a device that sends advertising packets but does not receive any connections. An observer scans for advertising packets from broadcasters but does not initiate connections [23].

3.1.2.2 Network Topology

BLE supports multiple network topologies, including point-to-point, broadcast, and mesh (Figure 3.3). In a point-to-point topology, a peripheral device connects to a central device. This setup is simple and suitable for small networks. However, if the central device fails, the network communication is disrupted. In the data broadcast topology, a broadcaster sends data to many observers, which is useful for applications such as beacons and proximity services where unidirectional data

is sufficient. The mesh topology allows devices to relay messages to each other, creating a robust network with multiple pathways for data transmission. [24].

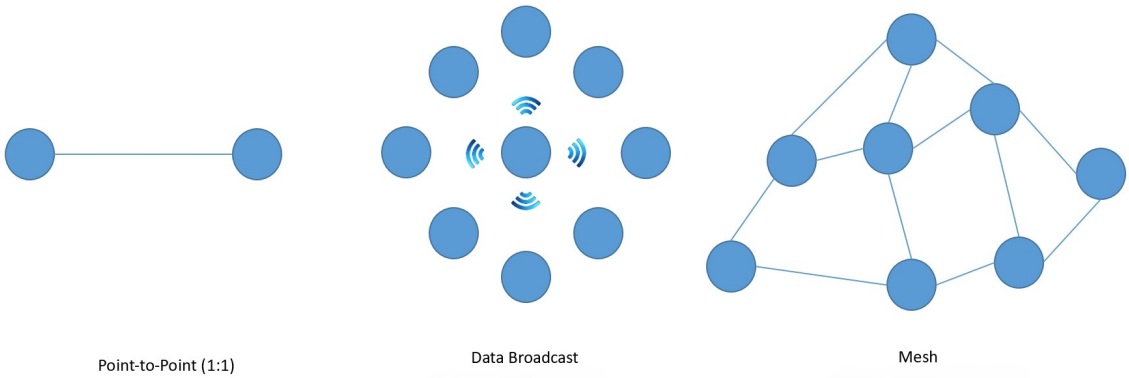

Figure 3.3: BLE Topologies [25]

3.1.2.3 Protocol Stack

The BLE protocol stack is divided into two main parts: the Controller and the Host, each responsible for different layers and functions of the stack (Figure 3.4).

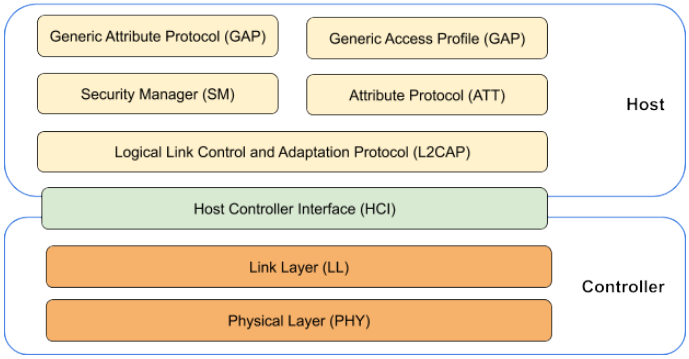

Figure 3.4: BLE Protocol Stack [26]

The controller is responsible for the low layers of the BLE stack: The Physical layer and the Link layer. The Physical layer handles the modulation and demodulation of radio signals. BLE uses Gaussian Frequency-Shift Keying (GFSK) modulation and operates in the 2.4 GHz ISM band. For versions before Bluetooth 5, the maximum data rate is 1 Mbps, while Bluetooth 5 and later versions support data rates of up to 2 Mbps [27]. The link layer manages the radio and defines procedures for advertising, scanning, connection setup, and data transfer.

The host is responsible for the higher layers of the BLE stack: Logical Link Control and Adaptation Protocol (L2CAP), Attribute Protocol (ATT), Generic Attribute Profile (GATT), Security Manager (SM), and Generic Access Profile (GAP) [28].

The L2CAP layer multiplexes data from different higher-layer protocols, handles packet segmentation and reassembly, and manages flow control. The ATT and GATT define how data is structured and accessed on BLE devices. ATT manages the communication of attribute data, while GATT defines the service and characteristic profiles used in applications. The SM provides methods for pairing devices, generating and distributing security keys, and encrypting and authenticating data. The GAP defines how BLE devices discover and connect and how they broadcast data. It includes the roles of broadcaster, observer, peripheral, and central, and specifies the procedures for establishing connections [23].

The Host Controller Interface (HCI) acts as a middle layer that allows communication between the host (software stack) and the controller (hardware), enabling the host to send commands to the controller and receive events and data.

3.1.2.4 Applications

BLE technology is widely used in wearable devices and smart home devices. Its low power consumption makes it ideal for wearable devices such as fitness trackers, smartwatches, and health monitors. BLE is also utilized in smart home applications, including lighting control, thermostats, door locks, and security systems [29].

3.1.2.5 Limitations and Challenges

Although BLE offers several advantages, it also has limitations. Its range can extend up to 200 meters outdoors, but the indoor range is often limited due to physical obstructions and interference. Additionally, operating in the 2.4 GHz band, BLE can face interference from other devices using the same frequency, such as Wi-Fi and Bluetooth devices, which can affect performance [23].

3.1.3 Sigfox

Sigfox is a proprietary Low Power Wide Area Network (LPWAN) technology that operates in unlicensed ISM bands and focuses on ultra-narrowband communication. It is designed to provide simple, cost-effective connectivity for low-data rate applications. Sigfox is also designed for long-range communication, typically covering distances of up to 10 kilometers in urban areas and up to 40 kilometers in open areas. This extensive range is achieved through the use of ultra-narrowband signals, which are less susceptible to interference and can travel longer distances than wider-band signals [30].

3.1.3.1 Network Topology

Sigfox supports a simple star topology, where all Sigfox devices connect directly to a base station. This setup simplifies network management and is suitable for wide-area coverage. However, if a base station fails, the communication for all connected devices is disrupted [30].

3.1.3.2 Protocol Stack

The Sigfox protocol stack is relatively simple, emphasizing efficiency and minimalism. The stack can be divided into four layers, each responsible for different aspects of communication: Physical layer, MAC layer, Network layer, and Application layer.

The Physical layer uses Differential Binary Phase-Shift Keying (DBPSK) modulation for uplink communication and Gaussian Frequency-Shift Keying (GFSK) for downlink communication. Operates in the sub-GHz unlicensed ISM bands (433 MHz in Asia, 868 MHz in Europe, and 915 MHz in North America), supporting a data rate of up to 100 bps. The MAC layer manages data packet structure and channel access. Sigfox employs a simple random access method to communicate with base stations [30, 31].

In Sigfox's architecture, the Network layer's function is closely integrated with the MAC layer and manages the routing of data from devices to the backend servers through base stations, which are operated by network providers. This means that the network infrastructure, including base stations, is typically managed by the Sigfox network operator. The Application Layer defines the format of the data transmitted and ensures its proper delivery to the backend servers. It includes encryption and authentication mechanisms to protect the data [30, 31].

3.1.3.3 Applications

Sigfox technology is widely used in various applications. In logistics and supply chain management, Sigfox's long-range tracking capabilities allow for real-time monitoring of goods and assets, improving supply chain visibility and efficiency. For energy management, Sigfox facilitates the monitoring and management of energy consumption in buildings and industrial facilities, contributing to energy efficiency and sustainability efforts. In smart cities, Sigfox enables various applications such as waste management, smart parking, air quality monitoring, and street lighting, enhancing urban living conditions and sustainability [32].

3.1.3.4 Limitations and Challenges

Although Sigfox offers several advantages, it also has limitations. The low data rate of up to 100 bps may restrict its use in applications requiring higher data throughput. Sigfox allows a maximum of 140 uplink messages per device per day, with each message being up to 12 bytes, and 4 downlink messages per device per day, each up to 8 bytes [30].

3.1.4 LoRa

LoRa is an LPWAN developed by Cycleo (later acquired by Semtech) and standardized by the Lora Alliance. It uses chirp spread spectrum (CSS) modulation to achieve long-range communication with low power consumption. CSS is a wideband linear frequency modulation in which carrier frequency varies for the defined extent of time. LoRa serves as the physical layer of the LoRaWAN protocol, which includes additional layers for network management and application support. LoRa

is designed for long-range communication, typically covering distances of up to 5 kilometers in urban areas and 15 kilometers in open areas. This extensive range is achieved through the use of CSS modulation, which provides robustness against interference and allows signals to travel long distances with low power consumption [33].

3.1.4.1 Network Topology

LoRa networks typically work over a "Star-of-Stars" topology. In this topology, LoRa end devices communicate directly with gateways. These gateways then forward the data to a central network server. This setup simplifies network management and is suitable for wide-area coverage. Each gateway can handle thousands of devices, and the network server manages the overall network, including device authentication and data deduplication (elimination of redundant data) [34].

3.1.4.2 Protocol Stack

LoRa is specifically focused on the physical layer. It uses CSS modulation, which provides robustness against interference and allows long-range communication. Operates in the sub-GHz unlicensed ISM bands (433 MHz in Asia, 868 MHz in Europe, and 915 MHz in North America). This modulation technique spreads the signal over a wide frequency range, making it resilient to noise and capable of achieving long-distance communication with low power consumption. LoRa supports data rates from 0.3 kbps to 50 kbps, depending on the spreading factor (SF) and bandwidth used. The SF ranges from 7 to 12, with higher spreading factors providing a longer range but lower data rates, and lower spreading factors providing higher data rates but shorter range. Bandwidth options typically include 125 kHz, 250 kHz, and 500 kHz, with wider bandwidths allowing for higher data rates at the expense of reduced range [33].

3.1.4.3 Applications

LoRa technology is widely used in various applications. Smart agriculture and environment applications, enhance agricultural practices by monitoring soil moisture, weather conditions, and livestock, improving resource management and crop yields. Environmental applications include tracking air quality and water levels and contributing to environmental conservation efforts [35].

In smart cities and buildings, LoRa facilitates urban planning and management through applications like smart street lighting, waste management, and air quality monitoring. In buildings, LoRa integrates IoT solutions for energy management, security, and building automation, contributing to energy efficiency and safety [35].

For logistics and supply chain management, LoRa provides real-time tracking of goods and assets, optimizing supply chain operations and improving efficiency. This includes monitoring the condition and location of shipments and ensuring timely and secure deliveries [35].

In industrial IoT, LoRa supports applications such as predictive maintenance, equipment monitoring, and factory automation, enhancing operational efficiency, reducing downtime, and improving overall productivity in manufacturing and industrial environments [35].

3.1.4.4 Limitations and Challenges

Although LoRa offers several advantages, it also has limitations. One limitation is its lower data rates compared to other wireless communication standards, such as cellular networks and Wi-Fi, which may restrict its use in applications requiring high data throughput. Additionally, while LoRa operates in unlicensed bands, which can sometimes lead to interference from other devices, the impact on performance varies. This is because the effect of interference depends on factors such as the specific signal modulation used and the density of other devices operating in the same frequency range.

In Europe, LoRa devices operating in the 868 MHz ISM band are subject to regulatory limitations such as duty cycle restrictions. These restrictions, which limit transmission to 1% of the time within a given hour (36 seconds per hour), are designed to minimize interference and ensure fair spectrum usage. While these limitations ensure fair usage of the spectrum and minimize interference, can also limit the number of messages that can be sent, affecting applications requiring frequent data transmissions or continuous connectivity [36].

3.1.5 Comparison of Wireless Technologies

In the vast landscape of wireless communication technologies, many solutions are available beyond those discussed here. However, for this dissertation, we focus on summarizing and comparing the most important technologies in terms of their practical applications.

The table below provides a comparison between Zigbee, BLE, Sigfox, and LoRa, focusing on key aspects such as frequency bands, data rates, and transmission distances for urban and open areas. These details are critical as they directly impact the selection of the wireless technology to be used in this dissertation.

Technology	Frequency Bands	Data Rate	Urban Range	Open Range
Zigbee	2.4 GHz, 868 MHz, 915 MHz	20-250 kbps	100 meters	100 meters
Bluetooth Low Energy (BLE)	2.4 GHz	2 Mbps	200 meters	200 meters
Sigfox	868 MHz, 902-928 MHz	100 bps	10 km	40 km
LoRa	868 MHz, 915 MHz	0.3-50 kbps	5 km	15 km

Table 3.1: Comparison of Wireless Technologies [15, 23, 33]

3.2 IoT Communication Protocols at the Application Layer

Having reviewed various wireless technologies and their characteristics, it is important to explore the communication protocols used at the application layer. These protocols define how data is structured, exchanged, and interpreted once it reaches its destination.

In the context of IoT telemetry communications, application layer protocols are commonly employed to facilitate effective communication between devices and receiving systems. The most common are [37]:

- **MQTT (Message Queuing Telemetry Transport):** is a lightweight publish/subscribe messaging protocol optimized for constrained environments like IoT. Its efficiency in low-bandwidth, high-latency, or unreliable networks makes it ideal for telemetry applications. MQTT supports Quality of Service (QoS) levels to ensure message delivery and has broad industry adoption. However, it may have higher overhead compared to more minimalistic protocols in highly constrained environments, and it has limited support for resource discovery [38].

- **CoAP (Constrained Application Protocol):** is designed for Machine-to-Machine (M2M) communication in constrained environments. It employs a RESTful architecture that facilitates integration with web-based systems and operates efficiently over User Datagram Protocol (UDP), minimizing header overhead. CoAP's design is well-suited for IoT devices with limited resources. Despite its strengths, CoAP is less widely adopted compared to MQTT and might be less effective in non-constrained environments [38].

- **XMPP (Extensible Message and Presence Protocol):** is an open Extensible Markup Language (XML) technology for real-time communication, commonly used for instant messaging and collaboration. It features a presence mechanism to indicate availability for communication and supports efficient real-time messaging. However, the XML format can be verbose, leading to increased bandwidth usage, and XMPP may be more complex for devices with limited resources [38].

- **AMQP (Advanced Message Queuing Protocol):** is a messaging protocol designed for reliable and asynchronous communication. It supports multiple messaging patterns, including publish/subscribe and request/response, making it suitable for complex message-oriented middleware scenarios. While AMQP offers advanced features and flexibility, it is more complex to implement and configure than simpler protocols and generally has higher resource requirements [38].

