



GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Diseño de un sistema de telemetría completo para el
equipo de formula Student de la Universidad de
Illinois – ICE (Illinois Formula Electric)

Autor: Gonzalo Barceló Álvarez

Director: Luis Francisco Sánchez Merchante

Madrid, 26 de junio de 2023

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título
Diseño de un sistema de telemetría completo para el equipo de formula Student de la
Universidad de Illinois – ICE (Illinois Formula Electric)

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Gonzalo Barceló Álvarez

Fecha: 26/ 06/2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Luis Francisco Sánchez Merchante

Fecha: 26/ 06/ 2023



GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Diseño de un sistema de telemetría completo para el
equipo de formula Student de la Universidad de
Illinois – ICE (Illinois Formula Electric)

Autor: Gonzalo Barceló Álvarez

Director: Luis Francisco Sánchez Merchante

Madrid, 26 de junio de 2023

Agradecimientos

Mostrar un especial agradecimiento a los departamentos de electrónica de los equipos de Formula Student de la universidad de ICAI (ICAI Speed Club - ISC) y de la universidad de Illinois (Illinois Formula Electric - IFE).

DISEÑO DE UN SISTEMA DE TELEMETRÍA COMPLETO PARA EL EQUIPO DE FORMULA STUDENT DE LA UNIVERSIDAD DE ILLINOIS – ICE (ILLINOIS FORMULA ELECTRIC)

Autor: Barceló Álvarez, Gonzalo.

Director: Sánchez Merchante, Luis Francisco.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este proyecto se ha diseñado e implementado una herramienta de telemetría en tiempo real para su incorporación en el formula Student de la universidad de Illinois en Urbana-Champaign. El sistema de telemetría incorpora radio LoRa, subida de datos a la nube y visualización adaptada a las necesidades de los ingenieros.

Como consecuencia, permite tener un sistema único y plenamente adaptado a las necesidades del equipo pudiendo ser modificado a voluntad del equipo sin tener que depender de licencias o empresas terceras.

Palabras clave: *IOT, sensorización, telemetría, tiempo real, API server, visualización en tiempo real, competición automovilística, Formula Student*

1. Introducción

La telemetría en los vehículos actuales es un pilar fundamental para su correcto funcionamiento, y en los coches de competición una herramienta que sirve para optimizar y mejorar los recursos de la propia máquina, siendo la fuente de datos de todos los ingenieros a la hora de tomar las decisiones más importantes sobre cambios y mejoras del propio coche.

Actualmente, en el campo de la automoción existen diversos sistemas de telemetría basados todos en el sistema de comunicación y protocolo BUS CAN. Todos ellos pretenden optimizar los recursos del vehículo y dar mayor información al usuario o piloto.

En los últimos años, podemos ver como este campo ha tenido un gran desarrollo siendo uno de los principales atractivos para los coches convencionales [1]. La telemetría se ha convertido en un sinónimo de seguridad en el coche, como de capacidad autónoma del mismo para poder tomar decisiones e incluso llevar a cabo una conducción por sí mismo.

En el mundo de la competición, su desarrollo ha sido más constante sin encontrar ningún punto de inflexión. La telemetría sigue siendo un pilar fundamental, pero hasta los grandes equipos contratan softwares y empresas de terceros para su gestión, lo cual hace ineficiente

ante cambios específicos y rápidos. En este proyecto se pretende crear una plataforma de telemetría capaz de ser adaptada a las especificaciones propias de un equipo de Formula Student.

2. Definición del proyecto

En este proyecto nos centramos en un sistema de telemetría para un vehículo de la Formula Student: su recepción, transmisión por radio, recepción por radio, almacenamiento y visualización.

A partir de los datos obtenidos por parte de los sensores del coche, se realiza una visualización *on-board* al piloto y una visualización *on-box* más completa para tomar decisiones y cambios a corto plazo. Cabe destacar que esta versión del sistema de telemetría se centra fundamentalmente en la visualización en tiempo real de los datos y su almacenamiento.

3. Descripción del modelo/sistema/herramienta

El sistema propuesto consta de diversas partes y etapas para el correcto funcionamiento. En primer lugar, los datos deben ser recogidos por los sensores y procesados por las distintas ECUS. Este proceso no se recoge en este proyecto. Una vez los datos se encuentran en las distintas ECUS son transmitidos a la ECU de telemetría para su almacenamiento, procesado, visualizado y transmisión por radio. Todas estas comunicaciones internas dentro del vehículo son gracias al uso del protocolo de comunicaciones BUS CAN (software) y el uso de MCP-2515 (hardware)[2].

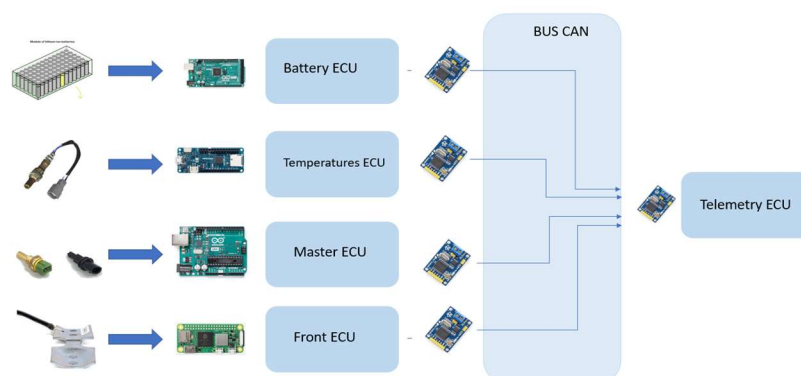


Figura 1: Primera etapa: recogida y transmisión de los datos desde las distintas ECUs hasta la Ecu de telemetría

Una vez que los datos ya se encuentran en la ECU de telemetría, se lleva a cabo la segunda etapa, el procesado en local (dentro del propio vehículo), su transmisión y visualización en el *dashboard* del piloto.

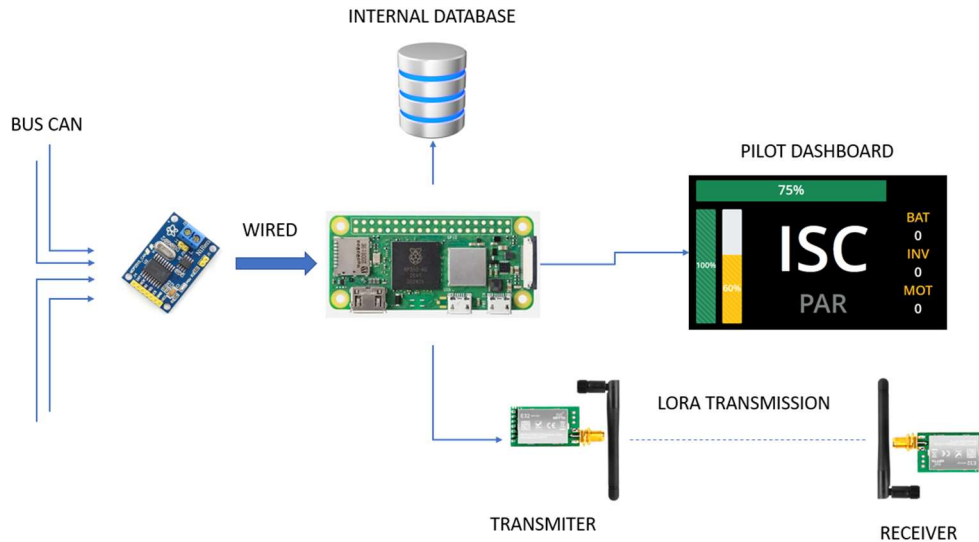


Figura 2: Segunda etapa; procesado, almacenado y transmisión de los datos

En la última etapa, se lleva a cabo la recepción, almacenamiento en la nube y la visualización en tiempo real de los parámetros más relevantes y amplios de los que el piloto puede interpretar. Algunos de los datos analizados en el muro por los ingenieros son el estado de las baterías, su temperatura y voltaje, la velocidad, el par de los pedales o el estado de la señal que se está transmitiendo.

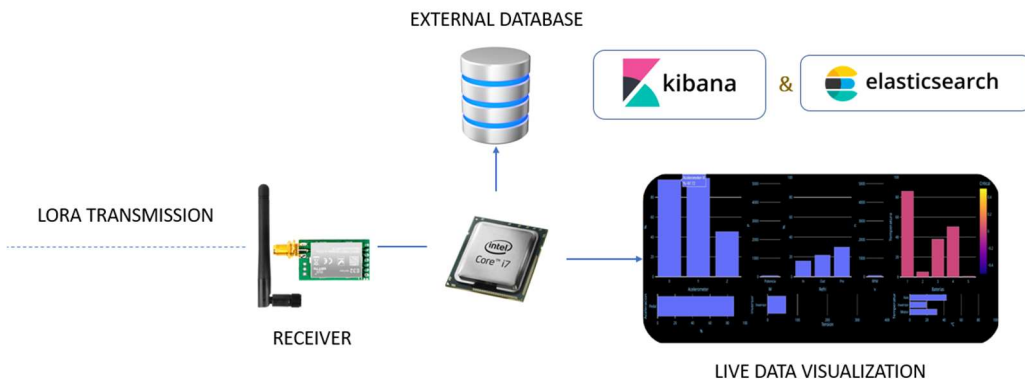


Figura 3: Tercera etapa; recepción, visualización y almacenado

El objetivo final del sistema de telemetría es la visualización y el estudio de las señales captadas por los sensores para poder actuar conforme a sus lecturas y optimizar los recursos.

4. Resultados

Los resultados son obtenidos durante el transcurso de la ejecución de todo el sistema. Son principalmente gráficas, mensajes al piloto y la toma de decisiones relacionada con los datos evaluados.

Un ejemplo puede ser la medición del voltaje de las celdas sin necesidad de abrir el acumulador (batería) del vehículo para poder determinar si el formula está lo suficientemente cargado para poder soportar una prueba.

5. Conclusiones

A modo de conclusión, podemos determinar que el modelo que se propone garantiza un sistema de seguimiento y análisis en tiempo real para equipos que no tienen medios o tiempo suficiente como para poder desarrollarlo. Todo este seguimiento proporciona seguridad, prevención ante costes extras y mejora de rendimiento al vehículo, piloto y equipo.

6. Referencias

[1] <https://www.facebook.com/DriveSmart.ES>, «DriveSmart | La telemetría en la F1. Clave para el ingeniero, trampa para el piloto», *Website de :DriveSmart*. <https://drive-smart.com/es/blog/2015/07/22/la-telemetria-en-la-f1-clave-para-el-ingeniero-trampa-para-el-piloto/> (accedido 14 de junio de 2023).

[2] strange_v, «Car Telemetry». 11 de abril de 2022. Accedido: 14 de junio de 2023. [En línea]. Disponible en: <https://github.com/strange-v/car-telemetry>

[3] «Especial. Estado del coche autónomo: dónde estamos y hacia dónde vamos», *forocheselectricos*, 16 de septiembre de 2018. <https://forocheselectricos.com/2018/09/estado-del-coche-autonomo-donde-estamos-y-hacia-donde-vamos-i.html> (accedido 14 de junio de 2023).

DESIGN OF A COMPLETE TELEMETRY SYSTEM FOR THE ILLINOIS FORMULA STUDENT TEAM AT THE UNIVERSITY OF ILLINOIS - ICE (ILLINOIS FORMULA ELECTRIC).

Autor: Barceló Álvarez, Gonzalo.

Director: Sánchez Merchante, Luis Francisco.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

PROJECT SUMMARY

This project has designed and implemented a real-time telemetry tool for incorporation into the formula Student at the University of Illinois at Urbana-Champaign. The telemetry system incorporates LoRa radio, data upload to the cloud and visualization adapted to the needs of the engineers.

As a result, it allows for a unique system that is fully adapted to the team's needs and can be modified at the will of the team without having to rely on licenses or third-party companies.

Keywords: *IOT, sensorization, telemetry, real time, API server, real time visualization, automated competition, Formula Student*

1. Introduction

Telemetry in today's vehicles is a fundamental pillar for their correct operation, and in racing cars it is a tool that serves to optimize and improve the resources of the machine itself, being the source of data for all engineers when making the most important decisions about changes and improvements to the car itself.

Currently, in the automotive field there are several telemetry systems based on the CAN BUS communication system and protocol. All of them aim to optimize the vehicle's resources and provide more information to the user or driver.

In recent years, we can see how this field has had a great development being one of the main attractions for conventional cars [1]. Telemetry has become a synonym for safety in the car, as well as for the car's autonomous capacity to make decisions and even drive itself.