To better understand the strengths and limitations of each protocol, the following table summarizes the key advantages and disadvantages of MQTT, CoAP, XMPP, and AMQP in the context of IoT application layer communication:

Protocol	Advantages	Disadvantages
MQTT	Lightweight publish/subscribe model. Efficient in low-bandwidth, high-latency, or unreliable networks. QoS levels for message delivery. Wide industry adoption.	May have higher overhead compared to extremely lightweight protocols in constrained environments. Limited resource discovery.
CoAP	Designed for constrained environments (IoT devices). RESTful architecture. Low header overhead. Efficient for UDP communication.	Less widely adopted compared to MQTT. May not be as suitable for non-constrained environments.
XMPP	Presence-based communication. Real-time messaging. XML for data exchange. Suitable for instant messaging.	XML format can be verbose and may lead to increased bandwidth usage. May be more complex for resource-constrained devices.
AMQP	Advanced Message Queuing Protocol with multiple exchange types. Supports various messaging patterns (publish/subscribe, request/response).	More complex to implement and configure compared to simpler protocols. May have higher resource requirements.

Table 3.2: Advantages and Disadvantages of MQTT, CoAP, XMPP, and AMQP [38]

3.3 User Interface Design

In the field of UI design, creating effective dashboards for telemetry systems involves adhering to several well-established principles that enhance usability, clarity, and user satisfaction. Telemetry systems, which collect and transmit data from remote sources, require dashboards that can present complex data succinctly and meaningfully.

Dashboards should prioritize clear and straightforward presentation of data, focusing on essential information and minimizing unnecessary complexity. A clean layout and straightforward navigation are very important components (Figure 3.5). This approach aligns with usability principles that emphasize minimalist design and aesthetic integrity [39].

Figure 3.5: Dashboard with clean layout [40]

Consistency in design elements such as colors, typography, icons, and layout helps users understand and predict interface behavior, reducing cognitive load. Maintaining a uniform style throughout the dashboard ensures that similar functions are easily recognizable and usable (Figure 3.6). Interaction design principles highlight that consistency in design contributes to a more intuitive user experience [41].

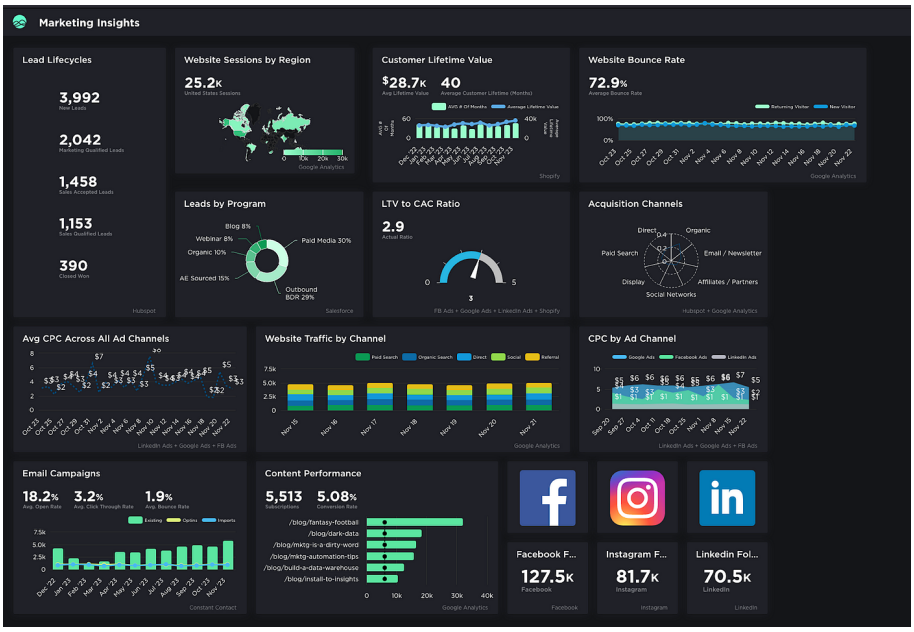

Figure 3.6: Dashboard with consistent use of colors and formats across different sections [42]

Understanding the users and their needs is fundamental in user-centered design. Telemetry dashboards should be designed with the end-users in mind, involving them in the design process through user testing and feedback. This approach ensures the dashboard meets their needs and preferences, leading to higher satisfaction and better usability. User-centered design practices are widely defended in User Interface/User Experience (UI/UX) literature [43].

Dashboards must be scalable to handle increasing amounts of data and flexible to adapt to different user requirements. This includes designing responsive interfaces that work well on various devices and screen sizes (Figure 3.7). Scalability and flexibility are key considerations in modern UI design, ensuring that dashboards remain functional and effective as needs evolve [44].

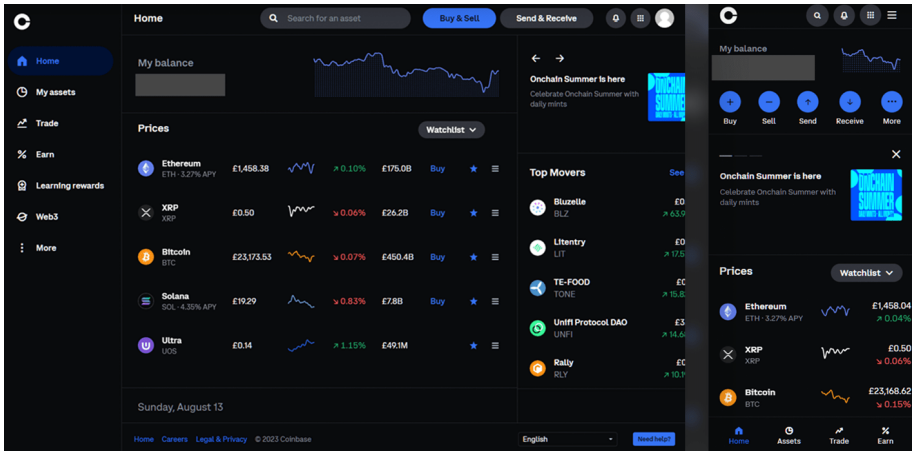

Figure 3.7: Comparison of a dashboard on different devices [45]

3.3.1 Visual Perception and Interactions

Visual perception and interactions are also fundamental to the effectiveness of telemetry dashboards. Understanding how users perceive and interact with visual elements can significantly enhance the usability and functionality of the interface.

Certain principles describe how users naturally perceive and organize visual information. For instance, proximity refers to the tendency to perceive elements that are close together as related. Similarity indicates that users group similar elements. Enclosure is the principle where elements enclosed within a boundary are perceived as a cohesive unit. Closure describes the tendency to see incomplete shapes as complete. Continuity involves favoring continuous lines and patterns, while connection refers to the perception of elements connected by lines or visual cues as related. These principles are known as *Gestalt* principles (Figure 3.8, and they help to understand how users perceive visual information [46].

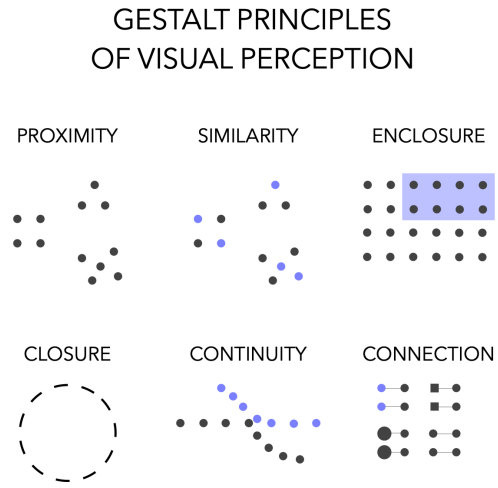

Figure 3.8: Gestalt Principles in Data Visualization [47]

Color theory is essential in UI design for telemetry dashboards, as it affects how users perceive information and interact with the interface. High contrast between text and background improves readability, while color coding can differentiate between data types or status indicators. Colors also have emotional impacts. For example, red can indicate errors or warnings, while green often signifies success or normal operation (Figure 3.9) [48].

Figure 3.9: Color-coded dashboard [42]

3.3.2 Comparison of technologies for building telemetry dashboards

Various technologies can be used to build telemetry dashboards, each with its strengths and weaknesses. Here, it compares some of the most popular options: *React*, *Angular*, *D3.js*, and *Chart.js* for front-end development and data visualization.

React is a popular JavaScript library for building user interfaces, maintained by Facebook. It allows developers to create reusable UI components, making the development process more efficient. *React's* virtual Document Object Model (DOM) improves performance, making it a suitable choice for complex and dynamic dashboards [49].

Angular is a comprehensive framework developed by Google for building web applications. It includes tools for everything from data binding to form validation, making it a full-fledged solution for developing dashboards. *Angular's* dependency injection and modular architecture facilitate large-scale application development. However, its steep learning curve and complexity might be a drawback for smaller projects or less experienced developers [50].

D3.js is a powerful library for creating complex and interactive data visualizations using web standards like Scalable Vector Graphics (SVG), HyperText Markup Language (HTML), and Cascading Style Sheets (CSS). It offers fine-grained control over every aspect of the visualization, making it ideal for custom and advanced visualizations. However, *D3.js* has a steep learning curve and requires a deep understanding of both JavaScript and data visualization principles [51].

Chart.js is a simpler and more beginner-friendly library for creating responsive and customizable charts. It supports a variety of chart types, such as line, bar, pie, and radar charts. While *Chart.js* is easy to use and integrates well with *React* and *Angular*, it lacks the flexibility and customization options that *D3.js* offers. It is best suited for straightforward visualizations that do not require extensive customization [52].

In summary, the choice of technology for building telemetry dashboards depends on the specific needs and constraints of the project. *React* is suitable for dynamic and component-based UIs and *Angular* is for comprehensive and large-scale applications. For data visualization, *D3.js* offers unparalleled flexibility and control, while *Chart.js* provides simplicity and ease of use for standard chart types.

3.4 Conclusions

This chapter has provided a comprehensive overview of the state of the art in wireless technologies and communication protocols used in IoT telemetry systems, as well as guidelines and technologies for developing user interfaces. The analysis of wireless technologies, including Zigbee, BLE, Sigfox, and LoRa, highlights their specific advantages and limitations, offering a foundation for selecting the appropriate technology to be used in this dissertation.

In terms of communication protocols, MQTT, CoAP, XMPP, and AMQP were evaluated for their strengths and weaknesses in the context of IoT applications. This evaluation helps to understand the most suitable protocols for specific telemetry requirements, considering factors like

network reliability, resource constraints, and message delivery guarantees.

The principles of user interface design were discussed, emphasizing clarity, consistency, user-centered design, scalability, and visual perception to enhance the effectiveness of telemetry dashboards. Finally, a comparison of technologies for building telemetry dashboards, such as React, Angular, D3.js, and Chart.js, was presented, providing insights into their suitability based on project needs.

Overall, this chapter establishes a foundation for presenting an effective and efficient solution to the FS FEUP problem by leveraging the right combination of wireless technologies, communication protocols, and user interface design principles and technologies.

Chapter 4

Proposed Solution

This chapter presents a comprehensive solution proposal for the telemetry system. The solution encompasses a detailed overview of the system's requirements, architecture, and key components, providing a clear path for development and implementation. The telemetry system is designed to monitor and transmit vehicle data efficiently, leveraging modern technologies to ensure reliability, scalability, and user-friendliness. This proposal details the functional and non-functional requirements, outlines the system architecture, and explains the operational processes of each component.

4.1 Requirements

This chapter details the requirements established in collaboration with the FS FEUP team, essential for guiding the development of the telemetry system and ensuring its functionality and robustness. The requirements are categorized into functional and non-functional types. Functional requirements specify the features and functionalities of the system, while non-functional requirements define the standards and constraints to ensure the system's reliability, efficiency, and user-friendliness. The following table presents these requirements.

Functional Requirements	
FR1	The telemetry system must transmit data from the car every 10 seconds.
FR2	Communication between the car and the UI must be wireless and unidirectional.
FR3	Data transmitted from the car must be stored in a database.
FR4	The UI must feature four distinct pages: Time Charts, Single Values, Variables Comparison, and Previous Sessions.
FR5	The Time Chart page must display a chart for each variable transmitted.
FR6	The Time Chart page must include a menu to filter which charts to view.
FR7	The Variables Comparison page must display a chart with a horizontal axis for timestamps and two vertical axes to compare two selected variables.
FR8	The Variables Comparison page must include a menu to select variables for display.
FR9	The Single Values page must display the latest value received for each variable.
FR10	The Single Values page must use gauge charts to display the values of the variables.
FR11	The Session Values page must include their own Time Chart and Variables Comparison pages for reviewing previous sessions' data.
FR12	The UI must allow users to upload a file containing data from the car.
Non-Functional Requirements	
NFR1	The telemetry system must maintain communication over a distance of up to 1000 meters.
NFR2	The UI should be intuitive and easy to use.

Table 4.1: Functional and Non-Functional Requirements for the Telemetry System

These requirements will influence the choice of technologies to be used, ensuring that the selected solutions are aligned with the system's needs.

4.2 System Architecture

The following diagram presents the proposed system architecture for the telemetry system, detailing the structural design and key components. The diagram illustrates the different elements involved in the system and the technologies used to connect them.

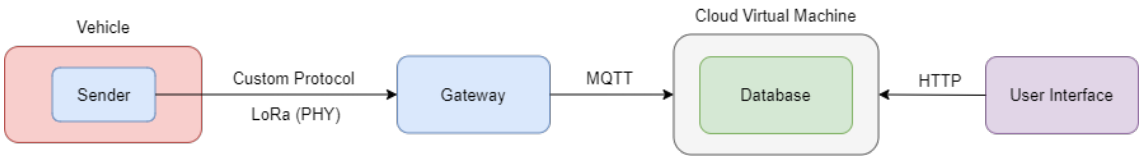

Figure 4.1: System Architecure

4.2.1 Sender

The ECU, which stores data on an SD card (2.4), transmits this data to a microcontroller present in the car designated as the Sender. This Sender is responsible for processing the data and sending it via LoRa to the gateway using a custom-made communication protocol.

4.2.1.1 Custom-made Communication Protocol

The decision to develop a custom-made protocol enhances communication efficiency by reducing unnecessary overhead. Standard communication protocols often include additional metadata or complexity that can consume bandwidth and increase latency. By designing a protocol with only the essential components, it is possible to reduce overhead, improve transmission efficiency (due to fewer bytes being transmitted), and maintain a compact data packet size.

Each packet of this protocol comprises three main parts, as demonstrated in Figure 4.2:

- **Sequence Number:** This 2-byte sequence number facilitates the monitoring of packet loss from the gateway's perspective. If the sequence numbers of received packets are consecutive, no packet loss is assumed. Any gap between received sequence numbers indicates packet loss, enabling the gateway to calculate the number of packets lost.
- **Payload:** The payload, consisting of 32 bytes, contains data sent from the ECU to the Sender. This data includes 16 variables (each occupying 2 bytes) read from the vehicle's sensors:
  - Timestamp: Time elapsed since the car started producing data.
  - DC Current: Battery current.
  - DC Voltage: Battery voltage.
  - Minimum Cell Temperature: Lowest temperature of the battery.
  - Maximum Cell Temperature: Highest temperature of the battery.
  - Average Cell Temperature): Mean temperature of the battery.
  - APPS1: Reading from the first throttle sensor.
  - APPS2: Reading from the second throttle sensor.
  - Brake Value: Pressure applied on the brake pedal.
  - Motor RPM
  - Motor Current
  - Inverter Temperature
  - Motor Temperature
  - Motor Torque
  - Battery Voltage (another measurement point for battery voltage)

  - Motor Voltage

  The ECU sends this data in integer format, which the Sender then converts into a 2-byte format.

- **Cyclic Redundancy Check (CRC):** The CRC is a 2-byte error detection code calculated using the CRC-16-CCITT (International Telegraph and Telephone Consultative Committee) algorithm. This process uses two predefined values: an initial value of 0xFFFF and a polynomial of 0x1021. The entire packet is processed in a 2-byte format to generate a CRC value that is appended to the data before transmission. This CRC value acts as a checksum, verified upon receipt to detect any changes or errors that might have occurred during transmission. If the recalculated CRC matches the transmitted CRC, the packet is considered intact. Otherwise, it indicates potential data corruption.

Figure 4.2: Custom-Made Protocol Packet Structure

This protocol's simplified design is crucial for ensuring high performance and reliability within the constraints of the telemetry system.

4.2.2 LoRa

LoRa technology is selected for its long-range communication capabilities, essential for this telemetry system. Its ability to handle long-range transmissions with minimal power consumption and low cost makes it ideally suited to the project's goals. The efficiency of LoRa is critical, not only for reducing operational costs but also for ensuring reliable data transmission over extended distances, which is a core requirement in automotive telemetry applications.

The decision to use LoRa is further validated by calculating the Time on Air (ToA), which confirms that the protocol's packet size and transmission parameters are well-matched with the system's operational requirements. Managing ToA is crucial for complying with duty cycle restrictions and regulatory limits on radio frequency use.

To determine the ToA for a LoRa packet, the following formulas from Semtech datasheets[53] are utilized:

$$N_{\text{symbol}} = N_{\text{symbol preamble}} + 4.25 + 8 + \text{ceil}\left(\frac{\max\left(8 \cdot N_{\text{byte payload}} + N_{\text{bit CRC}} - 4 \cdot SF + 8 + N_{\text{symbol header}}, 0\right)}{4 \cdot SF}\right) \cdot (CR + 4) \tag{4.1}$$

$$ToA = \frac{2^{SF}}{BW} \cdot N_{\text{symbol}} \tag{4.2}$$

- N_{symbol}: The total number of symbols transmitted per packet, which is critical in determining the duration of the transmission.
- $N_{\text{symbol preamble}}$: The number of preamble symbols in the packet. The preamble is essential for allowing the receiver to synchronize with the signal.
- $N_{\text{byte payload}}$: The number of bytes in the payload. This value determines how much data is being transmitted and affects the packet's length.
- $N_{\text{bit CRC}}$: The number of bits used for the Cyclic Redundancy Check. CRC helps in detecting errors that may have occurred during the transmission.
- $N_{\text{symbol header}}$: The number of header symbols used in the packet.
- SF (Spreading Factor): Affects the number of chirps per symbol. Higher SF increases the range but reduces the data rate.
- BW (Bandwidth): This is the frequency width used in transmission. Increasing the bandwidth can increase the data rate but may reduce sensitivity.
- CR (Code Rate): This represents the rate at which error correction bits are added to the transmission, improving resilience to interference.

The specific values used in the ToA calculation for our system are:

- $N_{\text{symbol preamble}} = 8$
- $N_{\text{byte payload}} = 36$
- $N_{\text{bit CRC}} = 0$
- $N_{\text{symbol header}} = 0$
- $SF = 7$
- $BW = 125\,\text{kHz}$
- $CR = 1$ (corresponding to a 4/5 rate)

The parameters $N_{\text{bit CRC}}$ and $N_{\text{symbol header}}$ are set to 0 because the CRC is included in the payload and there are no additional header symbols beyond the standard packet structure, respectively.

The calculated ToA for this configuration is 70.25 milliseconds. Given the duty cycle restrictions of 1%, this setup allows for transmission intervals of approximately 7 seconds, which aligns with the system's distance and data rate requirements. These settings are configurable when setting up LoRa communication, allowing for adjustments based on specific needs or conditions.

By implementing this LoRa with the custom-made protocol, efficiency and reliability are optimized while adhering to system requirements. This approach ensures effective telemetry data

transmission, balancing the need for long-range communication with cost considerations and compliance with duty cycle restrictions.

To meet the requirement of transmitting data every 10 seconds, and considering the constraints that only one value per variable could be sent within this interval, it was agreed upon with FS FEUP to send an average of the values received every 10 seconds for each variable. This approach ensures that the transmitted data is representative and consistent while meeting the transmission and duty cycle limitations imposed by the system design and regulatory constraints.

In the following flowchart (Figure 4.3), there are two processes happening concurrently on the sender to ensure no data is lost:

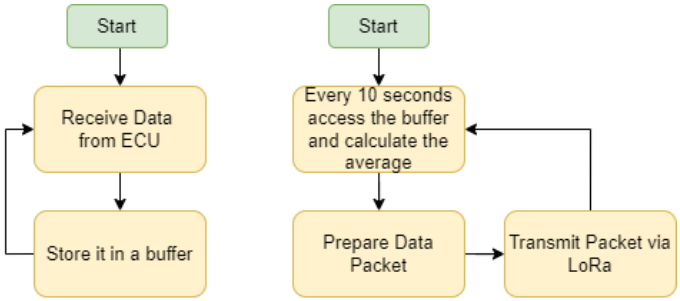

Figure 4.3: Flowchart of Sender process

The sender continuously receives data from the ECU and stores it in a buffer, ensuring that all data is captured and temporarily saved for processing. Every 10 seconds, the sender accesses the buffer to calculate the average of the values received for each variable. The averaged data is then prepared into a data packet and transmitted via LoRa. These concurrent processes are necessary to ensure that no data received from the ECU is lost. The continuous reception and buffering of data allow the system to capture all incoming information, while the periodic averaging and transmission process ensures that the data is sent efficiently and accurately.

4.2.3 Gateway

The gateway plays a critical role in the system by acting as an intermediary between the vehicle and the cloud. Using a gateway is essential for several reasons. Firstly, it ensures data integrity and accuracy. The gateway decodes the sequence number, extracts the payload, and verifies the data using the CRC. This process guarantees that only accurate and complete data is processed, preventing corrupted or incomplete data from being transmitted to the cloud virtual machine (VM) and ultimately to the UI.

Secondly, the gateway optimizes data transmission. It uses the MQTT protocol, which is specifically chosen for its lightweight messaging capabilities and efficient bandwidth usage. This optimization is crucial for reliable data transmission, particularly in environments with limited bandwidth or high latency, such as vehicular communication systems. By processing the data before it reaches the cloud VM, the gateway prevents data overload and ensures that the system can handle large amounts of data efficiently. This step is vital for maintaining scalability and system responsiveness.

Without the gateway, direct communication between the vehicle and the UI would bypass these critical functions, potentially leading to inefficiency, data loss, and system instability. The gateway ensures that the cloud VM and UI receive only clean, validated data, which is essential for the smooth operation of the entire system. The sequence of actions in the gateway process when receiving a packet is illustrated in Figure 4.4.

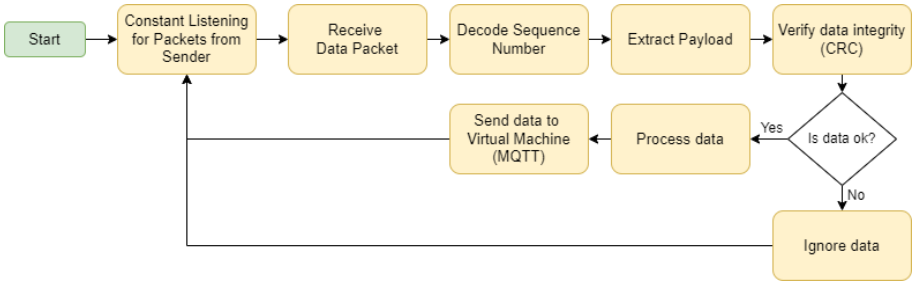

Figure 4.4: Flowchart of Gateway process

4.2.4 Cloud Virtual Machine

The cloud VM is a main component of the system, responsible for storing and managing the data received from the gateway. This solution is chosen for several key reasons. Firstly, the cloud VM offers scalability, allowing the system to handle the large volumes of telemetry data generated by the vehicle. The cloud infrastructure can be easily adjusted based on the data load, ensuring efficient management of resources. This scalability ensures that the system can accommodate varying data loads without compromising performance.

Secondly, the cloud VM provides high availability, ensuring continuous operation with minimal downtime. Cloud service providers offer robust infrastructure with redundant systems, which is essential for real-time data processing and accessibility.

A database hosted on the cloud VM stores the telemetry data. This database is designed to handle large volumes of data, enabling quick access and retrieval. The database architecture supports efficient querying and indexing, speeding up data retrieval and supporting real-time data analysis. Having the database on the cloud VM also enables data access from different locations, making it possible for various parts of the system or different users to retrieve and analyze the data as needed.

The sequence of actions in the cloud VM process when receiving data from the Gateway is illustrated in Figure 4.5.

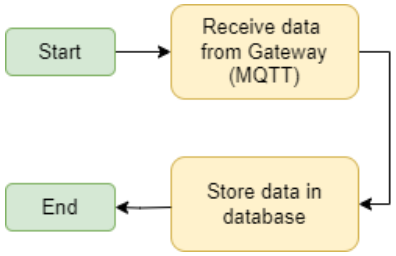

Figure 4.5: Flowchart of Cloud Virtual Machine Process when receiving data from the Gateway

4.2.5 User Interface

The UI retrieves data from the cloud VM via HTTP requests to its API. HTTP is chosen for its simplicity and compatibility with web technologies, making it an ideal choice for data transmission.

The UI is developed using Flask [54] for the backend, which provides a lightweight and modular framework for easy development and deployment. Flask allows for rapid development and is well-suited for building robust APIs that facilitate communication between the UI and the cloud VM.

For the frontend, HTML, JavaScript, and Chart.js are used to create interactive and visually appealing data visualizations. HTML and JavaScript provide the structure and functionality for the web pages, while Chart.js is utilized to generate dynamic charts and graphs that present the telemetry data in an accessible and comprehensible manner.

The UI will consist of several pages, each with specific functionalities. The Time Charts page allows users to view charts for every variable. Users can select which charts to display, making it easy to monitor specific data points over time. The Variables Comparison page features a chart with one horizontal axis and two vertical axes, where two variables chosen by the user are presented. This allows for a direct comparison of different variables on the same timeline. The Single Values page displays a gauge chart for each variable, showing the latest value present in the database for every variable. The Sessions page combines the functionalities of the Time Charts and Variables Comparison pages. It allows users to view data from previous sessions. Additionally, users can load a Comma-Separated Values (CSV) file and utilize the Time Chart and Variables Comparison functionalities with the loaded data, making it easy to analyze and visualize data from external sources.

To navigate between these pages, buttons will be available at the top right of each page, providing easy access to different functionalities and ensuring a smooth user experience.

The sequence of actions when the UI requests data from the cloud VM is shown in Figure 4.6.

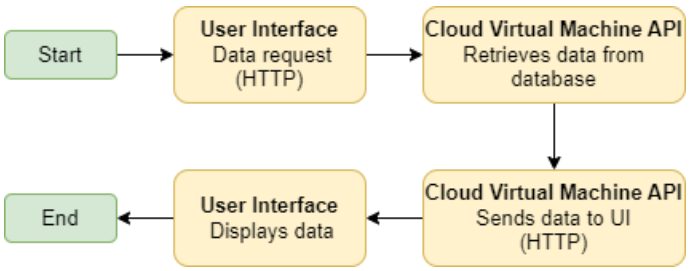

Figure 4.6: Flowchart of UI data request process

The process begins with the UI sending an HTTP request to the API hosted on the cloud VM. This request will occur at regular intervals while viewing a session in progress or it happens once when loading a page to view a session that has already occurred. The API processes this request, retrieves the required data from the database, and sends the data back to the UI in JavaScript

Object Notation (JSON) format. The UI then processes this data and updates the visualizations accordingly.

This architecture ensures that users can easily access and interact with the telemetry data, allowing for real-time monitoring and analysis. The use of HTTP and APIs facilitates communication between the UI and the cloud VM, while the combination of Flask, HTML, JavaScript, and Chart.js provides a robust and flexible platform for developing and displaying the user interface.

4.3 Conclusions

This chapter has outlined the design and functionality of the telemetry system. We detailed the requirements, system architecture, and technology choices, showing how they come together to create an efficient and reliable system.

The telemetry system uses LoRa for long-range communication, with a custom-made protocol to reduce overhead and improve efficiency. The gateway processes and forwards data using MQTT, while the cloud virtual machine stores and manages the data. The user interface, built with Flask and JavaScript, provides an intuitive and interactive way for users to view and analyze the data.

The design ensures that data flows continuously from the vehicle to the user, meeting all the requirements for distance, data rate, and usability. This structured approach will guide the development and implementation of the system, ensuring it performs effectively and meets all project goals.