In the world of racing, its development has been more constant without encountering any turning point. Telemetry is still a fundamental pillar, but even the big teams hire third-party

software and companies to manage it, which makes it inefficient in the face of specific and rapid changes. This project aims to create a telemetry platform capable of being adapted to the specifications of a Formula Student team.

2. Project definition

In this project we focus on a telemetry system for a Formula Student car: its reception, radio transmission, radio reception, storage, and visualization.

From the data obtained from the car's sensors, an on-board visualization is provided to the driver and a more complete on-box visualization to make decisions and changes in the short term. It should be noted that this version of the telemetry system focuses primarily on real-time data visualization and storage.

3. Model description

The proposed system consists of several parts and stages for proper operation. First, data must be collected by the sensors and processed by the various ECUs. This process is not covered in this project. Once the data is in the various ECUs, it is transmitted to the telemetry ECU for storage, processing, display, and radio transmission. All these internal communications within the vehicle are thanks to the use of the BUS CAN communications protocol (software) and the use of MCP-2515 (hardware) [2].

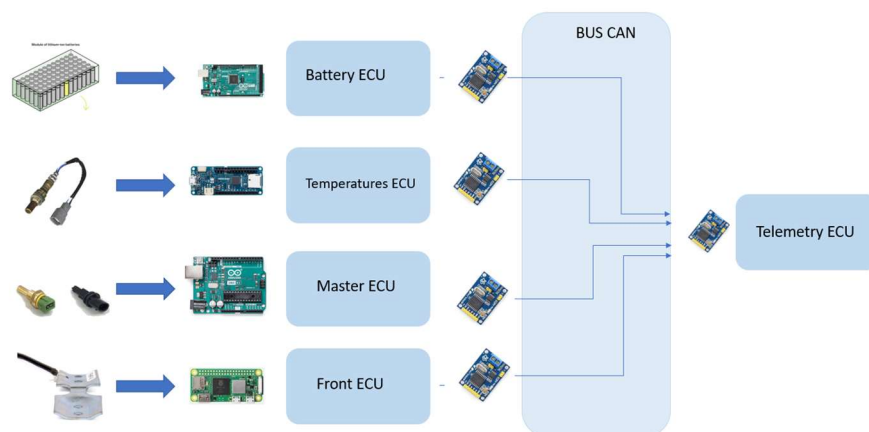


Image 1: First stage: pick and transmit data from ECUs to Telemetry ECU

Once the data is already in the telemetry ECU, the second stage, local processing (inside the vehicle itself), transmission and display on the driver's dashboard, takes place.

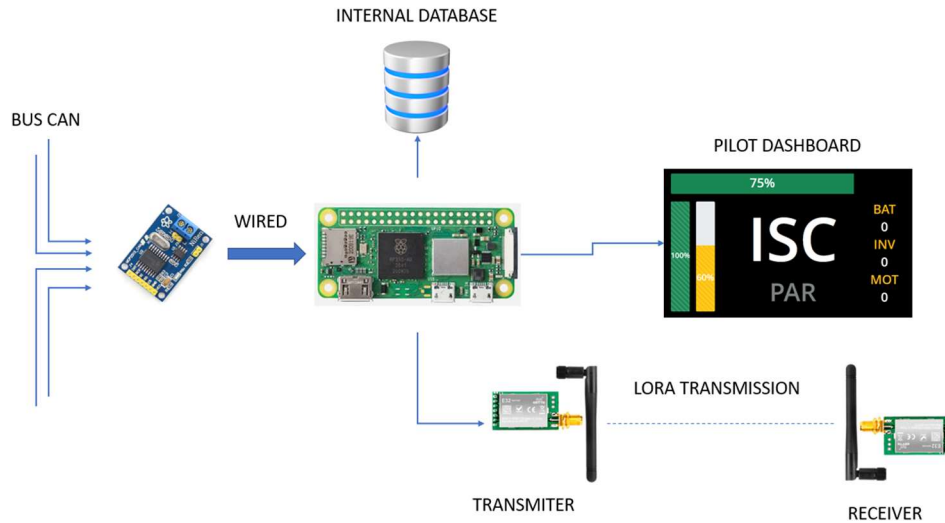


Image 2: Second stage: process, store and transmission

In the last stage, the most relevant and extensive parameters that the rider can interpret are received, stored in the cloud and displayed in real time. Some of the data analysed on the wall by the engineers are the state of the batteries, their temperature and voltage, the speed, the torque of the pedals or the state of the signal being transmitted.

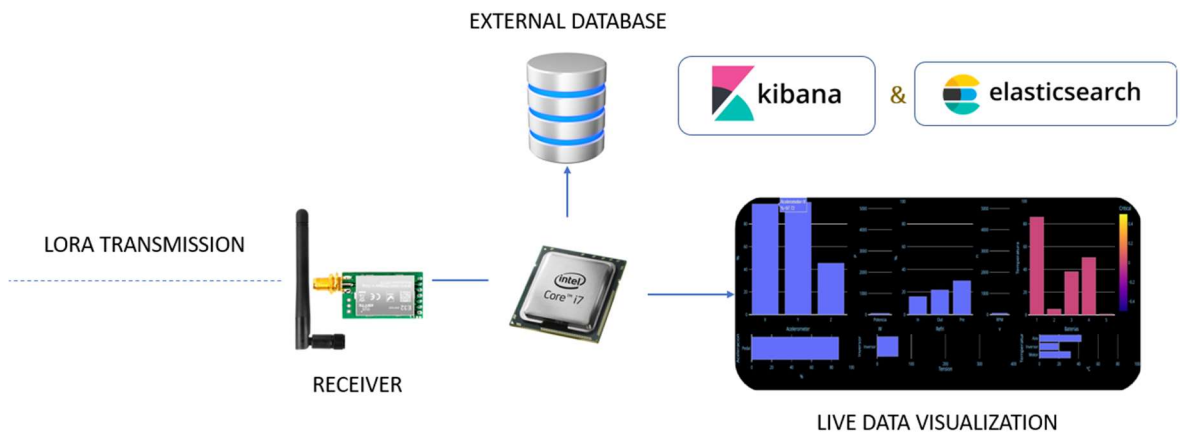


Image 3: Third stage: reception, visualization, and store

The ultimate goal of the telemetry system is to visualize and study the signals captured by the sensors in order to act on their readings and optimize resources.

4. Results

The results are obtained during the course of the execution of the whole system. They are mainly graphs, messages to the driver and decision making related to the evaluated data.

An example could be the measurement of the cell voltage without opening the vehicle's accumulator (battery) in order to determine whether the formula is sufficiently charged to withstand a test.

5. Conclusions

In conclusion, we can determine that the proposed model guarantees a real-time monitoring and analysis system for teams that do not have sufficient means or time to be able to develop it. All this monitoring provides safety, prevention of extra costs and improved performance for the vehicle, driver and team.

6. References

[1] <https://www.facebook.com/DriveSmart.ES> , «DriveSmart | La telemetría en la F1. Clave para el ingeniero, trampa para el piloto», *Website de :DriveSmart*. <https://drive-smart.com/es/blog/2015/07/22/la-telemetria-en-la-f1-clave-para-el-ingeniero-trampa-para-el-piloto/> (accedido 14 de junio de 2023).

[2] strange_v, «Car Telemetry». 11 de abril de 2022. Accedido: 14 de junio de 2023. [En línea]. Disponible en: <https://github.com/strange-v/car-telemetry>

[3] «Especial. Estado del coche autónomo: dónde estamos y hacia dónde vamos», *forocheselectricos*, 16 de septiembre de 2018. <https://forocheselectricos.com/2018/09/estado-del-coche-autonomo-donde-estamos-y-hacia-donde-vamos-i.html> (accedido 14 de junio de 2023).

Índice de la memoria

Capítulo 1. Introducción	7
1.1 Motivación del proyecto.....	8
Capítulo 2. Descripción de las Tecnologías.....	9
2.1 BUS CAN.....	9
2.2 NodeJs	11
2.2.1 NodeJs en la recepción de datos y apertura de puertos.....	12
2.2.2 NodeJs en la subida de los valores a la base de datos.....	12
2.2.3 NodeJs en transmisión al frontend.....	12
2.2.4 NodeJs como interfaz	13
2.3 ARDUINO	13
2.4 Base de datos relacional (DB2).....	15
2.5 Spring boot (webserver).....	17
2.6 PostgreSQL	18
2.6.1 pgADMIN4	18
Capítulo 3. Estado de la Cuestión.....	20
Capítulo 4. Definición del Trabajo	35
4.1 Justificación.....	35
4.2 Objetivos	37
4.3 Metodología.....	38
4.4 Organización y Estimación Económica.....	38
Capítulo 5. Sistema/Modelo Desarrollado.....	41
5.1 análisis del sistema	41
5.1.1 Análisis del protocolo interno	41
5.1.2 Análisis del dashboard on-board	43
5.1.3 Análisis de la transmisión inalámbrica del vehículo.....	46
5.1.4 Análisis del lenguaje utilizado para la visualización de los datos en tiempo real.....	50
5.2 Diseño del sistema.....	52
5.2.1 Ordenador a bordo.....	53

5.2.2 envío y recepción de datos	56
5.2.3 Diseño del tratamiento de los datos en la recepción, almacenado y visualización en tiempo real.....	59
5.2.4 Diseño del módulo de predictive maintenance	63
5.3 Implementación.....	70
5.3.1 Limitaciones en la implementación en el monoplaza de la Universidad de Illinois	70
5.3.2 Desarrollo entorno de pruebas.....	71
5.3.3 Implementación simulada del dashboard del piloto.....	73
5.3.4 Implementación simulada de la recepción	78
Capítulo 6. Análisis de Resultados.....	86
6.1 Correcto funcionamiento de peticiones y almacenamiento instantáneo de los valores del vehículo.....	86
6.2 Visionado en tiempo real de los datos y renderizado de las gráficas	88
6.3 Acceso de los datos desde la red interna del servidor receptor de datos por parte de cualquier usuario.....	89
Capítulo 7. Conclusiones y Trabajos Futuros.....	90
7.1 Limitaciones en la implementación.....	91
7.1.1 Formato real del dato.....	91
7.1.2 Falta de sensorización real	91
7.1.3 Identificación de los sensores.....	92
7.1.4 Tasas de actualización de los datos	93
7.1.5 Limitación de rango	93
7.1.6 Limitación computacional.....	95
7.2 Trabajos futuros.....	97
7.2.1 Creación de interfaz de usuario para acceso seguro y cifrado en la base de datos	97
7.2.2 Integración en un monoplaza	97
7.2.3 Integración en la industria	98
7.2.4 Creación del módulo de predictive maintenance a través de valores reales	98
Capítulo 8. Bibliografía.....	101
ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS	109
ANEXO II 111	

Índice de figuras

Figura 1: Primera etapa: recogida y transmisión de los datos desde las distintas ECUs hasta la Ecu de telemetría	8
Figura 2: Segunda etapa; procesado, almacenado y transmisión de los datos	9
Figura 3: Tercera etapa; recepción, visualización y almacenado	9
Figura 4: ISO 11898: the definition of the standard CAN bus (1986)	9
Figura 5: Hardware y estructura BUS CAN actual	10
Figura 6: Arduino Uno: placa utilizada para recrear los valores	15
Figura 7: Primer display de valores de telemetría	23
Figura 8: Primer titular donde se menciona el uso de la telemetría en los pitlanes.....	23
Figura 9: Dashboard F1 1960	25
Figura 10: Podium Connect MK2 - RaceCapture	30
Figura 11: Visualización de los datos desde Pi Toolbox.....	31
Figura 12: Bosch Motorsports Telemetry - MoTec.....	31
Figura 13: Visualización del display de Dash2	32
Figura 14: Comparativa entre las diferentes tecnologías.....	34
Figura 15: Planificación semanal del proyecto y redacción del TFG.....	39
Figura 16: Modelo de hardware interno del vehículo y conexionado principal.....	43
Figura 17: Dashboard on-board.....	45
Figura 18: Efecto Doppler	48
Figura 19: Módulos de transmisión LoRa	50
Figura 20: MCP25-15 CAN bus module.....	52
Figura 21: Feedback recibido por parte de los pilotos	55
Figura 22: Tipos de error en baterías	68
Figura 23: Definición de la tensión de flotación	69
Figura 24: definición del punto de cambio.....	69
Figura 25:Funcionamiento de la predicción del punto de cambio	69
Figura 26: Pinout Arduino Zero	73
Figura 27: Almacenamiento instantáneo en la base de datos db2 controlada por pgAdmin87	

Figura 28: Correcta conexión entre base de datos y frontend	87
Figura 29: Valores modificados en tiempo real	88
Figura 30: Generación de graficas en tiempo real.....	89
Figura 31: Acceso remoto a través de la red local o Internet	89
Figura 32: Sistema propuesto	92

Índice de tablas

Tabla 1: Reparto provisional de identificadores de sensor.....	72
Tabla 2: Almacenado de los datos en la base	82

Capítulo 1. INTRODUCCIÓN

Todos los vehículos actuales constan de algún tipo de sistema de telemetría. Cada grupo desarrolla sus propios sistemas y gestiona sus vehículos. En los últimos años, con el surgimiento de la conducción asistida y conducción autónoma se ha vuelto uno de los campos con mayor desarrollo dentro de la industria del automóvil. Compañías como Tesla, Mercedes o Volkswagen se sitúan como las más punteras dentro del campo siendo estos grupos los más potentes del sector [3]. Todos estos sistemas están fuertemente protegidos por medio de patentes y secretos de compañía siendo prácticamente imposible integrar uno de estos sistemas en vehículos propios.