Chapter 5

Implementation

This chapter details the implementation and testing of the telemetry system developed for FS FEUP. The system comprises a sender, gateway, and user interface, each built with specific technologies and configurations to meet the requirements of the FS FEUP team. Additionally, the chapter presents the results of tests conducted to validate the functionality and reliability of the network, focusing on packet loss and Received Signal Strength Indicator (RSSI) at various distances.

5.1 Sender

During the development phase, it became apparent to the FS FEUP team that transmitting data every 10 seconds was too infrequent for their needs. The interval between transmissions was revised because the FS FEUP required more frequent updates. Since the Sender receives data from the vehicle's ECU every 10 milliseconds, the transmission strategy was adjusted. Instead of averaging data every 10 seconds, the system now averages data every 10 sets of values received, resulting in a transmission interval of 100 milliseconds.

Due to the advanced stage of the development, it was not feasible to change the wireless technology. Consequently, the increased transmission frequency rate, although better suited to the FS FEUP's needs, exceeds the 1% duty cycle limits of the 868 MHz frequency band imposed by regulatory restrictions. This system will be used for development purposes and not in competitive environments. To use it in competitive scenarios, the wireless technology would need to be revised to comply with regulations.

The Sender is implemented using a Raspberry Pi Pico (Figure 5.1) and a LoRa Module SX1262 (Figure 5.2). The Raspberry Pi Pico handles data reception and processing, while the LoRa Module SX1262 handles packet transmission. The code for Raspberry Pi Pico was written in MicroPython utilizing the micropySX126X library [55] for LoRa communication.

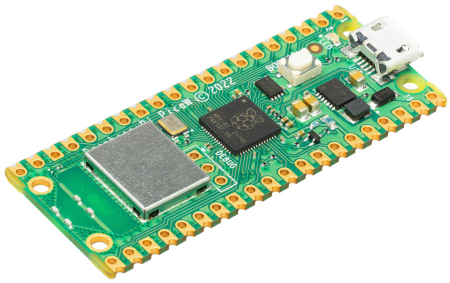

Figure 5.1: Raspberry Pi Pico [56]

Figure 5.2: LoRa Module SX1262 [57]

Data is received from the vehicle ECU through UART. The UART is initialized at a baudrate of 57600 on both sides of the communication channel. For communication with the LoRa module, it is used SPI. The LoRa module and the Raspberry Pi Pico fit together as seen in Figure 5.3. The pins of the Raspberry Pi Pico used for the SPI connections and control signals are detailed in Table 5.4:

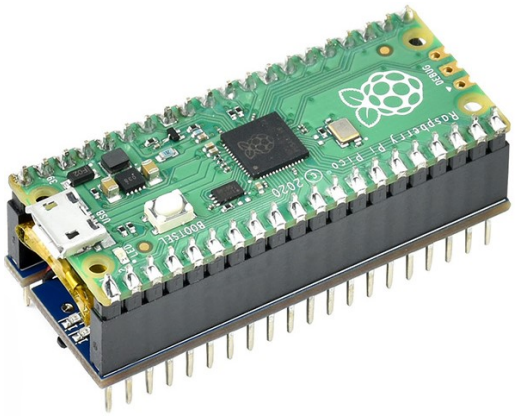

Figure 5.3: Raspberry Pi Pico with the LoRa module SX1262 [57]

LoRa Module SX1262	**Raspberry Pi Pico**
TX	0
RX	1
CLK	10
MOSI	11
MISO	12
CS	3
IRQ	20
RST	15
GPIO	2

Figure 5.4: Pin Configuration for LoRa Module SX1262 and Raspberry Pi Pico

The LoRa module is initialized using the following parameters: frequency 868 MHz, bandwidth 125 kHz, spreading factor 7, code rate 5 (equivalent to 4/5 code rate), synchronization word 0x12, and maximum power 22 dBm. The synchronization word is a unique identifier used to differentiate between different LoRa networks, ensuring that the module only communicates with devices within the same network. The maximum power setting is used to ensure the strongest possible signal, maximizing the communication range and reliability. These settings optimize the LoRa communication for the specific requirements of the FS FEUP telemetry system.

To handle the short time intervals between data receptions, tasks are distributed between the two cores of the Raspberry Pi Pico. One core handles data reception and storage in an array, accommodating all variables. As data is received, the received values are added to those already

there, except for the first one, which represents the timestamp and is always overwritten. The second core is responsible for checking when 10 sets of data have already been received. When this happens, it is responsible for calculating the average, preparing the packet with the data, and sending it via LoRa.

Mutexes are used to manage concurrent access to the data array. In the core responsible for receiving data, the mutex is acquired after confirming the reception of a complete data set and then released after data is stored. The second core acquires the mutex only when ten data sets have been received, performs the averaging, clears the array, and then releases the mutex. To acquire the mutex, the *acquire* function is used, and to release the mutex is used the *release* function, both from the *_thread* library [58].

The averaging process divides all values in the array by the number of data sets (10), except for the timestamp, which remains unchanged. After performing the average, clearing the data array, and releasing the mutex, the data packet is ready to be encoded. The averaged values and the sequence number (initialized to zero and incremented with each packet) are formatted into a 2-byte big-endian format. The packet is completed by calculating the CRC. For this, the CRC is calculated and appended, in a 2-byte big-endian format, to the end of the packet. The packet needs to be in byte format because the transmission function of the *micropySX126X* library requires the packet in this format. Finally, the prepared packet is transmitted. The processes handled by each core are described in flowcharts: Core 0 in Figure 5.5 and Core 1 in Figure 5.6.

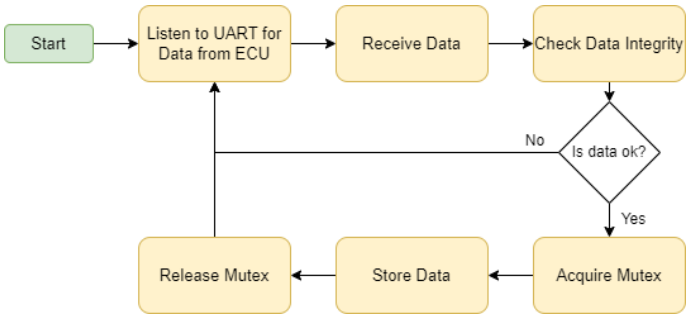

Figure 5.5: Flowchart of Raspberry Pi Pico Core 1 Process

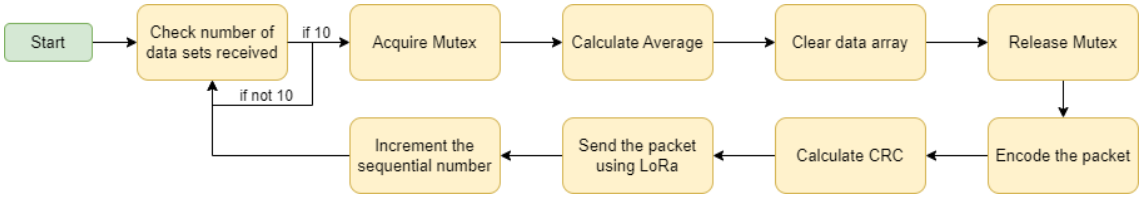

Figure 5.6: Flowchart of Raspberry Pi Pico Core 2 Process

5.2 Gateway

The gateway consists of a Raspberry Pi 5 [59] connected to a Raspberry Pi Pico with a LoRa SX1262 module. This setup was chosen due to specific technical requirements. The LoRa module

for communication only works with the Rasberry Pi Pico. However, the Raspberry Pi Pico does not have integrated Wi-Fi communication capabilities. To address this, the Pico is connected to a Raspberry Pi 5, which has the necessary Wi-Fi functionality. This connection allows the data collected and processed by the Pico to be transmitted via Wi-Fi using MQTT from the Raspberry Pi 5 to a cloud VM.

The Raspberry Pi Pico communicates with the Raspberry Pi 5 via UART. It is connected to a LoRa SX1262 module in the same way as in the Sender. This setup is responsible for receiving data, verifying its integrity, parsing it into integers, and sending it to the Raspberry Pi 5. The initialization of the LoRa module is the same as that done in Sender. The Raspberry Pi 5 is responsible for receiving the data from the Raspberry Pi Pico, processing it, and publishing it with MQTT.

The Raspberry Pi Pico continuously listens for packets from the Sender. When a packet of the expected size (36 bytes) is received, the CRC of the packet is calculated and compared with the received CRC to confirm data integrity. Once the integrity is confirmed, the data is converted from the 2-byte big-endian format to integers so that the Raspberry Pi 5 receives this data ready to be processed and sent to the cloud VM. The data is then sent as an array via UART to the Raspberry Pi 5. The UART is initialized with the same parameters used in the Sender. The Raspberry Pi 5 is powered by a power supply [60] and it, in turn, powers the Raspberry Pi Pico through the pins.

The connection between the Raspberry Pi Pico and the Raspberry Pi 5 is demonstrated in the following table:

Raspberry Pi Pico		**Raspberry Pi 5**	
TX	Pin 1	RX	Pin 10
RX	Pin 2	TX	Pin 8
GND	Pin 3	GND	Pin 6
VBUS	Pin 40	5V	Pin 4

Table 5.1: Connections between Raspberry Pi Pico and Raspberry Pi 5

On the Raspberry Pi 5 side, all programming was done in Python. The Raspberry Pi 5 continuously waits to receive data from the Raspberry Pi Pico. After receiving this raw data, it needs to be processed. This processing is done by using a script provided by FS FEUP. The processing is necessary to convert the raw sensor data into real data that can be stored in the database and analyzed by the FS FEUP team. Once the processing is done, the data is published as an array to the *sensor_data* topic on an MQTT broker located on the virtual machine in the cloud, using the *paho-mqtt* library [61].

The array published in the MQTT topic has 18 elements. The first one is the index, which is the primary key of the database and increments every time data is published on the topic. This value is controlled by the gateway. The index is necessary to have a sequential primary key in the database, ensuring each entry is unique.

The second element is the sequence number, which comes from the packet received by the gateway. The sequence number is used to track the order of the packets as they are received.

This helps in identifying any missing packets during transmission, providing a way to ensure data completeness and integrity.

The second one is the sequence number that comes from the packet received by the gateway. The following are the variables: *timestamp, current, voltage, mintmp, maxtmp, avgtmp, apps1, apps2, brake, rpm, motorcurrent, bamotemp, motortemp, torque, batteryvoltage, motorvoltage.*

The last element is the session. This is controlled by the gateway by checking if the received timestamp is lower than the previous one. If the timestamp is smaller, it indicates that the car has started a new session. This helps in organizing the data into different sessions, making it easier for the FS FEUP team to analyze data specific to each practice session.

In the following images, it is possible to see the flowcharts of the processes for the Raspberry Pi Pico (Figure 5.7) and the Raspberry Pi 5 (Figure 5.8).

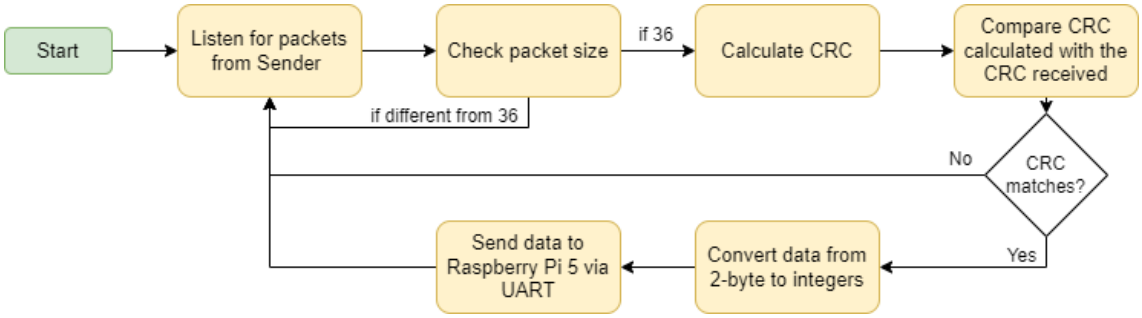

Figure 5.7: Flowchart of Raspberry Pi Pico Process

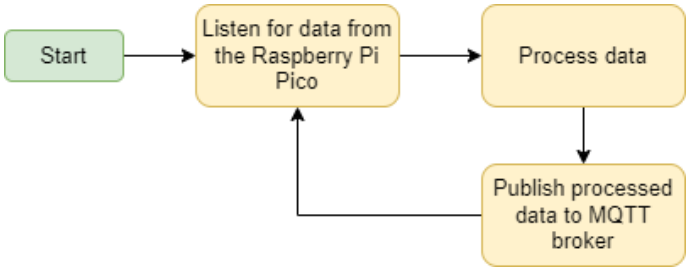

Figure 5.8: Flowchart of Raspberry Pi 5 Process

5.3 Cloud Virtual Machine

After an assessment of various cloud systems, Google Cloud Platform (GCP) [62] was chosen for deploying the cloud VM. The conditions offered in the GCP free tier were the most suitable for the team when compared to other cloud systems, making it the preferred choice in conjunction with the FS FEUP team. Upon deploying the VM, it was necessary to adjust some network settings to use essential development tools and enable communication between the gateway and the VM, and between the VM and the UI.

The following ports were opened from the VM side to facilitate these communications:

- **80 (HTTP):** Communication between UI and VM.
- **1883 (MQTT):** Communication between Gateway and VM.
- **3306 (MySQL):** Use of MySQL Workbench for database management.
- **3389 (RDP):** Enable of Remote Desktop Connection from Windows.

5.3.1 MQTT and Database

A Mosquitto MQTT broker [63] was set up on the VM. This broker enables the gateway to publish data to the VM, allowing the data to be stored in the database. The programming on the VM is done in Python. A script is subscribed to the *sensor-data* topic where the gateway publishes the data. When data is published, the script parses that data into a Structured Query Language (SQL) query to store it in the database present in the cloud VM. The query consists of an index, sequence number, values for every variable of the vehicle, and the session number.

The data is stored in a table called *data* in the MySQL database on the cloud VM. Figure 5.9 shows the structure of the *data* table.

index	seq_number	timestamp	current	voltage	mintmp	maxtmp	avgtmp	apps1	apps2	brake	rpm	motorcurrent	bamotemp	motortemp	torque	batteryvoltage	motorvoltage	session
0	1	0.00	0.0	410	25	25	25	0	0	7	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
1	2	0.01	0.0	410	25	25	25	0	0	7	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
2	3	0.02	0.0	410	25	25	25	0	0	15	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
3	4	0.03	0.0	410	25	25	25	0	0	23	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
4	5	0.04	0.0	410	25	25	25	0	0	23	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
5	6	0.05	0.0	410	25	25	25	0	0	31	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
6	7	0.06	0.0	410	25	25	25	0	0	39	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
7	8	0.07	0.0	410	25	25	25	0	0	47	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
8	9	0.08	0.0	410	25	25	25	0	0	47	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
9	10	0.09	0.0	410	25	25	25	0	0	55	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
10	11	0.10	0.0	410	25	25	25	0	0	63	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
11	12	0.11	0.0	410	25	25	25	0	0	70	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
12	13	0.12	0.0	410	25	25	25	0	0	70	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
13	14	0.13	0.0	410	25	25	25	0	0	78	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
14	15	0.14	0.0	410	25	25	25	0	0	86	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
15	16	0.15	0.0	410	25	25	25	0	0	94	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
16	17	0.16	0.0	410	25	25	25	0	0	94	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
17	18	0.17	0.0	410	25	25	25	0	0	102	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
18	19	0.18	0.0	410	25	25	25	0	0	110	0.000000	0.000000	0.000000	0.000000	0.000000	0	0.000000	0

Figure 5.9: Structure of the *data* table

5.3.2 API for responding to UI Requests

The VM hosts an API built with Flask [54]. This API is responsible for handling requests from the UI and interacting with the database to retrieve data as needed. The API facilitates the communication between the UI and the database, ensuring that data is accessible and up-to-date.

Below is a general overview of how the API works, followed by a detailed explanation of each endpoint's functionality.

5.3.2.1 API Functionalities

The API defines several routes (endpoints) that correspond to different functionalities. Each route is associated with a specific function that handles the request. When the API receives a request, it calls the appropriate function from a library developed to interact with the database. This module contains functions that retrieve data from the database. The data retrieved from the database is in JSON format and is sent back as a response to the UI request.

5.3.2.2 Detailed Explanation of API Functionalities

- **Get Column Data:**

  Endpoint: "/api/data/get_column/<column_name>/<session>"

  This endpoint retrieves all values from a specified column for a given session. It calls the "get_column" function, which queries the database to fetch the column values and returns them as a JSON response. This endpoint will be used to see data when a page (Time Charts, Variables Comparison, or Previous Sessions) is loaded.

- **Get Column Data Between Two Values:**

  Endpoint: "/api/data/get_column_from/<column>/<min_value>/<max_value>/<session>"

  This endpoint retrieves all values from a specified column that fall between two specified values for a given session. It calls the "get_column_from" function, which performs a query to fetch the required data range and returns it as a JSON response. This endpoint will be used so that the UI can update the data being displayed, and if a new session starts, the UI does not show the new data stored in the databse, only shows the data of the session that was being presented. This endpoint will be used by the Time Charts and Variable Comparison pages.

- **Get Maximum Primary Key Value:**

  Endpoint: "/api/data/get_max_pk/<session>"

  This endpoint retrieves the maximum primary key value for a given session. It calls the "get_max_pk" function, which queries the database to find the highest primary key value and returns it as a JSON response. This endpoint will be used so that when a page is loaded, the UI knows what index they have data. This allows the UI to retrieve only the new data without needing to retrieve all the data for the entire session.

- **Get Current Session:**

  Endpoint: "/api/data/get_current_session"

This endpoint retrieves the current session number. It calls the "get_current_session" function, which queries the database to find the most recent session number and returns it as a JSON response. This endpoint is used every time a page is loaded so that the UI can know from what session is the data being displayed and request the VM if there is new data for that session.

- **Get Last Value from Column:**

  Endpoint: "/api/data/get_last_value/<column_name>/<session>"

  This endpoint retrieves the last value from a specified column for a given session. It calls the "get_last_value" function, which queries the database to fetch the most recent value in the specified column and returns it as a JSON response. This endpoint will be used to tee see the values when the Single Pages value loads.

- **Get Value from Column by Index:**

  Endpoint: "/api/data/get_value/<column_name>/<value>/<session>"

  This endpoint retrieves a specific value from a specified column based on the primary key (index) for a given session. It calls the "get_value" function, which queries the database to fetch the value at the specified index and returns it as a JSON response. This endpoint will be used for the Single Values page so that the UI can update the data being displayed, and if a new session starts, the UI does not not show the new values, only shows the last value of the session that was being presented.

5.4 User Interface

The UI is a web application developed using Flask for the backend and HTML, CSS, and JavaScript for the frontend. HTML is used to structure the content on web pages, providing a framework for the layout and elements of the UI. CSS is employed to style the content, ensuring that the visual appearance aligns with the FS FEUP team's specifications. JavaScript is used to handle the dynamic aspects of the UI, such as updating values in real-time and enabling user interactions.

The layout of the charts and buttons, the color scheme, and the font type were all designed based on the specifications provided by the FS FEUP team. For every page, the values are updated every 500 milliseconds using the Asynchronous JavaScript and XML (AJAX) method [64]. AJAX is a technique for creating fast and dynamic web pages by allowing parts of a web page to be updated asynchronously, without needing to reload the entire page. There are two buttons at the top of the page to access the other two pages. These features are common across all pages, ensuring a consistent user experience. In Figure 5.10 is possible to see the general appearance of the UI when it is loaded.

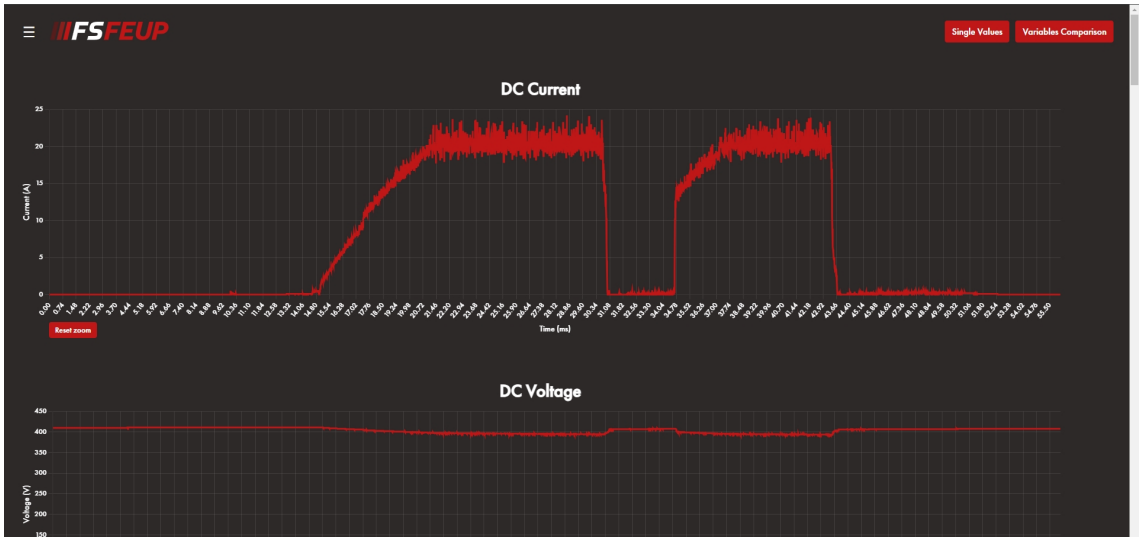

Figure 5.10: Time Charts page (Page shown when loading the UI)

5.4.1 Time Charts

The *Time Charts* page displays a series of charts, each representing a different variable (Figure 5.10). These charts are built using the *Chart.js* [52] library and are updated every 500 milliseconds to reflect the latest data received from the vehicle. There is a dropdown menu on the top left to choose which charts to display (Figure 5.11). By default, all charts are displayed when the page loads. Each chart has options for zooming and a button to reset the zoom. Panning, or moving the view of the graph, is also possible.

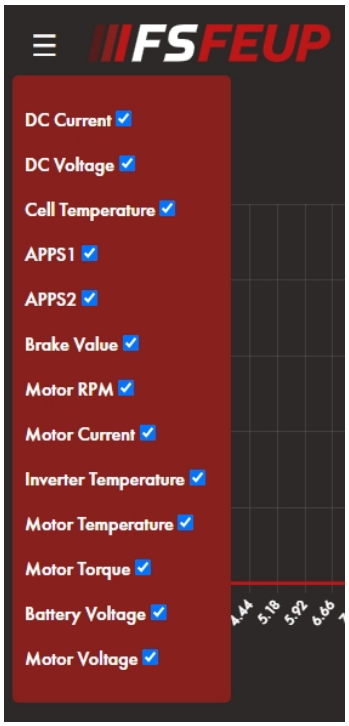

Figure 5.11: Dropdown Menu for Selecting charts

The *Time Charts* page uses the *Get Current Session* and *Get Column Data* endpoints when the page loads for the first time. By making these requests, the UI can determine the current session and present data already stored in the database for that session. Using the AJAX method, every 500 milliseconds the UI makes requests to the VM using the *Get Maximum Primary Key Value* endpoint to know the maximum index for the current session.

Then, it requests the *Get Column Data Between Two Values* endpoint to check if there is new data for this specific session. If there is new data, it is appended to the arrays that save and display it, reducing the computer effort required by the UI. If a new session starts, the page needs to be refreshed to see the values from the new session.

In Figure 5.12, it is possible to see a flowchart of the *Time Charts* page behavior.

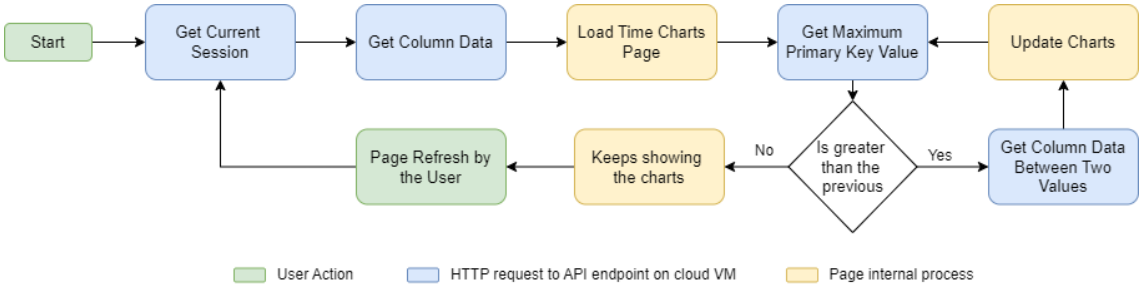

Figure 5.12: Flowchart of Time Charts Page Behavior

5.4.2 Variables Comparison

The *Variables Comparison* page (Figure 5.13), like the *Time Charts* page, features a button on the top left to choose the variables to be compared, the only difference is that can only be selected two variables. This page has the same behavior as the Time Charts page. It uses the *Get Current Session* and *Get Column Data* endpoints when the page loads for the first time to determine the current session and present data already stored in the database for that session.

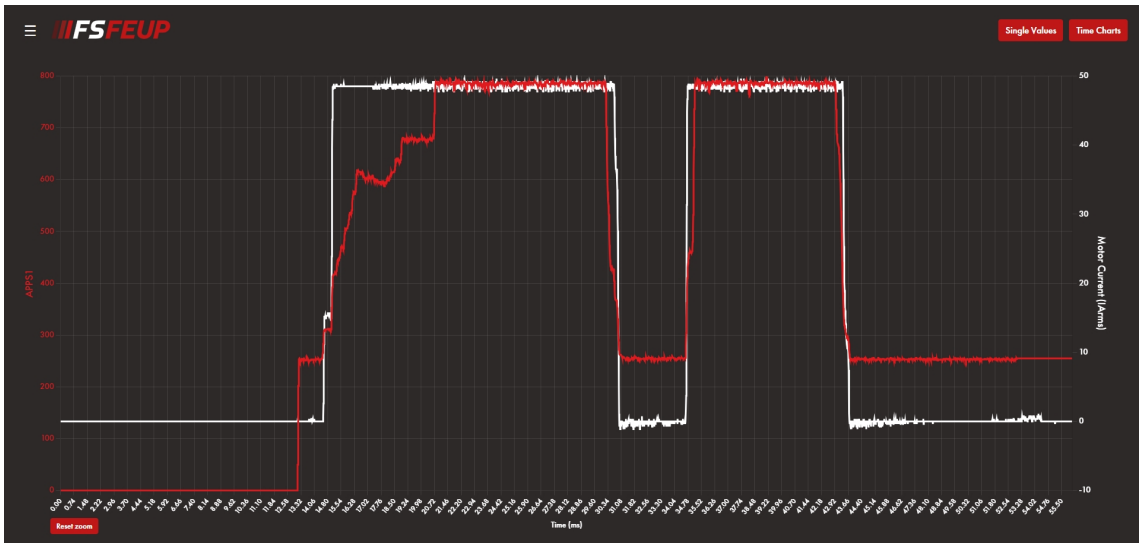

Figure 5.13: Variables Comparison page. Comparison between APPS1 (red) and Motor Current (white)

The update process is also equal to what happens in the *Time Charts* page.

In Figure 5.14, it is possible to see a flowchart of the *Variables Comparison* page behavior.