En este proyecto no se pretende aportar soluciones nuevas al sector comercial del automóvil si no al sector de la competición, aunque todas las soluciones planteadas pueden ser adaptadas para todo tipo de proyectos y vehículos.

En el sector de los formula, o vehículos de competición, estos sistemas de telemetría se pueden encontrar en dos grupos diferentes.

El primero, grandes marcas con equipos en diferentes competiciones, como Audi, presentes en el Dakar, formula 1 (2025) y formula E entre otros, que desarrollan sus propios softwares, protocolos y sistemas propios con el objetivo de desempeñar la misma función.

En el segundo grupo, encontramos pequeños fabricantes en competiciones inferiores que deben asumir gastos y, por regla general, no pueden costearse un equipo adaptado a las necesidades propias. Estos últimos deben confiar sus sistemas a productos producidos por terceros, marcas cuyo único propósito es el de realizar sistemas de telemetría genéricos que sirvan para el mayor número de proyectos. El principal problema de estas es que su adaptabilidad al proyecto suele ser extremadamente limitada.

Con este proyecto se pretende aportar una alternativa en este sector, distribuyendo un software y recursos los cuales hacen que, sin apenas modificaciones, podamos contar con un

sistema de telemetría completo sin tener que adaptar nuestro proyecto a marcas de terceros ni teniendo que invertir gran capital.

1.1 MOTIVACIÓN DEL PROYECTO

Tras haber participado en competiciones de Formula Student, y haber conocido a equipos de todas partes del mundo, Egipto, Francia, Italia, Israel entre otros, hemos destacado que pocos o casi ningún equipo de tamaño pequeño o mediano (equipos los cuales no tienen gran presupuesto) tienen un sistema funcional o útil de telemetría. Todos estos equipos corren a ciegas durante todas las sesiones y esperan al final de las mismas para poder recoger los datos y tomar decisiones. Esto se convierte en un factor diferenciador con el resto de los equipos capaces de modificar y adaptar el coche sin necesidad de terminar la sesión y desmontar el almacenamiento de los datos.

La importancia de este proyecto reside en dos pilares fundamentales, un gran valor tecnológico a los equipos de la competición y la obtención de un sistema de costoso valor altamente adaptativo que prevenga errores y optimice los recursos.

Este sistema permite realizar un procesado masivo de todos los sensores del coche y su visualización en tiempo real reduciendo errores de gran valor económico en equipos de ajustado presupuesto. Además, se pretenden añadir herramientas de prevención y previsión de desgaste como un estudio del desgaste de baterías o evolución de temperaturas a futuro con el mismo motivo: aportar fiabilidad al monoplaza.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Para facilitar la lectura y comprensión de este proyecto, vamos a comentar las tecnologías de las que se harán uso para la realización del sistema de telemetría. Cabe destacar que muchas de ellas son popularmente utilizadas en el campo de la automoción, software o radiación de señales.

2.1 BUS CAN

El inicio de la tecnología BUS CAN surge cuando los elementos que formaban parte del monoplaza pasaron de ser analógicos a digitales y con ello comenzó a hacerse muy complejo un cableado único para cada sistema. Fue entonces cuando en 1982 se decide hacer un protocolo de comunicaciones único para el mundo de la automoción, consiguiendo así una simplificación significativa. Este protocolo nace con el nombre de CAN bus CAN (del inglés Controlled Area Network), desarrollado por la empresa BOSCH y fue lanzado oficialmente en 1986. Un año más tarde empresas como Intel y Philips sacaron los primeros controladores.

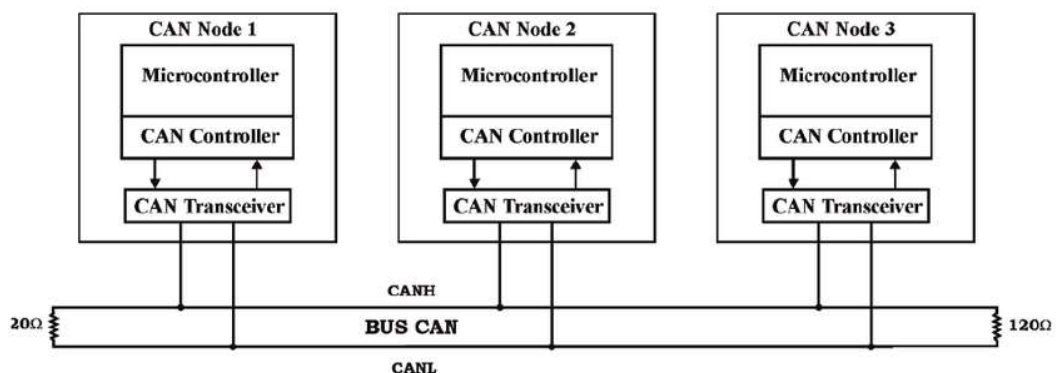


Figura 4: ISO 11898: the definition of the standard CAN bus (1986)

Después de múltiples versiones de BOSCH del protocolo, en 1991 se publica la última versión del protocolo y actual, CAN 2.0, con dos formatos. Un formato estándar, A y un

formato extendido, B cuya única diferencia es la cantidad de bits asociados a la identificación del mensaje, 11 bits y 29 bits respectivamente.

Con este contexto de comunicación entre los diferentes sistemas digitales y de simplificación del cableado, la marca Mercedes Benz lanza en 1991 [4] el primer vehículo comercializado que incorpora protocolo bus CAN, con 5 nodos de comunicación.

El funcionamiento básico del sistema de comunicaciones de los vehículos, como ya hemos visto, está basado en el protocolo CAN. En este protocolo, todos los participantes son potenciales emisores y receptores de los mensajes, es decir, todos los paquetes son recibidos por los integrantes. Estos paquetes de bytes son utilizados por motores, inversores y resto de sistemas ya que es el principal sistema de comunicación, por lo tanto, será el protocolo que deberemos utilizar en nuestro sistema de telemetría a la hora de poder recibir y registrar los valores de todo el sistema. Cada paquete viene precedido por una identificación única la cual indica el tipo de valor que se está recibiendo (temperatura, par motor, par pedal, etc....). Estos bytes previos se denominan ID del mensaje.

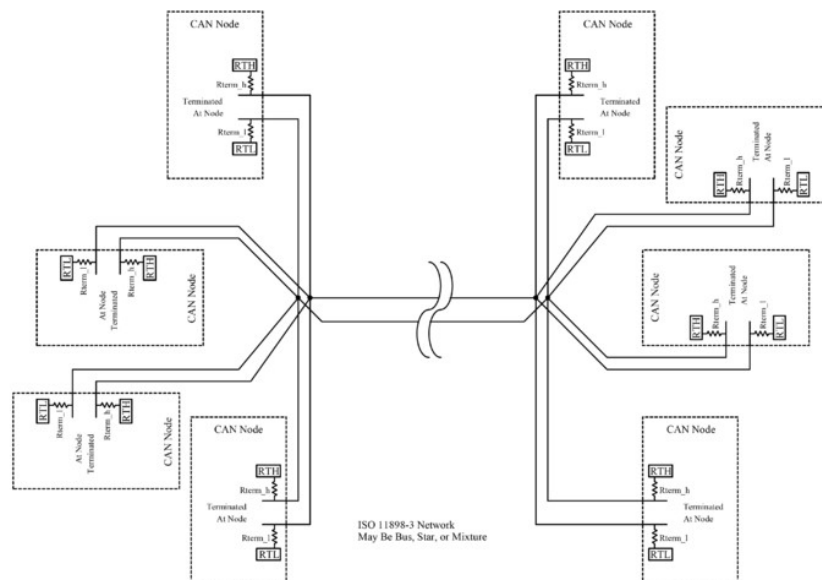


Figura 5: Hardware y estructura BUS CAN actual

El protocolo CAN, como se ha especificado previamente, puede diferenciarse en CAN 2.0A o CAN 2.0B. Estos se diferencian en esta última parte, en los bits que definen el ID del mensaje, pudiendo tener 2048 identificadores diferentes para el protocolo CAN2.0A y más de medio millón de posibles identificadores para el can extendido o CAN 2.0B [5].

El sistema de telemetría se integra en la comunicación CAN principalmente como receptor, es decir, el sistema solo escucha los comandos, órdenes a motores y temperaturas.

Con las primeras aplicaciones del BUS CAN surgen los primeros sistemas de telemetría digital aplicados a la automoción y maquinaria. Como suele ocurrir en toda industria, las primeras apariciones y las tecnologías más novedosas siempre surgen en un contexto de competición y reto por parte de los ingenieros. Por lo tanto, en este caso, estas mejoras las encontramos en la competición más importante de la historia del automovilismo como es la fórmula 1.

2.2 NODEJS

Para el procesado, recepción y transmisión de los datos al *backend* y al servidor, se hará uso de NodeJs. NodeJs es una herramienta que funciona en un entorno web desarrollada en implementada en JavaScript [6]. En los últimos años ha ganado bastante popularidad por las múltiples aplicaciones y versatilidad a la hora de poderse aplicar en múltiples entornos. Cabe destacar que NodeJs es open source, lo cual significa que es la propia comunidad la que implementa las mejoras. Esto hace que siempre esté actualizada con las últimas versiones y que los problemas que puedan surgir son resueltos por las personas que lo han desarrollado o por personas que están trabajando con la misma versión.

El uso de NodeJs para este proyecto, como se ha especificado previamente es el de procesar la recepción de los datos del monoplaza, su recepción y transmisión a diferentes servidores, *frontends*, etc.

Para ver el enorme potencial de NodeJs en el proyecto, a continuación, se describen detalladamente las funcionalidades que tiene en este caso específico.

2.2.1 NODEJS EN LA RECEPCIÓN DE DATOS Y APERTURA DE PUERTOS

En primer lugar, a través de las librerías y funciones de NodeJs, somos capaces de abrir un puerto del dispositivo que está recibiendo los datos del receptor de la radio. Esto nos permite leer todos los datos que se están enviando desde el monoplaza y poder mostrarlos por consola. Cabe destacar que los datos recibidos necesitan un procesado, ya que, para que cumplan las especificaciones de la radio, deben ser tratados y adaptados al canal por el que se van a transmitir. Una vez hecha la conversión, se procede al almacenamiento en local y en remoto mediante peticiones a un servidor.

2.2.2 NODEJS EN LA SUBIDA DE LOS VALORES A LA BASE DE DATOS

Previo a la subida de los datos será necesaria la creación de un servidor el cual se mantenga a la escucha y tenga conectado internamente una base de datos, la cual almacenará todos los valores

El servidor puede ser una base de datos en local (localhost:8080) o un servidor el cual tenga el dominio y las autorizaciones pertinentes el equipo en cuestión. Estos servidores pueden ser productos de Google como Google Cloud, Amazon como AWS o ElasticSearch.

Este apartado será más desarrollado en la explicación detallada de la base de datos y del servidor. Por parte de NodeJs, la función que desarrolla en esta etapa es la de realizar las peticiones para guardar los datos recibidos y procesados.

2.2.3 NODEJS EN TRANSMISIÓN AL FRONTEND

La transmisión al *frontend* es el intercambio de datos entre el receptor de los datos y la interfaz encargada de mostrar los datos, calcular las gráficas, generar alertas, etc. El desempeño de NodeJs a la hora de realizar esta tarea es el de crear un servidor interno que sirva como canal de comunicaciones entre el *backend* (receptor) y el *frontend* (interfaz).

Este servidor interno cumple múltiples funciones, como la de crear el objeto que intercambiará con la interfaz, controlará la velocidad de actualización de la interfaz o creará

los archivos web a los que deberá conectarse el usuario para que pueda hacer uso de los recursos gráficos de NodeJs, muchas veces necesarios para la visualización de los datos.

2.2.4 NODEJS COMO INTERFAZ

Además de contar con gran cantidad de funcionalidades a la hora de recibir, procesar y transmitir mediante puertos, servidores, peticiones o servidores internos, NodeJs además cuenta con gran cantidad de recursos gráficos que son de gran ayuda a la hora de mostrar datos en tiempo real o de poder mostrar datos con estilos similares a los utilizados en el mundo de la automoción.

2.3 ARDUINO

Arduino es una plataforma open-source dedicada al ámbito de la electrónica que permite a sus usuarios diseñar y realizar soluciones en el ámbito de la electrónica mediante creaciones propias, muchas de estas utilizando su función de open-source.

Su principal cometido es diseñar, fabricar y admitir dispositivos electrónicos y software, lo cual permite el rápido acceso a tecnologías avanzadas y potentes.

En esta tesis su uso ha sido fundamental ya que, por motivos ajenos a este proyecto, no se ha podido llevar a cabo las pruebas en un entorno real. Bajo estas condiciones, los Arduinos han servido como entornos de prueba para recrear todos los posibles entornos, valores y frecuencias

Estos Arduinos han sido utilizados como transmisores de datos y receptores, recreando de la manera más fiel posible, el funcionamiento del monoplaza y generando valores pseudo reales.

El uso de Arduino para replicar entornos puede referirse a varias aplicaciones, como la creación de modelos a escala de sistemas ambientales o la creación de simuladores para el entrenamiento de pilotos y conductores.

Los Arduino pueden utilizarse para replicar entornos:

1. Simuladores de vuelo: los Arduino pueden utilizarse para crear simuladores de vuelo realistas que permitan a los usuarios practicar el pilotaje de aeronaves. Estos simuladores pueden incluir una variedad de componentes, como paneles de control, pedales y joystick, que se conectan a un ordenador y se programan para simular la experiencia de vuelo. Un ejemplo de esto es el LINK2FS, el cual utiliza el hardware directamente conectado al ARDUINO para replicar la sensación y mandos de un pilotaje real [7].
2. Modelos a escala de sistemas ambientales: los Arduino también se utilizan para replicar entornos naturales y crear modelos a escala de sistemas ambientales, como acuarios o terrarios. Los sensores conectados a los Arduino pueden monitorizar la temperatura, la humedad y otros factores ambientales y controlar dispositivos como ventiladores y bombas para mantener un equilibrio ambiental. Un ejemplo es el Sistema de monitorización ambiental basado en hardware open-source alimentado mediante energía fotovoltaica el cual utiliza la combinación de Arduino y RaspberryPi para conseguirlo [8].
3. Simuladores de conducción: los Arduino también se pueden utilizar para crear simuladores de conducción para el entrenamiento de conductores o sistemas de telemetría. Los simuladores pueden incluir una variedad de componentes, como sensores tales como pedales, pantallas o galgas extensiométricas, que se conectan a la corriente y se programan para simular la experiencia de conducción. Prueba de ello es Arduino UNO desarrollado por Jordi Rafart y Marc Tomàs que crearon un simulador de carreras [9].

Como conclusión, este hardware permite a los usuarios crear sistemas interactivos y automatizados que pueden simular una amplia variedad de entornos y sistemas. Los Arduino son una plataforma de programación de bajo costo y fácil de usar, lo que los hace ideales para proyectos de bricolaje y para la educación en ciencias y tecnología.



Figura 6: Arduino Uno: placa utilizada para recrear los valores

2.4 BASE DE DATOS RELACIONAL (DB2)

DB2 es un sistema de gestión de bases de datos relacionales desarrollado por IBM. Este software se utiliza comúnmente en servidores para almacenar grandes cantidades de información y gestionar su acceso y uso de manera eficiente.

Una base de datos relacional se compone de tablas que contienen datos relacionados. Cada tabla consta de filas y columnas, donde cada columna representa un campo de datos y cada fila representa un registro. Las tablas están relacionadas entre sí por medio de claves, lo que permite acceder a la información de forma organizada y eficiente.

DB2 es un sistema de base de datos altamente escalable y robusto que puede manejar grandes cantidades de datos y soportar una gran cantidad de usuarios y aplicaciones simultáneas. El software DB2 se utiliza comúnmente en servidores para almacenar información empresarial crítica, como información de clientes, historiales de transacciones y registros financieros.

Algunas de las características y capacidades de DB2 incluyen:

1. Multiplataforma: DB2 está disponible en una variedad de plataformas, lo que permite su uso en una amplia gama de entornos empresariales.

2. Seguridad: DB2 cuenta con herramientas de seguridad robustas que permiten la gestión de acceso y permisos a los datos, garantizando que solo los usuarios autorizados puedan acceder a la información.
3. Alta disponibilidad: DB2 cuenta con herramientas de alta disponibilidad que garantizan que los datos estén disponibles en todo momento, incluso en caso de fallo del sistema o de pérdida de datos.
4. Escalabilidad: DB2 es altamente escalable y puede manejar grandes cantidades de datos y usuarios simultáneos.

Las características y capacidades de DB2 lo hacen una opción popular para empresas que requieren un sistema de gestión de bases de datos fiable y seguro.

Por otro lado, el uso de bases de datos por parte de servidores para almacenar datos de un cliente y disponer de ellos cuando lo necesite un tercero se ha convertido en una práctica común en la actualidad. Esto se debe a que permite a las empresas almacenar grandes cantidades de datos de manera eficiente y gestionarlos de manera más efectiva.

Un ejemplo común de esto es cuando un cliente realiza una compra en línea en una tienda en línea. La información del cliente y la transacción se almacenan en una base de datos en el servidor de la tienda. Si el cliente necesita devolver un artículo o si se produce algún problema con la transacción, la tienda puede acceder a la información almacenada en la base de datos para solucionar el problema.

Además, la información almacenada en la base de datos puede ser utilizada por terceros, como proveedores de servicios de pago o empresas de envío, para procesar la transacción. Estos terceros pueden acceder a la información almacenada en la base de datos a través de una API (Interfaz de Programación de Aplicaciones) para procesar la transacción y realizar las operaciones necesarias.

La utilización de bases de datos en servidores también permite a las empresas gestionar grandes cantidades de información de manera más eficiente.

En este ejemplo, la base de datos será utilizada de manera completamente privada y con seguridad en la escritura de los datos. Esto permitirá no depender de aplicaciones de terceros y también ayuda en no incrementar los costes de producción del sistema de telemetría.

2.5 SPRING BOOT (WEBSERVER)

Spring Boot es un marco de trabajo (framework) de código abierto basado en Java [10] que se utiliza para crear aplicaciones web y de microservicios. Fue desarrollado por Pivotal Software en 2012 y se basa en el marco de trabajo Spring Framework.

Se enfoca en la simplicidad, permitiendo a los desarrolladores crear aplicaciones rápidamente y sin la necesidad de realizar una gran cantidad de configuración. Esto se logra gracias a que Spring Boot incluye una serie de configuraciones predefinidas y una estructura de proyecto sencilla que facilita la creación de aplicaciones.

Entre las principales características de Spring Boot se encuentran:

1. **Autoconfiguración:** cuenta con un mecanismo de autoconfiguración que permite a las aplicaciones ajustarse automáticamente en función de las dependencias y el entorno de ejecución.
2. **Inyección de dependencias:** gestión de las dependencias de los componentes de la aplicación.
3. **Integración con bases de datos:** proporciona soporte para la integración con bases de datos relacionales y NoSQL, lo que facilita la creación de aplicaciones que requieren acceso a bases de datos. Esta cualidad será muy utilizada en este proyecto.
4. **Soporte para servicios web:** añade soporte para la creación de servicios web RESTful y SOAP, lo que facilita la creación de aplicaciones que requieren integración con otros sistemas como el sistema propuesto en este proyecto.
5. **Administración remota:** incluye una consola de administración remota que permite a los desarrolladores supervisar y gestionar la aplicación en tiempo de ejecución.

6. Integración con otras bibliotecas de Spring: se integra fácilmente con otras bibliotecas de Spring, como Spring Data y Spring Security, lo que permite una mayor flexibilidad en el desarrollo de aplicaciones.

En este proyecto, su relevancia reside en ser el framework que se utilizará para el desarrollo del servidor, la gestión de peticiones, el *frontend* y resto de funciones que se esperan de una base de datos online.

2.6 POSTGRESQL

PostgreSQL es un sistema de gestión de bases de datos relacionales de alto rendimiento de código abierto. Utiliza el modelo de datos relacional para organizar los datos en tablas con filas y columnas y permitir la definición de relaciones utilizando claves primarias y foráneas [11].

Este sistema se destaca por su escalabilidad, extensibilidad, replicación, integridad transaccional, control de concurrencia y soporte para consultas complejas y avanzadas. Aunque usa el lenguaje de consulta estructurado SQL estándar, también tiene extensiones y características avanzadas como funciones definidas por el usuario, subconsultas, vistas, triggers y procedimientos almacenados.

El soporte de transacciones ACID de PostgreSQL agrupa operaciones en transacciones y garantiza que se cumplan las propiedades ACID para garantizar la integridad transaccional. Además, implementa mecanismos de control de concurrencia para permitir que múltiples transacciones funcionen al mismo tiempo sin robar datos.

2.6.1 PGADMIN4

PgAdmin es una herramienta de administración y desarrollo de bases de datos PostgreSQL, un sistema de gestión de bases de datos relacional de código abierto y de alto rendimiento. Es el entorno más popular para administrar bases de datos PostgreSQL y proporciona una interfaz gráfica de usuario (GUI) intuitiva para administrar, diseñar y desarrollar bases de datos PostgreSQL [12].

Las principales características de PgAdmin aplicadas en este proyecto son:

- **Funciones de administración:** ofrece una amplia gama de funciones de administración para gestionar bases de datos PostgreSQL. Adaptan tareas como la creación, modificación y eliminación de bases de datos, tablas, índices, vistas, funciones y secuencias. También puedes realizar copias de seguridad y restauraciones de bases de datos, administrar permisos de usuario y realizar seguimiento de las actividades en el servidor.
- **Editor SQL:** incluye un editor SQL integrado que te permite escribir y ejecutar consultas SQL directamente en la interfaz. Puedes trabajar con múltiples pestañas de consulta, resaltar la sintaxis SQL, autocompletar y analizar consultas para encontrar posibles errores.
- **Diseñador de bases de datos:** proporciona un diseñador visual de bases de datos que te permite crear y modificar el esquema de la base de datos utilizando una interfaz gráfica. Puedes crear tablas, definir relaciones entre tablas, establecer restricciones y generar scripts SQL correspondientes.
- **Importación y exportación de datos:** PgAdmin permite importar y exportar datos desde y hacia diferentes formatos, como CSV, Excel, JSON, etc. Puedes cargar datos en tablas existentes o crear tablas nuevas a partir de los datos importados.

Capítulo 3. ESTADO DE LA CUESTIÓN

En primer lugar, previo a la solución planteada en este proyecto, debemos saber en qué posición se encuentra el sector en el que vamos a desarrollarlo.

Para comenzar, debemos ampliar el foco y fijarnos en las soluciones históricas en el ámbito de la electrónica que se han planteado en el sector de las competiciones automovilísticas ya que las competiciones de Formula Student no tienen historia suficiente como para poder dar un contexto amplio y completo.

La telemetría electrónica comenzó a desarrollarse en la década de 1950, cuando los científicos y los ingenieros necesitaban medir y transmitir datos desde satélites y naves espaciales a la Tierra. Se utilizó un sistema de transmisión de radio para enviar los datos desde el espacio a la Tierra, lo que permitió a los científicos monitorear y controlar los sistemas de manera remota. En este periodo destaca la contribución de Alan Turing considerado uno de los padres de la ciencia de la computación [13].

En la década de 1960, se sigue utilizando principalmente en el ámbito de la exploración espacial y la carrera espacial entre Estados Unidos y la Unión Soviética. Por ejemplo, en 1961, la telemetría fue utilizada en la misión Mercury-Redstone 3 de la NASA [14], que llevó al astronauta estadounidense Alan Shepard al espacio. La telemetría permitió a los científicos y los ingenieros monitorear la nave espacial y los sistemas de Shepard en tiempo real, lo que fue esencial para la seguridad y el éxito de la misión. No obstante, comienzan a surgir nuevas utilizades de este tipo de sistemas para su uso más cotidiano y cercano al ciudadano de a pie. En 1960 se empiezan a implementar los primeros sistemas de telemetría en hospitales [15] para conseguir una monitorización de los pacientes con afecciones cardíacas y así prevenir posibles fallos.

Tardaría diez años más de desarrollo y uso en diferentes campos para ver las primeras aplicaciones de la telemetría en el mundo de la competición automovilística. La telemetría

en coches de competición es el proceso de recopilar datos del rendimiento del vehículo en tiempo real y transmitirlos a un equipo de análisis en la pista o en una ubicación remota. Esta información se utiliza para tomar decisiones en tiempo real sobre el rendimiento del vehículo y para ajustar la configuración del mismo.