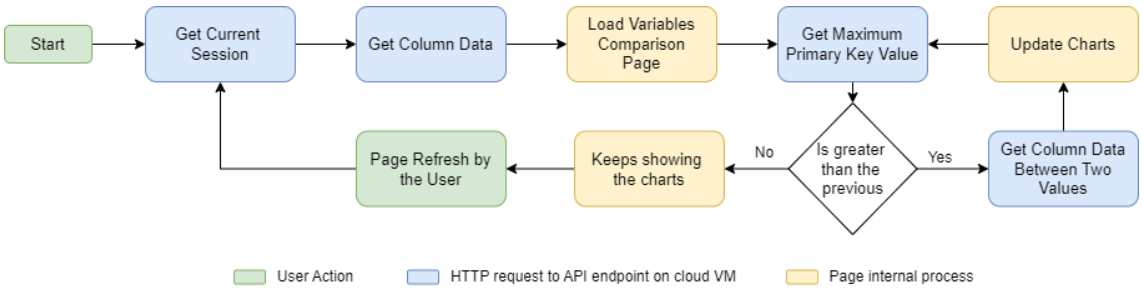

Figure 5.14: Flowchart of Variables Comparison Page Behavior

5.4.3 Single Values

The *Single Values* page aggregates the single values into four groups (Figure 5.15) and shows the last value for the current session present in the database. The values are presented using gauge charts from the *JustGage* library [65]. These gauge charts were defined to have a green-to-red color gradient, as requested by FS FEUP. For some variables, like temperatures, the gradient goes

from green to red based on predefined values agreed upon with FS FEUP. For other variables, such as DC Voltage, the gradient goes from red to green.

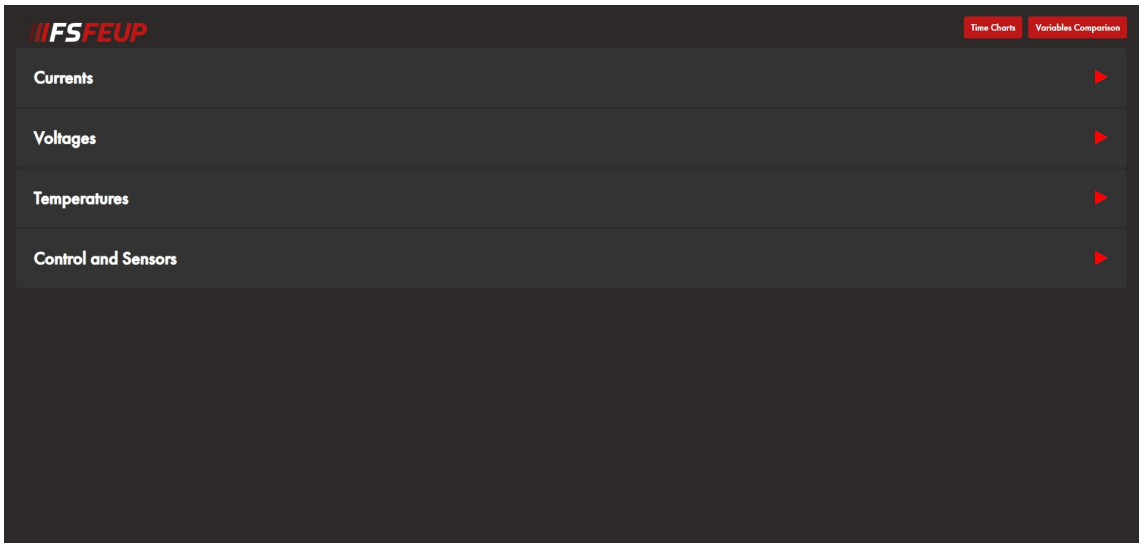

Figure 5.15: Single Values Page when is loaded

The *Currents* group includes *DC Current* and *Motor Current* (Figure 5.16). The *Voltages* group includes *Motor Voltage and DC Voltage* (Figure 5.17). The *Temperatures* group includes *Cell Average Temperature, Inverter Temperature, and Motor Temperature* (Figure 5.18). The *Control and Sensors* group includes *APPS1, APPS2, Brake, RPM, and Torque* (Figure 5.19).

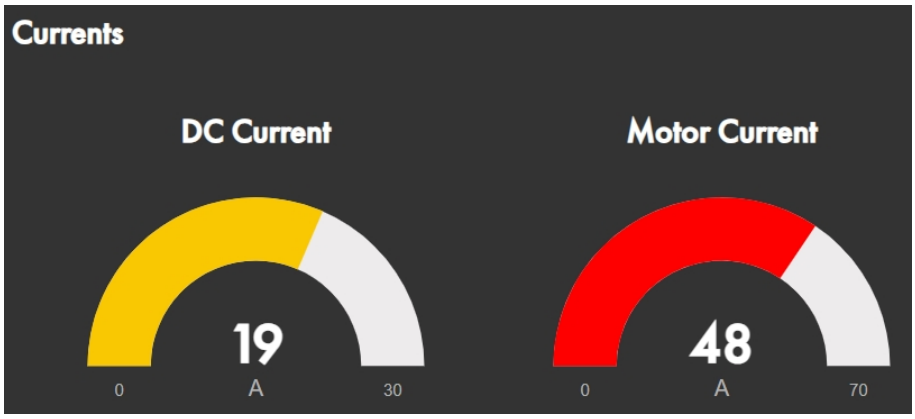

Figure 5.16: Currents

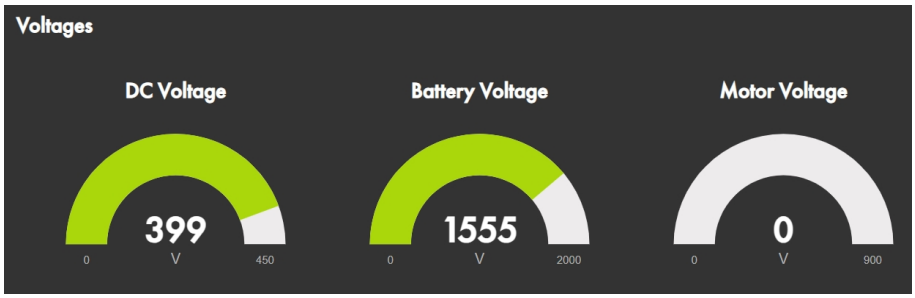

Figure 5.17: Voltages

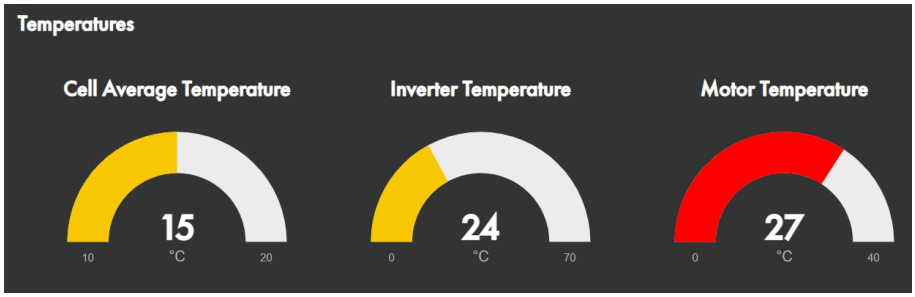

Figure 5.18: Temperatures

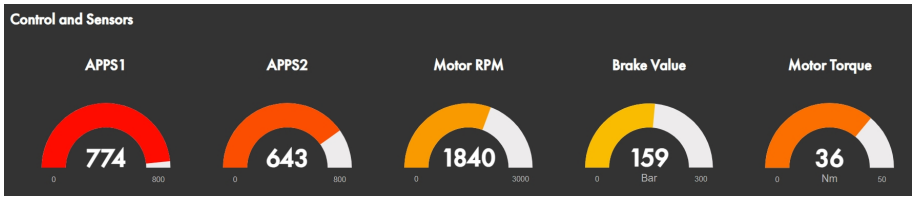

Figure 5.19: Control and Sensors

The *Single Values* page makes requests to the *Get Current Session* and *Get Last Value From Column* endpoints when the page loads for the first time to determine the current session and present the last value for the current session. Using the AJAX method, every 500 milliseconds, the UI makes requests to the *Get Maximum Primary Key Value* endpoint to know the maximum index for the current session. If the received primary key is greater, a request is made to the *Get Value From Column By Index* endpoint for every variable to retrieve the last value for the current session, and then the values are updated. If not, the last values shown are kept. If a new session starts, the values will only be displayed if the user refreshes the page.

In Figure 5.20, it is possible to see a flowchart of the *Single Values* page behavior.

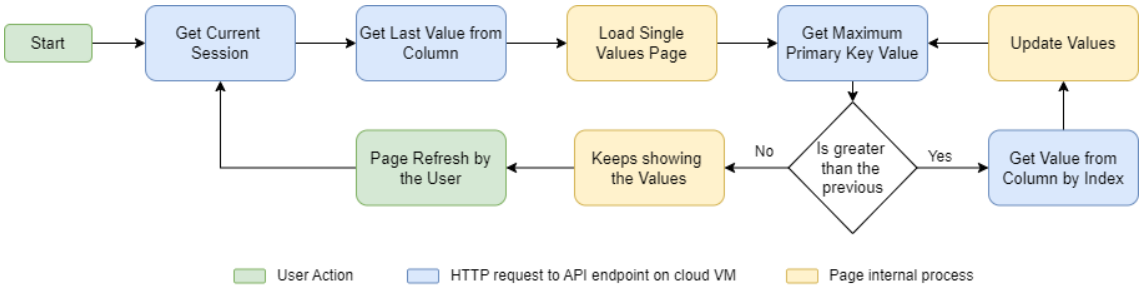

Figure 5.20: Flowchart of Single Values Page Behavior

5.5 Tests

Due to restrictions that prevented testing the system directly with the vehicle, an alternative method was used to validate the functionality and reliability of the system. The experiment involved using two Raspberry Pi Pico devices, each connected to a LoRa module SX1262, to test packet loss and RSSI at various distances. The LoRa setup used in this experiment was the same as the configuration described previously. Two laptops were used to run the experiment, each connected to a Raspberry Pi Pico with a LoRa module. The tests were conducted on a street with little traffic near a train line, covering a distance of 1 km. The street had no trees or other types of objects that could interfere with the signal, ensuring a clear line of sight between the devices. The following image shows the street where the tests were carried out, marked with the path taken for measurements.

Figure 5.21: Aerial view of the testing location, showing the 1 km path

Two antennas [66], each specifically designed for the 868 MHz frequency band with a gain of 2.8 dBi and a length of 195 millimeters, were employed—one for each LoRa module. In terms of

coding, the transmitter reads data from a CSV file containing values from a practice session of the vehicle. Every 10 milliseconds, it reads a line from the file. After receiving 10 lines, it averaged the data, encoded the packet, and sent it via LoRa. The receiver then received the packet, confirmed its integrity by calculating and comparing the CRC, and checked whether the sequence numbers were incremented by one to detect any packet loss. The RSSI was calculated using a function from the *micropySX126X* library. Both laptops remained stationary during the experiment. For each distance, the test lasted 10 minutes, equivalent to sending 6000 packets (one packet every 100 milliseconds).

5.5.1 Results and Discussion

The experiment was conducted at different distances: 100m, 250m, 500m, 750m, and 1000m. At each distance, the packet loss was monitored, and the RSSI values were recorded. The system exhibited zero packet loss at all distances tested, demonstrating its reliability.

The RSSI values for each distance are presented in the box plot shown in Figure 5.22

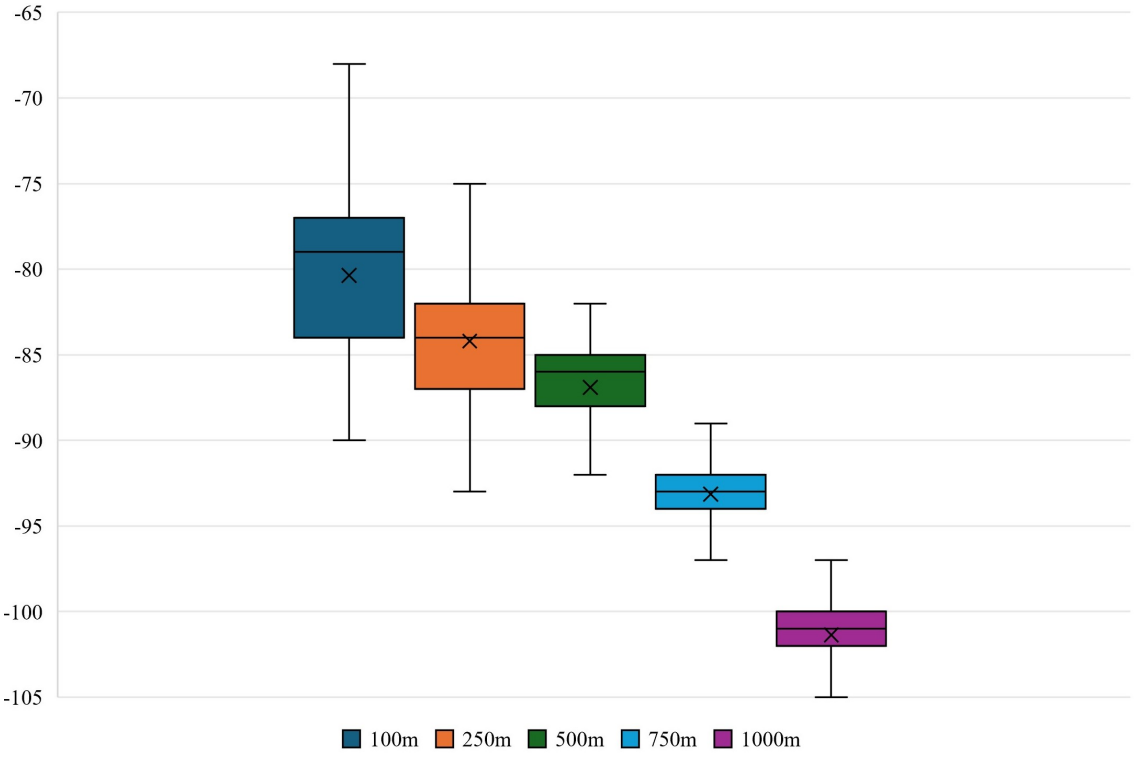

Figure 5.22: RSSI values at different distances

The results indicate a decrease in RSSI with increasing distance, which is consistent with the characteristics of radio frequency signals that tend to attenuate over longer distances [67].

Despite the reduction in signal strength, the system showed remarkable robustness by maintaining a consistent communication link across all tested distances, up to 1000 meters. This indicates that the LoRa setup, including the configuration and hardware used, is effective in these

conditions.

The zero packet loss shows the system's reliability in data transmission, even at maximum tested distances. This capability ensures that the data from the vehicle is accurately transmitted to the gateway.