La telemetría en coches de competición se ha utilizado desde la década de 1970, aunque los primeros sistemas eran muy simples en comparación con los sistemas modernos. En los años 70, los equipos de Fórmula 1 utilizaban dispositivos analógicos que medían la velocidad, la temperatura del motor y otros datos básicos del vehículo [16].

A medida que la tecnología avanzaba, la telemetría se hizo más sofisticada. A fines de los años 80 y principios de los 90, los equipos de Fórmula 1 comenzaron a utilizar sistemas digitales más avanzados para recopilar y transmitir datos [17]. Estos sistemas permitían a los ingenieros del equipo ver en tiempo real la información del vehículo en una pantalla de ordenador.

El inicio de la telemetría en la fórmula 1 surge en 1980, previo al protocolo CAN, donde se hizo un primer intento de introducir este tipo de tecnología. El diseño consistía en un miniordenador cuya función era leer un conjunto de sensores de captación de datos y controlar la suspensión electrónica del Tyrrell 010[18]. Dicho proyecto se abandonó debido a su complejidad de implementar.

Ya en 1990, y gracias al protocolo CAN bus, se consiguió afinar el sistema y llegó a la competición de la mano del equipo Williams y McLaren[19]. Desde este momento, la electrónica pasa a tener un papel principal en el desempeño de los monoplazas en las competiciones siendo notable la diferencia entre los equipos que implementan este tipo de tecnología con aquellos que se resignaban o no podían utilizarla.

En este inicio, los datos se recogían y se almacenaban en el propio monoplaza y una vez que el coche regresaba a boxes, los ingenieros analizaban los valores y adaptaban el coche a las necesidades observadas. Eran pocos los cambios que se realizaban de manera autónoma por parte del propio vehículo.

Tiempo más tarde, ya no era necesaria la vuelta del coche a boxes para la recepción de los datos ya que se comenzaban a implementar los primeros sistemas inalámbricos, aunque no podían modificar o intervenir el coche a distancia[20].

En la temporada 1993-1994, se prohibiría la ayuda electrónica en la conducción alegando que los pilotos debían conducir el monoplaza “por sí mismos y sin ayudas”[21]. El siguiente gran cambio que recibió la telemetría ya fue entrado el siglo XXI (2001) de la mano TAG Electronics[22]. La compañía consiguió introducir en la máxima competición del automovilismo, la telemetría bidireccional, la cual además de leer datos en tiempo real, permitía a los ingenieros modificar la configuración del monoplaza y tan solo un año mas tarde, en 2002, los ingenieros ya eran capaces de modificar por si mismos mapas de motor.

Una vez más la FIA volvió a prohibir las ayudas a la conducción electrónicas[22] pero los equipos cada vez más conseguían disimular o camuflar estas ayudas dentro del complejo sistema de sus cajas de electrónica. Es en este momento donde la FIA decide implantar una electrónica común para todos los coches y en 2008 se introdujo una ECU común para todos los vehículos destacando la eliminación del control de tracción y otro tipo de ayudas electrónicas. La empresa encargada de realizar esta ECU (Electronic Control Unit) fue McLaren Electronics[23].

En la década de 1990, la telemetría se convirtió en una herramienta cada vez más importante para los equipos de competición. Los sistemas digitales se volvieron más avanzados, lo que permitió a los ingenieros del equipo medir una amplia gama de parámetros del vehículo en tiempo real, incluyendo la velocidad del vehículo, la aceleración, la temperatura del motor y la presión de los neumáticos [24].

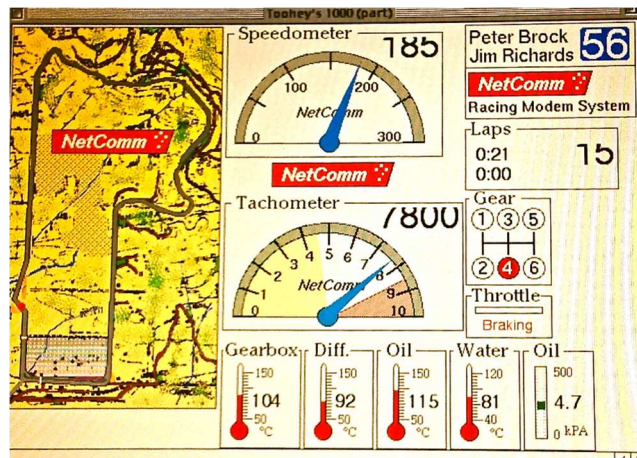


Figura 7: Primer display de valores de telemetría

En la década de 2000 se convierte en una herramienta aún más importante para los equipos de competición. Los sistemas se volvieron más sofisticados, lo que permitió a los ingenieros del equipo monitorear aún más parámetros del vehículo, incluyendo el desgaste de los neumáticos, la eficiencia del combustible y el uso del freno. Una prueba de la importancia que cobra es su popularización en prensa como se puede ver en la figura 8.

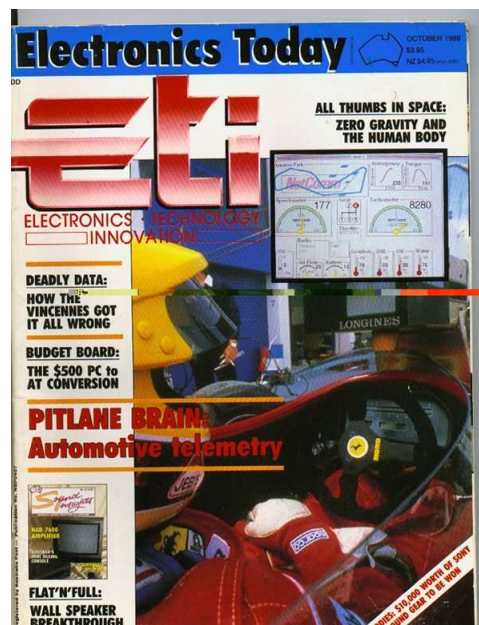


Figura 8: Primer titular donde se menciona el uso de la telemetría en los pitlanes

En los coches de competición de los años 2000, los sistemas de telemetría utilizaban una amplia gama de sensores en el coche, que registraban todo, desde la velocidad del motor y la posición del acelerador hasta la presión de los neumáticos y la temperatura del combustible. Los datos se recopilaban y se transmitían a los ingenieros a través de una variedad de canales de comunicación, incluyendo antenas satelitales, sistemas de radio y conexiones de fibra óptica.

Los ingenieros de los equipos utilizaban estos datos para hacer ajustes en tiempo real en el coche durante las carreras, por ejemplo, cambiando el reparto de frenado para mejorar el agarre en curvas específicas o ajustando la mezcla de combustible para optimizarlo. Además, los datos de telemetría también se utilizaban para ayudar a los equipos a planificar estrategias de carrera, como el momento adecuado para realizar una parada en boxes.

Hoy en día, la telemetría en coches de competición se ha convertido en una herramienta vital para los equipos de carreras en todo el mundo. Los sistemas modernos son altamente sofisticados y permiten a los ingenieros del equipo monitorear y ajustar una amplia gama de parámetros del vehículo en tiempo real, lo que ayuda a los equipos a mejorar el rendimiento y la fiabilidad del vehículo.

Una vez entendido el contexto en el que surge el sistema de telemetría que se plantea, debemos saber la historia sobre los protocolos que utilizan, su funcionamiento y como se implementaran en este trabajo y ver el estado actual de los sistemas de comunicación tanto dentro del monoplaza como a distancia del mismo.

La comunicación en el interior del monoplaza ha experimentado grandes cambios desde sus orígenes hasta nuestros días. En un primer momento los datos se transmitían mediante un conjunto de cable individuales conectados a sensores y actuadores en el vehículo. Estos cables estaban conectados a un panel de instrumentos que contenía indicadores y medidores que mostraban información relevante, como la velocidad y la temperatura del motor.

Este sistema planteaba gran complejidad a la hora de realizar el cableado del coche ya que para cada sensor se necesitaban un mínimo de dos cables si se querían utilizar sus datos en

un cuadro. Si en lugar de un cuadro, se quería utilizar el valor en más de un sistema, se debía cablear el recorrido desde el sensor hasta el sistema de nuevo, incrementando el número de cables de manera exponencial.

Al empezar a complicarse el diseño y aumentar la complejidad del cableado, se empezó a utilizar un ordenador central, capaz de agrupar mensajes y simplificar notablemente el hardware. Todos los sensores del coche se comunicaban con este ordenador y este se encargaba de distribuir los paquetes a los diferentes sistemas del vehículo[25]. Sus principales limitaciones eran que, si un sensor fallaba, tardaba mucho en mandar el dato o el ordenador sufría daños, la telemetría quedaba inútil.



Figura 9: Dashboard F1 1960

Los coches de F1 modernos (y la mayoría de los demás coches de carreras) utilizan el estándar de comunicación CAN Bus para todas las comunicaciones entre componentes. CAN son las siglas de Controller Area Network (red de área de controlador). El estándar fue desarrollado por Bosch GmbH en 1983 específicamente para su uso en automoción[26]. Desde entonces, se ha actualizado varias veces y ahora admite hasta 1 megabit/segundo de transmisión de datos.

El protocolo CAN Bus se utiliza porque, a diferencia de muchos otros protocolos de comunicación, no requiere un ordenador central [27] para gestionar el flujo de datos. Por ejemplo, una LAN (red de área local), que se utiliza para cosas como WiFi y datos móviles celulares, requiere un router para coordinar el movimiento de los paquetes de datos.

Esto es deseable en un coche de F1 por la fiabilidad añadida de la red global. Si un dispositivo de control falla, nada del tráfico CAN se ve afectado y, por tanto, todos los demás componentes siguen funcionando.

El proceso de recopilación de datos en los sistemas de telemetría se lleva a cabo utilizando una serie de sensores[28] y dispositivos de medición que se instalan en el vehículo de competición.

Una vez que los datos se han transmitido a través del Bus CAN, se almacenan en la memoria del sistema de telemetría del vehículo o se transmiten a un ordenador de telemetría en el garaje o en una ubicación remota. El ordenador de telemetría utiliza un software especializado para analizar y visualizar los datos recopilados, lo que permite a los ingenieros del equipo evaluar el rendimiento del vehículo en tiempo real.

El uso del Bus CAN en los sistemas de telemetría ofrece varias ventajas. En primer lugar, el Bus CAN es capaz de transmitir grandes cantidades de datos a una velocidad extremadamente alta, lo que permite una recopilación de datos en tiempo real. Además, el Bus CAN es muy confiable y tolerante a fallos, lo que es crítico en un entorno de carreras de alta velocidad y alta presión.

En la transmisión y externalización de los datos desde el monoplaza hasta el exterior encontramos diferentes protocolos [29] [30] y mecanismos a lo largo de la historia y diferentes soluciones al problema.

La transmisión de datos por radio en los automóviles de competición ha evolucionado significativamente desde sus inicios en la década de 1960. En sus primeros días, los equipos de carreras utilizaban radios de banda ciudadana para transmitir información desde el automóvil a los boxes. Estas radios eran limitadas en términos de alcance y calidad de la señal, y eran propensas a interferencias y distorsiones.

En la década de 1980, los equipos de carreras comenzaron a utilizar radios de frecuencia modulada (FM) para transmitir datos en tiempo real desde el automóvil a los boxes. Estas radios eran mucho más confiables y ofrecían una calidad de señal más clara, lo que permitía

una mejor transmisión de datos. Los ingenieros podían recibir información sobre el rendimiento del motor, la velocidad, la temperatura y otros parámetros importantes directamente desde el automóvil en tiempo real [20].

En la década de 1990, la tecnología de radio digital comenzó a utilizarse en las carreras de automóviles. Los sistemas de radio digital ofrecían una calidad de señal aún mejor y permitían una mayor cantidad de datos para ser transmitidos. Además, la tecnología de radio digital también reducía significativamente la interferencia y las interrupciones en la señal de radio, lo que mejoraba la precisión de los datos transmitidos [20].

Cabe destacar que todas estas soluciones han sido aplicadas en el mundo de las competiciones más importantes como la fórmula 1 o NASCAR, pero nunca en Formula Student, donde pretende centrarse este proyecto.

La solución que plantea este trabajo es el uso de una radio LoRa. Las radios LoRa (Long Range) son una tecnología de comunicación inalámbrica [31] que se utiliza cada vez más en entornos de carreras de competición debido a su alta eficiencia energética y su capacidad para transmitir datos a largas distancias.

Una de las ventajas principales de las radios LoRa [32] es que ofrecen un alcance mucho mayor que las radios convencionales de corto alcance, lo que es muy importante en las carreras de competición, ya que las señales deben transmitirse a lo largo de una pista larga y en entornos complejos donde hay obstáculos y cambios de terreno.

Finalmente, y previo a ver las soluciones actuales, debemos ver el último elemento principal del sistema de telemetría: la visualización de los datos en tiempo real.

La visualización de datos en las competiciones automovilísticas ha evolucionado significativamente a lo largo de los años, y ha sido una herramienta importante para los equipos de carreras en la toma de decisiones sobre ajustes en el automóvil y estrategias en la carrera.

En las primeras décadas del automovilismo, los datos se recopilaban y registraban manualmente. Los ingenieros de los equipos de carreras llevaban un registro de los tiempos de vuelta, la velocidad y otros parámetros importantes en una libreta y realizaban cálculos y análisis posteriores en busca de patrones y tendencias.

En la década de 1970, los equipos de carreras comenzaron a utilizar los primeros sistemas de telemetría para recopilar y transmitir datos en tiempo real desde el automóvil a los boxes. Los datos se mostraban en un display en los boxes y los ingenieros podían analizarlos en tiempo real para hacer ajustes en el automóvil durante la carrera[33].

A medida que la tecnología de la telemetría se desarrollaba, también lo hacía la visualización de datos. En la década de 1980, se comenzaron a utilizar monitores de video para mostrar los datos en tiempo real. Los datos se podían mostrar en forma de gráficos y tablas, lo que permitía a los ingenieros analizar los datos con mayor facilidad y tomar decisiones más informadas.

En la década de 1990, los sistemas de visualización de datos se volvieron aún más avanzados. Los monitores de video se hicieron más grandes y los datos se podían mostrar en tiempo real en varias pantallas. Además, los ingenieros podían personalizar las pantallas para mostrar los datos específicos que eran más importantes para ellos, lo que les permitía analizar la información de manera más efectiva [34].

En la actualidad, los sistemas de visualización de datos son altamente avanzados. Los equipos de carreras pueden utilizar pantallas táctiles para mostrar y analizar datos en tiempo real. Los datos se pueden mostrar en gráficos en 3D y los ingenieros pueden realizar simulaciones en tiempo real para evaluar diferentes estrategias de carrera.

Todos estos sistemas de visualización utilizan softwares o librerías, principalmente en JavaScript, desarrollada por empresas de terceros. Algunas de estas empresas se dedican exclusivamente a este cometido y cuentan con productos de visualización de datos extremadamente completos y optimizados. El principal problema de estos softwares y códigos es su elevado coste.

Algunas soluciones que se encuentran disponibles en el mercado son implementadas por empresas como Tesla o Microsoft entre otras. Las cuales hacen uso de productos de compañías como SciChart [35] o LightningChart. [36] Estas empresas tienen como único objetivo, el desarrollo de graficas actualizadas en tiempo real y que hacen fácil e intuitivo la visualización de los datos. El principal problema y la gran diferencia de estos sistemas y el que se propone en este proyecto es el coste de las licencias para su uso, ya que el precio de la licencia más básica y muy limitada en SciChart asciende a 1200\$ al año y en lightningchart, herramienta utilizada en el sector de las competiciones automovilísticas como NASCAR o Formula 1, las licencias más básicas ascienden hasta más de 1500\$.

Por lo tanto, el sistema que se plantea, pretender aportar una solución o sistema completo de telemetría al alcance de todo tipo de presupuestos, haciendo uso de herramientas de protocolos y visualización de Código abierto y de uso libre y gratuito como Chatr.js [37].

Una vez entendido el estado de las diferentes tecnologías que se utilizan, debemos ver que sistemas cumplen en la actualidad con las especificaciones esperadas, sus tecnologías y ver las similitudes y diferencias con el proyecto que aquí se describe.

Cabe destacar que en el mundo de la Formula Student, no existen apenas sistemas de telemetría en tiempo real, por lo que no se pueden hacer grandes comparaciones en este sector. Los principales sistemas que encontramos en Formula Student son diferentes productos.

Como sistemas completos de telemetría en competiciones automovilísticas amateur o de presupuestos más ajustados encontramos pocos productos como Pódium de RaceCapture [38], una empresa dedicada a la telemetría de bajo presupuesto.

El producto que se ofrece en este caso es una herramienta de hardware, similar a una caja negra, encargada de leer los datos del Bus CAN, mostrarlos al piloto y transmitirlo por radio.



Figura 10: Podium Connect MK2 - RaceCapture

La principal diferencia de este producto comercial y el proyecto que aquí se describe es en primer lugar la diferencia económica y en segundo lugar la adaptabilidad y capacidad de modificación.

El pódium connect MK2 [39] tiene un coste inicial de 649,99\$, lo cual supone un coste muy elevado para equipos medianos y pequeños de la competición de formula Student. La propuesta aquí descrita supone un solamente un 10% del coste de este producto incluyendo antenas y cableado (algo que habría que añadir al coste previo).

En segundo lugar, destacar la falta de adaptabilidad por parte del MK2. Una de las diferencias más notables la baja adaptabilidad a cambios ya que dependemos del fabricante para introducirle mejoras y modificaciones. Además, su uso se restringe a un uso local, dentro del vehículo, sin transmisión ni recepción en tiempo real.

Otros dispositivos que también cumplen algunas especificaciones son por ejemplo Pi Toolbox [40], la cual es una herramienta de análisis de datos de telemetría que ofrece una amplia variedad de funciones de análisis de datos, como gráficos y tablas de datos. También ofrece la capacidad de analizar datos históricos y comparar diferentes carreras y ajustes.

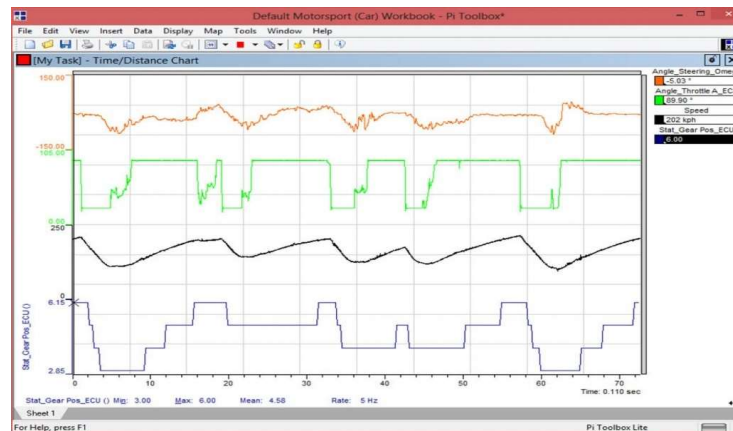


Figura 11: Visualización de los datos desde Pi Toolbox

Existen otras opciones en el mercado igual de completas que las previas descritas pero cuyo coste es muy elevado por lo que no se contemplan en este estado de la cuestión ya que superan con creces el presupuesto de cualquier tipo de equipo mediano, llegando a costar algunas TCU (Telemetry Control Unit) más de 5000\$ como en el caso de MoTec [41], compañía de Bosch especializada en telemetría deportiva.



Figura 12: Bosch Motorsports Telemetry - MoTec

Otro producto que encontramos en el mercado y muy popular dentro del mundo de la Formula Student para solucionar la telemetría sin un departamento propio o que se dedique a ello es el DASH2 [42].

Es un sistema de visualización y adquisición de datos en tiempo real diseñado específicamente para monitorear y analizar el rendimiento de un vehículo de carreras. Se compone de un conjunto de pantallas digitales y sensores que se instalan en el automóvil de competición. Estas pantallas muestran información crucial sobre el funcionamiento del

vehículo, como la velocidad, las revoluciones por minuto (RPM) del motor, la temperatura del motor, los niveles de combustible y otros parámetros relevantes.

Además de mostrar datos en tiempo real, el Dash2 también tiene la capacidad de registrar y almacenar información para su posterior análisis. Los datos recopilados se pueden utilizar para evaluar el rendimiento del vehículo en diferentes condiciones de conducción, optimizar estrategias de carrera y realizar ajustes en la configuración del automóvil para obtener un mejor desempeño.



Figura 13: Visualización del display de Dash2

También cabe destacar que actualmente no hay un sistema completo de telemetría en el mundo de la formula Student que agrupe todas las funcionalidades que pretende recoger el aquí descrito: recepción del bus CAN, transmisión de los datos de manera inalámbrica en tiempo real mediante el uso de antenas LoRa, su recepción, almacenamiento y visualización en tiempo real.

Finalmente, en la Figura 14, se adjunta la comparativa entre las diferentes tecnologías descritas en la sección de descripción donde se estudian las mismas dependiendo de las siete principales características que debe presentar o que son relevantes para los sistemas de telemetría de un formula Student. Las características son las siguientes:

- Disponibilidad: cualidad que mide la viabilidad del sistema y su existencia de sistemas similares en la industria. Siendo el 5 la existencia de sistemas similares o de tecnologías, códigos ampliamente utilizados por la comunidad en el mismo campo y en campos diferentes y 1 la existencia de ese sistema únicamente en el campo de la telemetría de competición y con uso muy restringido.
- Adaptabilidad: capacidad del sistema a acoplarse a las entradas o al sistema electrónico previamente diseñado sin demasiadas modificaciones. El mayor punto de adaptabilidad es la conexión con el resto del vehículo siendo un esclavo más en la comunicación del bus can y escuchando únicamente los mensajes que se transmiten (5). Un punto intermedio es aquel cuya conectividad con el resto del vehículo se lleva a cabo mediante el bus can pero requiere de mandar mensajes al resto de participantes para que adapten sus comunicaciones con el mismo (3). Finalmente, una adaptabilidad baja son todos aquellos sistemas que precisan de procesados previos al sistema o incluso la existencia de sistemas de filtrado o adaptadores de señal BUS Can a la entrada del sistema (1).
- Coste: varias de las tecnologías planteadas previamente tienen costes elevados y no asumibles por parte de muchos de los equipos de la competición. Esta cualidad mide de más asumible o barata (5) a más costosos y cuyas licencias son inasumibles por los equipos de presupuesto medio o bajo (1). Alcance (Transmisión): cualidad que mide el alcance o transmisión de la señal desde el monoplaaza a la antena receptora de los datos.
- Modular: capacidad de adaptar nuevos elementos al sistema si así en un futuro lo requiere
- Tiempo de respuesta: tiempo desde la generación del dato hasta su actualización en el visionado en tiempo real y su almacenamiento en la base de datos.
- Complejidad de lenguaje: complejidad del lenguaje de programación utilizado para el desarrollo del sistema. También debe tenerse en cuenta la no

unicidad o unicidad del lenguaje siendo preferible y recomendado el uso de un mismo lenguaje o el menor número de ellos.

- Disponibilidad de hardware o de soporte de software: la disponibilidad mide la reacción frente a fallos y sus distintos métodos de solución. Por ejemplo, ante un fallo de software, debes contactar con la empresa encargada de la producción de ese software o con su página de duda, soporte o ayuda al usuario. En el campo del hardware, se mide la simpleza o complejidad de la sustitución o arreglo por parte del particular o del equipo dueño del sistema de telemetría de un fallo electrónico.

Como conclusión a este capítulo, podemos ver como sí que existen soluciones similares en la industria, pero todas ellas son altamente costosas, siendo el precio medio de las mismas superior a 700 euros y con una adaptabilidad y capacidad de modificación bajas. Todos estos productos en la industria son dependientes de actualizaciones por parte del fabricante y con posibles errores difícilmente solucionables por parte del equipo encargado de desarrollar el sistema de telemetría.