The results also highlight the stability of the LoRa communication system. The use of appropriate antennas and well-chosen LoRa parameters (frequency, spreading factor, bandwidth) have contributed to this performance. However, it is important to note that these tests were conducted with both the transmitting and receiving devices stationary. In a real-world scenario, the vehicle will be moving, which could affect the RSSI and packet loss. Therefore, while the stationary tests provide a strong indication of the system's capabilities, further testing in dynamic conditions is necessary to fully validate the system's performance under operational conditions.

Overall, the experiment confirms that the chosen technology and configuration are well-suited for the intended application, offering the necessary reliability and long-range communication capabilities for the telemetry system, with the caveat that additional testing under movement conditions is recommended for complete validation.

5.6 Conclusions

This chapter provided an overview of the implementation and testing of the telemetry system designed for FS FEUP. The system's components, including the sender, gateway, and user interface, were developed using a combination of Raspberry Pi devices, LoRa modules, and various software tools such as Flask, HTML, CSS, and JavaScript. The sender and gateway were configured to handle real-time data transmission and reception, ensuring data integrity and efficient communication. The user interface was designed to meet the specific requirements of the FS FEUP team, featuring real-time updates and an intuitive layout.

The chapter also covered the deployment of a virtual machine on GCP, detailing the network settings and API development necessary for seamless data interaction between the UI and the database. The setup of the MQTT broker and the database schema were discussed, highlighting the infrastructure needed to manage and store telemetry data effectively. Although the API has the endpoints for managing sessions, the development of the Sessions page was not completed due to time constraints.

Finally, the testing phase demonstrated the system's reliability and performance, with zero packet loss recorded at various distances and consistent RSSI values observed. These tests validate the robustness of the system under stationary conditions. Even without the development of the sessions page and the functionality to upload a CSV file, the implementation was successful. However, it is important to note that testing was limited to stationary conditions. Future testing should be conducted in more varied conditions to fully assess the system's capabilities and ensure it can handle the demands of real-world applications.

Chapter 6

Conclusions and Future Work

6.1 Overview

In this dissertation, a telemetry system was developed for the FS FEUP team. This system is designed to enhance the team's ability to monitor and analyze real-time data from their vehicle, thereby improving decision-making processes and overall vehicle performance during competitions.

Before the implementation, an extensive study was conducted to understand the current state of telemetry systems and the technologies that could be used for the system. The study included an analysis of various wireless technologies such as Zigbee, Bluetooth Low Energy (BLE), Sigfox, and LoRa. Each technology was evaluated based on its suitability for long-range communication, power consumption, data rate, and ease of integration. Communication protocols at the application layer, including MQTT, CoAP, XMPP, and AMQP, were also examined to determine the most efficient and reliable method for data transmission in a telemetry system.

Additionally, user interface design principles were explored to ensure the telemetry system's UI would be intuitive and user-friendly. Technologies for building telemetry dashboards were compared to identify the best tools for creating interactive and responsive data visualizations.

The implementation included several key components. The telemetry system was developed to transmit data from the vehicle every 100 milliseconds. This system uses LoRa technology for long-range communication, ensuring data is reliably transmitted over distances up to 1000 meters. The gateway decodes and validates data packets from the vehicle before transmitting them to a cloud virtual machine using MQTT. This setup ensures efficient data storage and retrieval. The UI, developed using Flask for the backend and HTML/JavaScript with Chart.js for the frontend, allows users to view and analyze data through interactive and responsive charts.

The system met most of the defined requirements, including wireless, unidirectional communication, real-time data visualization, and an intuitive, user-friendly interface. However, due to time constraints, the Sessions page was not developed.

Testing focused on evaluating the data transmission performance over LoRa. The tests demonstrated that the telemetry system was able to provide accurate and timely data transmission over

the required distance of up to 1000 meters by having no packet loss over all the distances tested and having consistent values for the RSSI. However, it was not possible to test the system with the car in motion, so the tests were limited because the transmitter and receiver remained stationary. Additionally, it was the duty cycle requirements for LoRa were not met, indicating a need for further optimization or exploration of alternative wireless technologies.

6.2 Future Work

While the implemented telemetry system has significantly improved the data acquisition and analysis capabilities of the FS FEUP team, there are areas for future enhancement.

Although the backend infrastructure for the Sessions page is complete, the frontend implementation is pending due to time constraints. Future work should focus on developing the UI for the Sessions page to allow users to review and compare data from previous sessions. This will provide valuable insights into the vehicle's performance over time. Additionally, integrating functionality to upload and visualize CSV files with data from previous sessions on the UI will further enhance data analysis and provide more comprehensive insights into the vehicle's performance.

The current system using LoRa technology has shown some limitations, particularly concerning duty cycle requirements. Future research should explore alternative wireless technologies or optimization techniques to ensure compliance with duty cycle regulations while maintaining reliable long-range communication. This could involve investigating other technologies that can achieve the long-range needed and sufficient data rate transmission to handle all the required data.

To fully understand the system's real-world capabilities, extensive testing should be conducted using the vehicle in various conditions. This will help identify the system's limits and potential areas for improvement. Testing in dynamic environments, such as during actual races or practice sessions, will provide critical data to further refine the system.

By addressing these areas in future work, the telemetry system can be further optimized to meet the evolving needs of the FS FEUP team and ensure they remain competitive in the highly demanding environment of Formula Student competitions. This concludes the development and evaluation of the telemetry system. The advancements made through this project provide a solid foundation for ongoing improvements and highlight the potential for integrating advanced telemetry systems in competitive racing environments.

Appendix A

Cloud VM and Database Implementation

A.1 Creating a cloud VM on GCP

To create a cloud VM on GCP, it is necessary to have a Google account and a valid payment method. Access the GCP console and navigate to the Compute Engine service (Figure A.1).

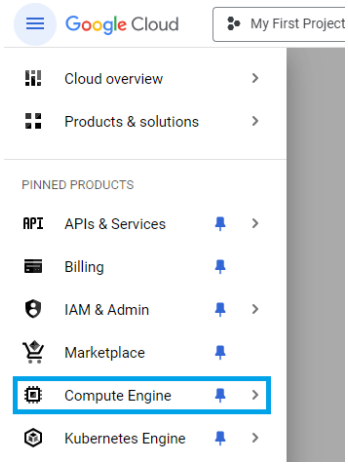

Figure A.1: Compute Engine service

Locate the option to create a virtual machine instance (Figure A.2).

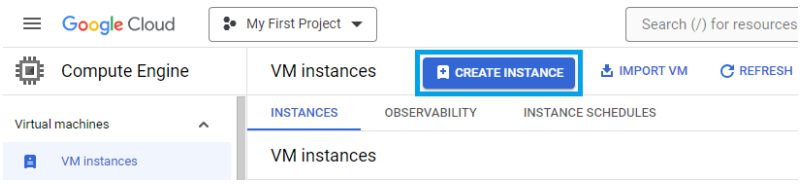

Figure A.2: Create a virtual machine instance

Configure its settings such as machine name, region, machine type, network settings, machine image, and disk type and size. Review the configuration details and confirm the creation of the VM instance (Figure A.3).

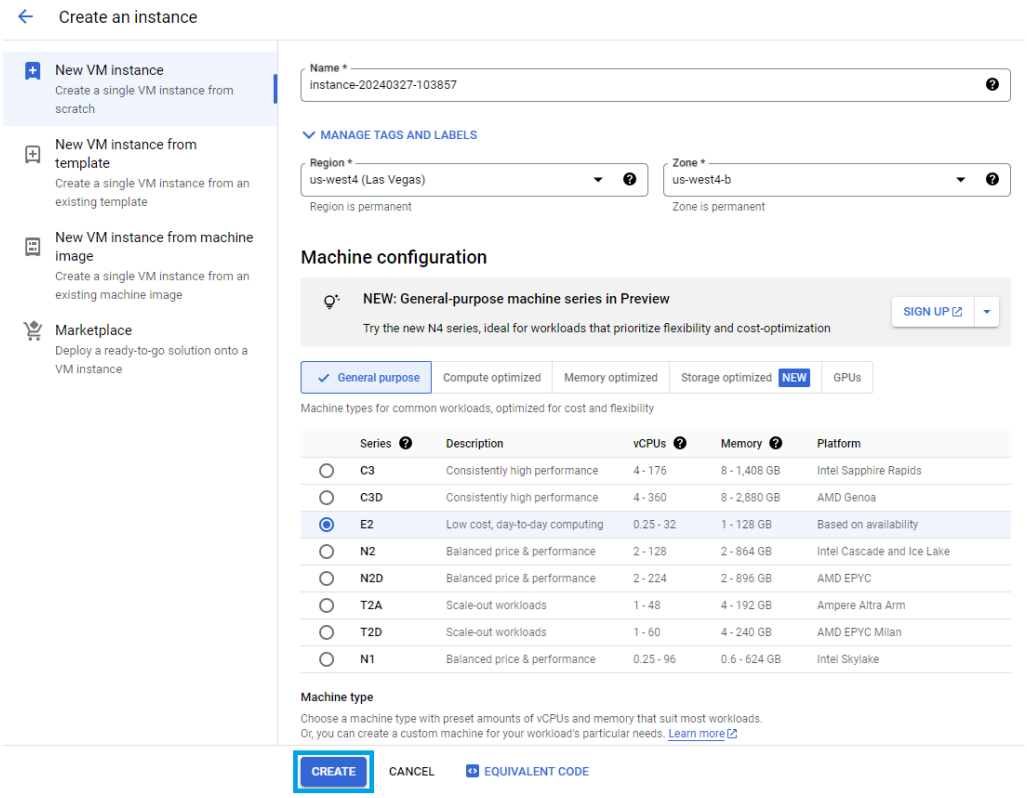

Figure A.3: Configuration of virtual machine instance

The VM used has the following configuration:

- **Machine Name:** sensor-data-fsfeup
- **Region:** europe-southwest1-a
- **Machine Type:** e2-medium (2 vCPU, 1 core, 4GB memory)
- **Network settings:**
  - **Open ports:** 22 (SSH), 80 (HTTP), 443 (HTTPS), 1883 (MQTT), 3306 (MySQL), 3389 (RDP)
  - **Protocols:** ICMP (diagnostic and control purposes)
- **Machine image:** Ubuntu 22.04 LTS, x86/64
- **Disk type:** Standard persistent disk of 20GB

A.2 Setting up a static IP address for the cloud VM

To set up a static IP address for the cloud VM, navigate to the VPC network in the GCP console (Figure A.4).

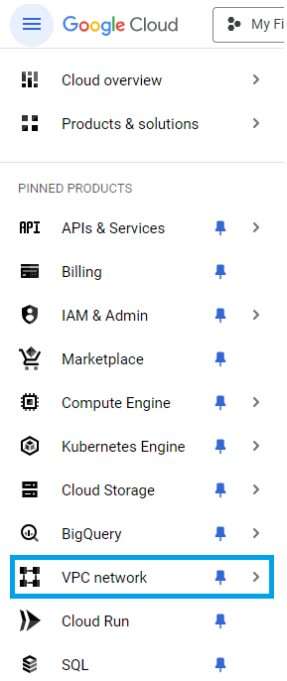

Figure A.4: VPC network service

In the VPC network, locate the IP address of the VM, click on the three dots button at the end of the line, and choose the option "Promote to static IP address". Enter the name for the static IP address and then click "Reserve". The IP address in use will be *'xx.xxx.xxx.xx'*.

A.3 Enabling SSH Access and SSH Key Authentication

To access the VM, it was necessary to enable SSH. Initially, SSH keys were generated using the command *'ssh-keygen -t rsa -b 4096 -C fsfeupuser'*. Once generated, the content of the public key *'fsfeupuser.pub'* was viewed using the command *'type fsfeupuser.pub'* and copied from the terminal. Subsequently, the copied public key was pasted into the SSH Keys section under Configurations > Metadata in the Computer Engine service (Figure A.5).

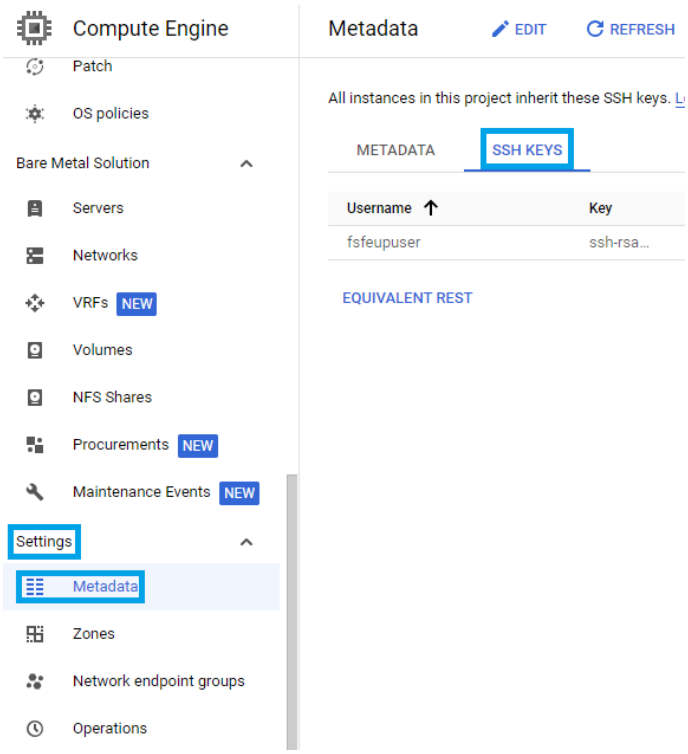

Figure A.5: Enabling SSH keys

To SSH into the VM, the following command was utilized: *'ssh -i fsfeupuser fsfeupuser@xx.xxx.xxx.xx'*, specifying the SSH key file *'fsfeupuser'* and the username *'fsfeupuser'* along with the VM's public IP address *'xx.xxx.xxx.xx'*. This facilitated secure access to the VM's command line interface.

A.4 Configuring Remote Desktop Protocol (RDP) Access

To enable remote desktop access through the Windows application "Remote Desktop Connection", it is necessary to configure the VM to have RDP capabilities. To achieve this, SSH into the VM and execute the following commands:

- *'sudo apt-get update'*
- *'sudo apt-get install -y xrdp'*
- *'sudo apt-get install -y xfce4'*
- *'sudo service xrdp restart'*

These commands will install the necessary components for RDP access and restart the XRDP service, facilitating connection to the VM using the Remote Desktop Connection app from Windows.