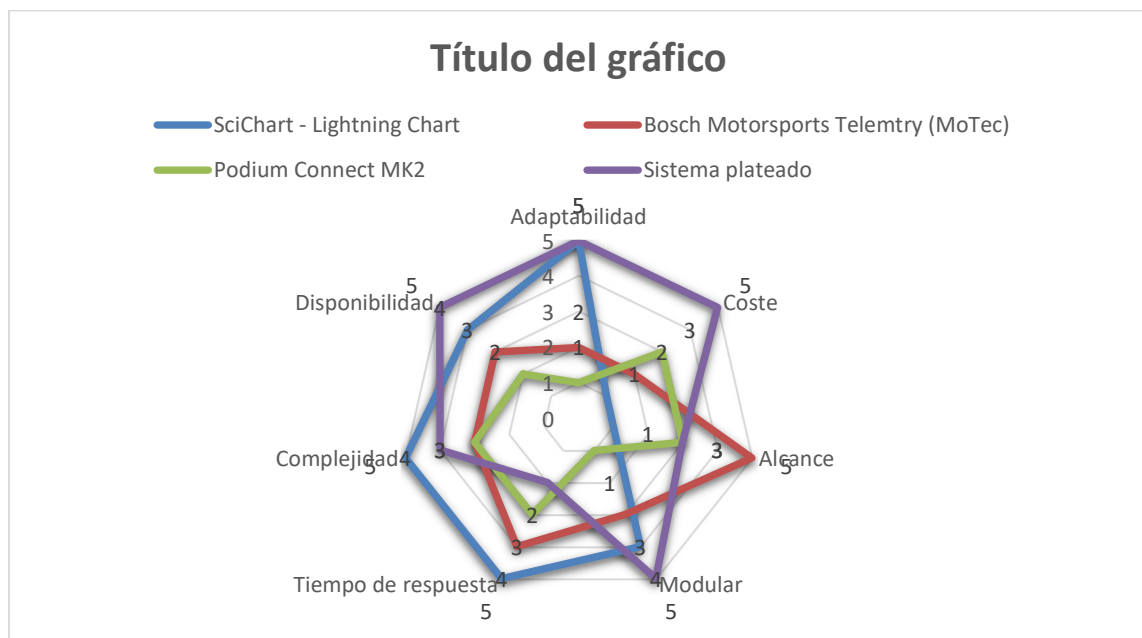


Figura 14: Comparativa entre las diferentes tecnologías

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

La Formula Student es una competición de diseño y construcción de coches de carreras para estudiantes universitarios de todo el mundo. Los equipos tienen la tarea de diseñar y construir un automóvil de carreras, luego competir en una serie de pruebas de aceleración, frenado y maniobrabilidad. La competición se originó en 1981 en el Reino Unido como la "Formula Student" y ha crecido rápidamente desde entonces, con eventos que se llevan a cabo en todo el mundo.

En España, la Formula Student se inició en 2006 con la creación del equipo ETSEIB Motorsport, formado por estudiantes de la Escuela Técnica Superior de Ingeniería Industrial de Barcelona. Desde entonces, han surgido numerosos equipos en diferentes universidades y centros de formación de todo el país, como la Universidad Politécnica de Madrid, la Universidad de Granada, la Universidad de Valencia, la Universidad Carlos III de Madrid, la Universidad de Vigo y la Universidad Pontificia Comillas entre otras.

Uno de los principales problemas de la competición es el poco tiempo que pueden dedicar los estudiantes de ingeniería a proyectos ajenos a sus grados y el poco tiempo que pueden colaborar en el proyecto ya que su participación en la formula Student se reduce a los años en los que estén matriculados en la universidad (entre 4 y 6 años). Debido a esto, el progreso es más lento de lo esperado ya que la rotación de la plantilla es constante y se tiene que documentar de manera extensa todos los conocimientos que se van aplicando para no perderlos en los constantes relevos generacionales.

Es por ello que los principales equipos suelen tener problemas los primeros años a la hora de construir el monoplaza, ya que no tienen pruebas, errores o versiones anteriores sobre los cuales empezar a desarrollar. Una vez desarrollado el primer prototipo se le van

introduciendo mejoras para obtener el máximo rendimiento de la inversión realizada por las empresas y patrocinadores.

Una vez entendido el punto de partida de los equipos, y su lento desarrollo, debemos comprender la situación actual de los departamentos de telemetría dentro de la competición. En este caso, son pocos los equipos que han podido empezar a desarrollar herramientas propias que desempeñen esta función por lo que es raro encontrar sistemas muy refinados.

Los equipos con presupuestos millonarios, superiores al millón de euros, son capaces de poder costearse las soluciones previamente mencionadas en la sección de estado del arte, no obstante, es común encontrar que los equipos medianos y pequeños no analicen su telemetría ya que carecen de sistemas para obtenerla.

En este proyecto, construimos una herramienta de código libre donde con hardware muy de muy bajo coste y con gran adaptabilidad para proveer a equipos pequeños y medianos de herramientas de análisis de sus datos sin que suponga un gran desembolso para el equipo.

El principal gasto de los equipos suele incurrir tras fallos que pueden ser prevenidos con una telemetría clara y que prevenga errores. Los fallos más costosos y críticos suelen ir relacionados con el acumulador en los coches eléctricos, ya que, debido a su composición y condiciones, es muy inflamable y puede arruinar el resto de los elementos del monoplaza. Son varios los accidentes que han ocurrido relacionado con las baterías y que pueden ser notablemente menores con un seguimiento en tiempo real de su temperatura y voltaje.

Un ejemplo de ello es el accidente de la Universidad de Oviedo en 2021[43] donde su monoplaza de carreras se incendió arruinando así todos los equipos electrónicos, mecánicos e incluso el lugar de trabajo.

Como consecuencia, se pretende igualar las condiciones entre equipos y la optimización de los recursos mediante una herramienta altamente modulable y adaptable a las necesidades de los equipos y monoplazas.

Se pretende proveer, el almacenado de los datos, la transmisión en tiempo real, su visualización tanto postcarrera como en tiempo real y su análisis con el objetivo de prevenir accidentes y errores, alargando y optimizando la inversión, así como la optimización de los recursos.

4.2 OBJETIVOS

Los principales objetivos de este proyecto son los previamente mencionado en el anexo B actualizados al estado actual y definitivo del mismo.

Los objetivos son:

- La elaboración e implantación de un sistema completo de telemetría en un monoplaza sin plataforma actual.
- La mejora y la optimización de los sistemas actuales de almacenamiento y visualización de los datos tanto en tiempo real como a la finalización de la sesión.
- Generar una base de datos propia controlada por un servidor propio y alojado en una propiedad del equipo, sin depender de costosas licencias de terceros en cloud como, por ejemplo, Google Cloud, AWS o ElasticSearch.

Para el correcto desarrollo de los principales objetivos, además se deberán implementar una serie de mejoras con las cuales se pretende:

- Solucionar problemas fluidez en la visualización de los datos en tiempo real mediante la migración de la visualización a un entorno web.
- Generar un entorno de pruebas completo y funcional capaz de simular las condiciones reales del monoplaza sin tener la necesidad de estar en funcionamiento.

En caso de cumplir todos los objetivos previos, se establecerán nuevos objetivos como;

- La generación de un sistema de prioridades en los mensajes a transmitir por radio, siendo los más relevantes aquellos valores que tengan mayor frecuencia (revoluciones por minuto, voltaje pico, etc.)
- Compresión de los paquetes de radio, obteniendo así mayor ancho de banda y por lo tanto mayor capacidad de transmisión.
- Sistema de detección de errores de transmisión.

4.3 METODOLOGÍA

Para la realización del proyecto se han utilizado Python, JavaScript, Java, SQL y C++ para el correcto funcionamiento del mismo. Python para el almacenamiento del ordenador de a bordo, JavaScript para la visualización y gestión de peticiones a la base de datos, Java para la programación de la aplicación que controla la base de datos, SQL para la descripción y uso de las tablas y C++ para crear el entorno simulado.

La metodología empleada para el desarrollo como para la redacción del documento se ha seguido un modelo de cascada con revisiones continuas. Se comenzó con una fase de especificación de los requisitos mediante sesiones de feedback con pilotos e ingenieros y posteriormente una de diseño con aceptación y validación de los mismos. Una vez terminada esta etapa se procedió a la implementación. Cabe destacar que, llegados a ese punto, se realizaban implementaciones y modificaciones continuas.

4.4 ORGANIZACIÓN Y ESTIMACIÓN ECONÓMICA

La planificación del proyecto y su desarrollo ha seguido la cronología estimada en el anexo B. No obstante, se han hecho modificaciones debido a los problemas en su implementación en el monoplaza real. En la figura 15, se puede ver dicha planificación inicial.

Semana	Semanas de duración																														
	Sep			Oct			Nov				Dic				Ene			Feb			Mar			Abr			May				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Redacción propuesta UIUC	█	█	█																												
Búsqueda de materiales y hardware necesario	█	█	█																												
Programación del entorno de pruebas																															
Testeo de sensores y Arduino																															
Cominezo del desarrollo en el entorno de pruebas																															
Propuesta UIUC definitiva de la visualización																															
Propuesta definitiva																															
Programación de la visualización definitiva																															
Aplicación de la visualización fuera del entorno de prueba																															
Testeo del correcto funcionamiento y solución de errores																															
Redacción del TFG	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█

Figura 15: Planificación semanal del proyecto y redacción del TFG

Los principales cambios que se han realizado han sido la búsqueda de materiales y hardware y el correcto testeo del funcionamiento del sistema ya aplicado a la vida real.

En cuanto a la estimación económica, encontramos diversos gastos en el entorno hardware mientras que en el software no existe ningún gasto ya que es todo código libre.

En el campo del hardware, encontramos como gastos:

- Arduino Uno para la recepción y transmisión de los datos. Se necesitarán dos Arduino uno on-board capaz de registrar y transmitir los datos y otro en el box encargado de la recepción de estos. Este hardware puede encontrarse por 26€ en Amazon o en la web oficial.
- MP2515 CAN BUS. El MCP2515 es un controlador de bus CAN que implementa la especificación CAN 2.0B. Es capaz de transmitir y recibir tanto datos estándar (11 bits) como extendidos (29 bits) y tramas remotas. Este módulo en particular se basa en el IC de controlador CAN MCP2515 y el IC tranceptor CAN TJA1050. El IC MCP2515 es un controlador CAN independiente y tiene una interfaz SPI integrada para la comunicación con microcontroladores. Estará conectado al Arduino on-board para la lectura de los datos. Se puede encontrar en el mercado por 2€.
- LoRa shield para Arduino y antenas. Estas antenas serán las utilizadas la transmisión de los datos desde dentro del coche hasta el receptor del box o pitlane. Coste aproximado de 40€.

Además, debemos tener en cuenta la amortización de los costes del proyecto. Su duración ha sido de 8 meses. El valor del ordenador en el que se ha realizado el proyecto tiene un

coste de 1219,99€ que han sido amortizados en 4 años por lo que representa un coste de 203,33€ en material. Y finalmente las horas de un ingeniero en España aumentan a 12,38€ y habiéndose dedicado unas 380 horas aproximadamente, el precio por mano de obra asciende a 4704,40€.

Por lo tanto, el coste del estudio se puede estimar en 5003,73€.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

5.1 ANÁLISIS DEL SISTEMA

Todos los monoplazas y sobre todo los de competición necesitan de una sensorización intensiva para poder comunicar las decisiones del piloto a los elementos y sistemas que controlan al coche. Una parte fundamental de este proceso es la transmisión en el menor tiempo posible de las señales medidas por los sensores a los sistemas finales o intermedios, para que el piloto cuente con una sensibilidad casi inmediata y completa de todas sus decisiones y movimientos y que realmente solo influya la destreza en su conducción.

5.1.1 ANÁLISIS DEL PROTOCOLO INTERNO

Para obtener la sensación de inmediatez se necesita un protocolo especializado en señales y sonorización de automóviles como es el protocolo extensamente explicado previamente: BUS CAN que, a diferencia del modelo previo de conexión punto a punto o de un ordenador central, el BUS CAN propone un funcionamiento descentralizado. El BUS CAN utiliza un sistema de comunicación multiplexado, lo que significa que múltiples dispositivos pueden compartir la misma línea de comunicación. Esto se logra mediante el uso de un mensaje de identificación (ID) único que se utiliza para distinguir los diferentes mensajes enviados a través del BUS CAN.

Este diseño ha sido realizado utilizando las especificaciones y distribuciones más comunes dentro del mundo de la Formula Student. La gran mayoría de los monoplazas implementan el protocolo CAN para transmitir sus mensajes internos entre sensores y resto de componentes como acumulador, placas base o inversor entre tantos.

El BUS CAN consta de dos líneas de comunicación: CAN High (CANH) y CAN Low (CANL). La transmisión de datos se realiza mediante la modulación de la amplitud de la señal eléctrica entre estas dos líneas. Una señal de voltaje de nivel alto en CANH y un voltaje

de nivel bajo en CANL indica un "1" lógico, mientras que una señal de voltaje de nivel bajo en CANH y un voltaje de nivel alto en CANL indica un "0" lógico.

Los datos se transmiten en paquetes de datos llamados tramas CAN. Cada trama contiene información sobre el emisor, el receptor y la prioridad del mensaje. Los mensajes de mayor prioridad tienen prioridad sobre los mensajes de menor prioridad en el BUS CAN.

El BUS CAN utiliza un sistema de detección de errores para garantizar la integridad de los datos transmitidos. Cada trama CAN incluye un código de comprobación de redundancia cíclica (CRC) [44] que se utiliza para detectar errores en la transmisión de datos. Si se detecta un error, el dispositivo emisor reenvía el mensaje hasta que se recibe una confirmación de que se ha recibido correctamente.