After setting up the RDP access, it is needed to create a password for the root user. In the terminal where the previous commands were executed, use the following command to elevate to

root: *'sudo -s'*. Then, enter the following command to set a password for the root user: *'passwd'*. After executing the command, the system will prompt for the desired password for the root user. The password that is used for the root is *'123456'*.

After creating a password for the root user, proceed to connect to the VM using the Windows Remote Desktop Connection app. Use the VM's IP address to establish the connection and log in using the root credentials.

Once inside the VM, open a terminal and change the password for the user *'fsfeupuser'* by executing the following command: *'passwd fsfeupuser'*. The system will prompt for the desired password for the user. After confirming the password, reboot the VM.

It's recommended to connect to the VM using the user account for enhanced security measures. This ensures that administrative tasks are performed using a standard user account rather than the root user, which can help mitigate potential security risks. The password that is used for the *fsfeupuser* is *'1234'*.

A.5 Installing MySQL Server in the VM

Before creating a database, it is necessary to install the MySQL Server. To install it execute the following commands in a terminal:

- *'sudo apt-get update'*
- *'sudo apt-get install mysql-server'*

To ensure the MySQL Server is installed run the following command: *'mysql –version'* and it should appear the installed version of MySQL Server.

After installing MySQL Server, it is necessary to configure it. To do this, run the following command: *'sudo mysql_secure_installation'*. It will ask for the root password which is *'123456'*. After entering the password it will ask many questions which should be responded to with no, only answer with yes when the system asks the following question: "Reload privilege tables now?".

After finishing the configuration, it is required to change the configuration file and to create a remote user so it is possible to connect to the database from outside the VM using the MySQL Workbench application. Execute the following command in the terminal: *'sudo nano /etc/mysql/mysql_conf.d/mysqld.cnf'*. Then in the "bind-address" parameter, change it to "0.0.0.0". Changing this parameter allows MySQL Workbench to connect to the database.

Bibliography

[1] Matteo Muratori et al. “The rise of electric vehicles—2020 status and future expectations”. In: *Progress in Energy* 3.2 (2021), p. 022002.

[2] Julio A Sanguesa et al. “A review on electric vehicles: Technologies and challenges”. In: *Smart Cities* 4.1 (2021), pp. 372–404.

[3] International Energy Agency. *Trends in Electric Cars*. [Online; Accessed: 2024-06-21]. 2024. URL: https://www.iea.org/reports/global-ev-outlook-2024/trends-in-electric-cars.

[4] Global Formula Racing. *Formula Student*. [Online; Accessed: 2024-06-21]. 2024. URL: https://www.global-formula-racing.com/en/formula-student/.

[5] Chris Rothe. *What F1 racing can teach us about telemetry*. [Online; Accessed: 2024-06-21]. 2023. URL: https://redcanary.com/blog/threat-detection/f1-racing-telemetry/.

[6] Raj Krishan Ghosh et al. “Intelligent IoT for automotive industry 4.0: Challenges, opportunities, and future trends”. In: *Intelligent Internet of Things for Healthcare and Industry* (2022), pp. 327–352.

[7] Darsh Parekh et al. “A review on autonomous vehicles: Progress, methods and challenges”. In: *Electronics* 11.14 (2022), p. 2162.

[8] Yalantis. *AN IOT-BASED SYSTEM FOR REAL-TIME FLEET MANAGEMENT*. [Online; Accessed: 2024-07-09]. 2021. URL: https://yalantis.com/works/iot-based-fleet-management/.

[9] Fazel Mohammadi and Rashid Rashidzadeh. “An overview of IoT-enabled monitoring and control systems for electric vehicles”. In: *IEEE instrumentation & measurement magazine* 24.3 (2021), pp. 91–97.

[10] FS FEUP. *About Us*. [Online; Accessed: 2024-06-28]. 2024. URL: https://formulastudent.fe.up.pt/aboutus/.

[11] FS FEUP. *Team*. [Online; Accessed: 2024-06-28]. 2024. URL: https://formulastudent.fe.up.pt/team/.

[12] Le Tian et al. “Wi-Fi HaLow for the Internet of Things: An up-to-date survey on IEEE 802.11 ah research”. In: *Journal of Network and Computer Applications* 182 (2021), p. 103036.

[13] Rajab Challoo et al. “An overview and assessment of wireless technologies and co-existence of ZigBee, Bluetooth and Wi-Fi devices”. In: *Procedia Computer Science* 12 (2012), pp. 386–391.

[14] Nils Braune. “Telemetry Unit for a Formula Student Race Car”. In: *Swiss Federal Institute of Technology Zurich* (2014).

[15] Tinku Kumar and PB Mane. “ZigBee topology: A survey”. In: *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. IEEE. 2016, pp. 164–166.

[16] Thoraya Obaid et al. “Zigbee technology and its application in wireless home automation systems: A survey”. In: *International Journal of Computer Networks & Communications* 6.4 (2014), p. 115.

[17] ShuYu Ding, JianLi Liu, and MingHong Yue. “The use of ZigBee wireless communication technology in industrial automation control”. In: *Wireless Communications and Mobile Computing* 2021.1 (2021), p. 8317862.

[18] Qiang Zhang, Yugeng Sun, and Zhenhui Cui. “Application and analysis of ZigBee technology for Smart Grid”. In: *2010 International Conference on Computer and Information Application*. IEEE. 2010, pp. 171–174.

[19] ICP DAS. *ZigBee Introduction*. Accessed: 2024-07-22. 2024. URL: https://www.icpdas.com/root/product/solutions/industrial_wireless_communication/wireless_solutions/zigbee_introduction.html.

[20] Zigbee Alliance. *Zigbee Specification*. Tech. rep. docs-05-3474-21-0csg. Version 21. Zigbee Alliance, Aug. 2015. URL: https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf.

[21] Digi International Inc. *Zigbee Stack Layers*. Accessed: 2024-07-24. 2017. URL: https://www.digi.com/resources/documentation/Digidocs/90002002/Content/Reference/r_zb_stack.htm?TocPath=zigbee%20networks%7C_____3.

[22] Wikipedia contributors. *Bluetooth Low Energy*. https://en.wikipedia.org/wiki/Bluetooth_Low_Energy. Accessed: 2024-07-18. 2024.

[23] Carles Gomez, Joaquim Oller, and Josep Paradells. “Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology”. In: *sensors* 12.9 (2012), pp. 11734–11753.

[24] Bluetooth SIG, Inc. *Topology Options*. https://www.bluetooth.com/learn-about-bluetooth/topology-options/. Accessed: 2024-07-23. 2024.

[25] Embedded Centric. *Lesson 2 – BLE profiles, services, characteristics, device roles and network topology*. https://embeddedcentric.com/lesson-2-ble-profiles-services-characteristics-device-roles-and-network-topology/. Accessed: 2024-07-23. 2024.

[26] Ashish Derhgawen. *Maximizing BLE Throughput Part 4: Everything You Need to Know*. https://punchthrough.com/ble-throughput-part-4/. Accessed: 2024-07-23. 2024.

[27] Martin Woolley. *Bluetooth® Core Specification Version 5.0 Feature Enhancements*. Version 1.1.0. Accessed: 2024-07-18. 2021.

[28] Texas Instruments. *SimpleLink CC2640R2 SDK Documentation*. Version 1.00.00.22. 2024. URL: https://software-dl.ti.com/lprf/simplelink_cc2640r2_sdk/1.00.00.22/exports/docs/blestack/html/ble-stack/index.html.

[29] Jian Yang et al. "Beyond beaconing: Emerging applications and challenges of BLE". In: *Ad hoc networks* 97 (2020), p. 102015.

[30] Kais Mekki et al. "Overview of cellular LPWAN technologies for IoT deployment: Sigfox, LoRaWAN, and NB-IoT". In: *2018 ieee international conference on pervasive computing and communications workshops (percom workshops)*. IEEE. 2018, pp. 197–202.

[31] RF Wireless World. *Sigfox Protocol Stack*. https://www.rfwireless-world.com/Tutorials/Sigfox-protocol-stack.html. Accessed: 2024-07-23. n.d.

[32] Sigfox. *5 IoT Applications to Make Our Planet Great Again*. https://www.sigfox.com/5-iot-applications-to-make-our-planet-great-again/. Accessed: 2024-07-23. n.d.

[33] Bharat S Chaudhari, Marco Zennaro, and Suresh Borkar. "LPWAN technologies: Emerging application characteristics, requirements, and design considerations". In: *Future Internet* 12.3 (2020), p. 46.

[34] Marco Centenaro et al. "Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios". In: *IEEE Wireless Communications* 23.5 (2016), pp. 60–67.

[35] Semtech. *LoRa Applications*. https://www.semtech.com/lora/lora-applications. Accessed: 2024-07-23. n.d.

[36] Sophia Antipolis Cedex. *Harmonised European Standard*. EN 302 245 V2.1.1. Accessed: 2024-07-23. European Telecommunications Standards Institute (ETSI). June 2018. URL: https://www.etsi.org/deliver/etsi_en/302200_302299/302245/02.01.01_20/en_302245v020101p.pdf.

[37] A10 Networks. *What is IoT Telemetry?* Accessed on: January 7, 2024. URL: https://www.a10networks.com/glossary/what-is-iot-telemetry/.

[38] Apostolos Gerodimos et al. "IoT: Communication protocols and security threats". In: *Internet of Things and Cyber-Physical Systems* (2023).

[39] Jakob Nielsen. *10 Usability Heuristics for User Interface Design*. Updated from original 1994 version. Jan. 2024. URL: https://www.nngroup.com/articles/ten-usability-heuristics/.

[40] Amin Ghale. *20 Inspirational Dashboard Designs*. Accessed: 2024-07-24. Oct. 2023. URL: https://www.jotform.com/blog/20-inspirational-dashboard-designs/.

[41] Bruce "Tog" Tognazzini. *First Principles of Interaction Design*. Mar. 2014. URL: https://www.nngroup.com/articles/first-principles-interaction-design/.

[42] Grow.com. *The Color-Coded Dashboard: Techniques for Improved Data Interpretation*. Accessed: 2024-07-24. Sept. 2019. URL: https://medium.com/@grow.com/the-color-coded-dashboard-techniques-for-improved-data-interpretation-047f4bfec4b4.

[43] Alan Cooper et al. *About face: the essentials of interaction design*. John Wiley & Sons, 2014.

[44] Andrea Resmini and Luca Rosati. *Pervasive information architecture: designing cross-channel user experiences*. Elsevier, 2011.

[45] Toptal Designers. *Designing a Mobile Dashboard UI: Tips and Best Practices*. Accessed: 2024-07-24. 2024. URL: https://www.toptal.com/designers/dashboard-design/mobile-dashboard-ui.

[46] Dejan Todorovic. "Gestalt principles". In: *Scholarpedia* 3.12 (2008), p. 5345.

[47] Nastengraph. *Gestalt Principles in Data Visualization*. Accessed: 2024-07-24. July 2024. URL: https://nastengraph.medium.com/gestalt-principles-in-data-visualization-a4e56e6074b5.

[48] Johannes Itten. *The elements of color*. Vol. 4. John Wiley & Sons, 1970.

[49] React Team. *React - A JavaScript library for building user interfaces*. Accessed: 2024-07-24. 2024. URL: https://react.dev/.

[50] Angular Team. *Angular - The modern web developer's platform*. Accessed: 2024-07-24. 2024. URL: https://angular.dev/.

[51] Mike Bostock and Inc. Observable. *D3.js - The JavaScript Library for Bespoke Data Visualization*. Accessed: 2024-07-24. 2024. URL: https://d3js.org/.

[52] Chart.js Contributors. *Chart.js - Open source HTML5 Charts for your website*. Accessed: 2024-07-24. 2024. URL: https://www.chartjs.org/.

[53] Semtech. *SX1261/2 Data Sheet Rev. 2.1*. Proprietary and Confidential. Semtech Corporation. Dec. 2021. URL: http://www.semtech.com/.

[54] Pallets. *Flask Documentation (3.0.x)*. Accessed: 2024-07-30. 2024. URL: https://flask.palletsprojects.com/en/3.0.x/.

[55] Linar Yusupov ehong-tl. *micropySX126X*. URL: https://github.com/ehong-tl/micropySX126X.

[56] *Microcontrolador Raspberry Pi Pico - Raspberry Pi SC0915*. URL: https://mauser.pt/catalog/product_info.php?products_id=096-9421.

[57] *Módulo SX1262 LoRa Node p/ Raspberry Pi Pico, LoRaWAN 868MHz*. URL: https://www.botnroll.com/pt/lora-868/4909-modulo-sx1262-lora-node-p-raspberry-pi-pico-lorawan-868mhz.html.

[58] Python Software Foundation. *thread — Low-level threading API*. Accessed: 2024-07-27. 2024.

[59] *Microcomputador Raspberry Pi 5 4GB - Raspberry Pi SC1111*. Accessed: 2024-07-31.

[60] *Fonte de alimentação USB-C (230VAC->5.1VDC) 3.0A 15.3W - branco - oficial para Raspberry Pi 4 - Raspberry Pi*. URL: https://mauser.pt/catalog/product_info.php?cPath=1667_2620_2960&products_id=035-3400.

[61] Eclipse Foundation. *paho-mqtt*. Accessed: 2024-07-27. 2024.

[62] Google Cloud. *Google Cloud Console*. Accessed: 2024-07-30. 2024. URL: https://console.cloud.google.com/welcome.

[63] Eclipse Foundation. *Eclipse Mosquitto*. https://mosquitto.org/. Accessed: 2024-07-27. 2024.

[64] W3Schools. *jQuery AJAX Methods*. https://www.w3schools.com/jquery/jquery_ref_ajax.asp. Accessed: 2024-07-27. 2024.

[65] Toorshia. *JustGage*. https://toorshia.github.io/justgage/. Accessed: 2024-07-27. 2024.

[66] *Antena LoRa 868MHz 2.8dBi c/ ligação SMA macho - 195mm*. Accessed: 2024-07-27.

[67] Digi International Inc. *Signal strength and the RSSI pin*. Accessed: 2024-07-30. 2021. URL: https://www.digi.com/resources/documentation/Digidocs/90001456-13/concepts/c_rssi_pin_and_signal_strength.htm.