Es altamente eficiente y confiable en la transmisión de datos en sistemas de alta velocidad e interferencia electromagnética, como los sistemas de telemetría de los monoplazas de carreras. El protocolo CAN es ampliamente utilizado en la industria del automóvil y se ha convertido en un estándar de facto en la transmisión de datos en sistemas electrónicos de todo tipo.

El sistema propuesto necesita de una lectura de todos los datos relevantes del monoplaza y de su transmisión al resto de ECUs que precisen de sus valores. Una de estas Electronic Control Units será la ECU de Telemetría la cual almacenará en un data logger, o base de dato interna, todos los valores del canal de comunicaciones replicando así una caja negra en un avión. Así, en caso de pérdida de la señal o de fallo en la transmisión se seguirán almacenando los datos para poder ser estudiados con detenimiento por los expertos cuando sea accesible físicamente una vez finalizadas las pruebas. Un esquema del protocolo interno es el reflejado en la figura 16.

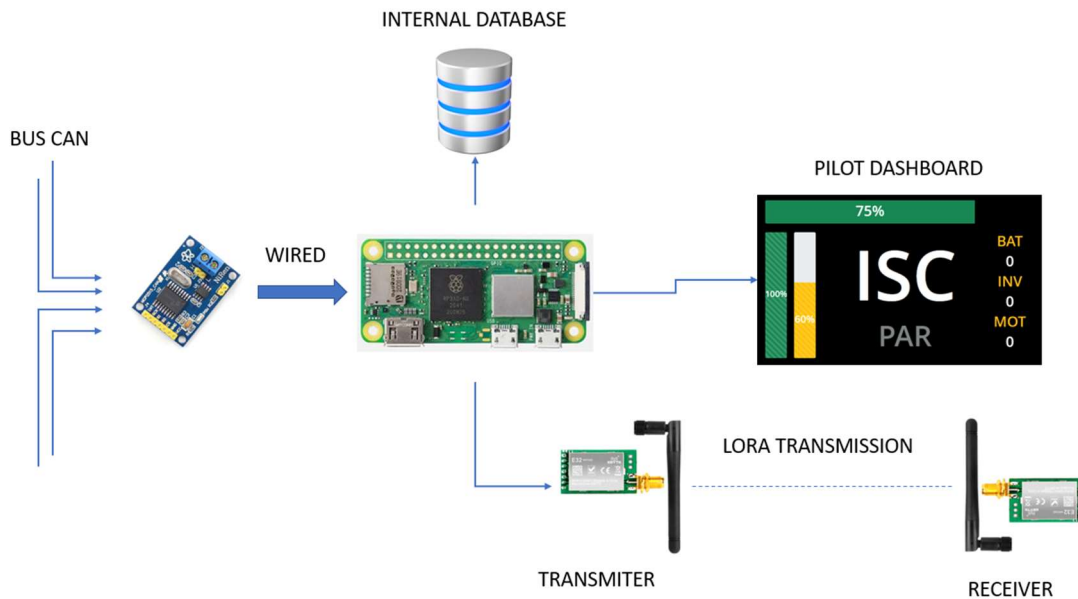


Figura 16: Modelo de hardware interno del vehículo y conexión principal

5.1.2 ANÁLISIS DEL DASHBOARD ON-BOARD

Además de esta labor, la Telemetry ECU o TECU debe cumplir más funciones como la de cargar el dashboard para el piloto, llamada dashboard on-board, la cual mostrará los valores más relevantes al piloto como:

- Presión del pedal de acelerador
- Presión del pedal de freno
- Estado de la batería: porcentaje restante y demanda actual
- Velocidad actual
- Par total ejercido por el motor
- Temperaturas críticas como la temperatura más alta del pack de baterías, temperatura del inversor y la temperatura del motor.

Previo al resto de elementos que formarán el sistema, debemos analizar cada uno de los puntos que se muestran al piloto y su razonamiento, ya que el espacio y la visión del mismo es reducida y cada espacio ocupado en la pantalla y ángulo de visión del piloto es muy preciado.

En primer lugar, los valores sensores de presión de los pedales son fundamentales ya que el piloto, dentro del cockpit o habitáculo de conducción, carece de sensibilidad en el recorrido de su pie y tiene que estimar el recorrido del pedal. Por lo tanto, y como se ha confirmado tras pruebas, este valor ayuda a estimar la fuerza real que se está ejerciendo sobre cada uno de los pedales.

Cabe destacar que, en el caso de los frenos, su accionamiento es puramente mecánico, lo cual hace que su efectividad sea independiente de un valor digital. En este caso, el piloto necesita saber el valor real de la fuerza que está experimentando el coche para saber si realmente es capaz de tener un frenado mayor al que presenta. En el caso del acelerador esta realidad está sesgada ya que no depende solamente del valor de presión del pie sobre el pedal sino también de las limitaciones que presenta el motor, la batería o el inversor.

En segundo lugar, encontramos el estado de la batería restante, el cual es de gran ayuda en todas las pruebas de la competición, pero se hace indispensable para la prueba de resistencia, la más dura de toda la competición. En esta prueba el monoplaza deberá estar corriendo durante 20 minutos aproximadamente y el piloto deberá hacer un uso eficaz de la batería ya que deberá aguantar toda la prueba además de no bajar de ciertos valores de voltaje ya que podrían ser peligrosos para la salud de las celdas. Además, resulta útil a la hora de estimar el rendimiento del monoplaza durante las pruebas y así poder realizarlas sin tener que cargar la batería tras la finalización de cada una.

La velocidad actual es quizás el dato menos relevante para las pruebas de larga duración, pero de gran relevancia para la prueba de aceleración y la prueba de frenado, ya que podemos estimar o hacer pruebas para saber en qué momento se bloquean las cuatro ruedas y ver si con la velocidad actual del monoplaza se conseguirá superar la prueba.

El par ejercido por el motor resulta relevante ya que podemos ver que cantidad de la fuerza que ejerce el piloto se convierte en fuerza real ejercida en sobre la transmisión y, por lo tanto, en el vehículo.

Finalmente, encontramos las diferentes temperaturas críticas del vehículo, las cuales ayudarán al piloto a gestionar la demanda respecto de la temperatura de cada uno de los elementos. En este caso, encontramos como temperatura más crítica la referente a TBAT o temperatura de las baterías ya que solo se mostrará la temperatura más alta leída en todo el acumulador. Esta temperatura es clave para ver si el desgaste que está teniendo la batería responde a una demanda en condiciones normales o se está demandando voltaje cuando la temperatura es demasiado alta y por lo tanto su vida útil se reduce.

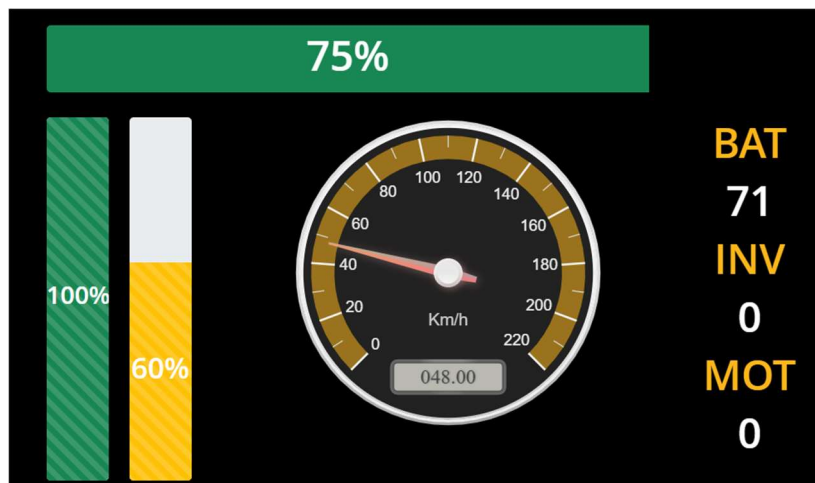


Figura 17: Dashboard on-board

Una vez analizada la visualización *on-board* (la que se muestra en la figura 17) y en tiempo real del monoplaza, se continua con el análisis del sistema de telemetría y resto de funcionalidades que debe satisfacer. Antes de especificar las funcionalidades en el garaje debemos entender el análisis realizado para solucionar el problema de la transmisión de los datos en tiempo real desde el monoplaza en movimiento hasta el receptor.

5.1.3 ANÁLISIS DE LA TRANSMISIÓN INALÁMBRICA DEL VEHÍCULO

Previamente a tomar la decisión sobre cómo solucionar la transmisión de señales, se realizó un estudio o análisis de las principales tecnologías que se podían utilizar con dicho propósito. Hay varias tecnologías que se utilizan para la transmisión de datos de manera inalámbrica en la competición automovilística, y la elección de la tecnología depende de varios factores, como el alcance requerido, la velocidad de transmisión y el costo.

Algunas de las tecnologías más comunes utilizadas en la transmisión de datos inalámbrica en la competición automovilística son:

- Wi-Fi: tecnología de comunicación inalámbrica que se utiliza para la transmisión de datos de alta velocidad a corta distancia. En la competición automovilística, el Wi-Fi puede utilizarse para transmitir datos de telemetría desde el vehículo a una estación base cercana.
- Bluetooth: transmisión de corto alcance que se utiliza para la comunicación entre dispositivos. En la competición automovilística, el Bluetooth puede utilizarse para la comunicación inalámbrica entre el vehículo y otros dispositivos cercanos, como un teléfono móvil o una tableta.
- Radio de banda estrecha: tecnología de comunicación inalámbrica de larga distancia que se utiliza para la transmisión de datos a través de grandes áreas. En la competición automovilística, la radio de banda estrecha puede utilizarse para la transmisión de datos de telemetría desde el vehículo a una estación base remota.
- Red celular: comunicación inalámbrica de larga distancia que se utiliza para la transmisión de datos a través de redes móviles. En la competición automovilística, la red celular puede utilizarse para la transmisión de datos de telemetría desde el vehículo a una estación base remota.

Tras ver las principales tecnologías se hizo un análisis de las más viables y económicas para su uso.

En primer lugar, se planteó el uso de un router o señal Wifi la cual crease una red Local o red LAN entre monoplaza y servidor y llevase a cabo el intercambio de datos haciendo uso de la misma. Las principales ventajas son:

- Velocidad de transferencia de datos: ofrecen altas velocidades de transferencia de datos, lo que permite una transferencia rápida y eficiente de grandes cantidades de datos.
- Conexión inalámbrica: permiten la conexión de dispositivos a la red, lo que elimina la necesidad de cables y permite una mayor movilidad.
- Accesibilidad: permiten la conexión de varios dispositivos a la misma red, lo que permite a los usuarios compartir información y recursos fácilmente.
- Fácil configuración: no requiere mucho conocimiento técnico.
- Flexibilidad: son muy flexibles y se pueden adaptar a diferentes entornos, lo que las hace ideales para su uso en hogares, oficinas, escuelas y otros lugares.

No obstante, y pese a sus grandes ventajas, encontramos una desventaja la cual imposibilita su uso y su aplicación en la vida real. El problema reside en la radiación y frecuencia de esta señal.

El efecto Doppler es un fenómeno que se produce cuando una onda (ya sea sonido, luz o cualquier otra forma de onda) es emitida por una fuente que se está moviendo en relación con el observador. Cuando la fuente se mueve hacia el observador, la frecuencia de la onda se comprime y suena más alta, y cuando la fuente se aleja, la frecuencia se estira y suena más baja. Este fenómeno se utiliza en una variedad de aplicaciones, incluyendo la medición de la velocidad y la distancia de objetos en movimiento, pero puede ser un problema en la transmisión de señales inalámbricas en carreras de monoplazas debido a la alta velocidad del vehículo y la frecuencia de transmisión de las señales.

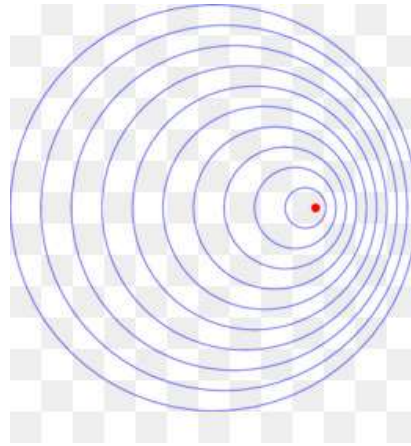


Figura 18: Efecto Doppler

En el contexto de la transmisión de señales WiFi dentro de un monoplaza de carreras, el efecto Doppler puede causar fluctuaciones en la señal a medida que el automóvil se mueve a altas velocidades.

Esto se debe a que la señal WiFi es una onda electromagnética y, por lo tanto, está sujeta al efecto Doppler. A medida que el monoplaza se mueve a alta velocidad, la frecuencia de la señal WiFi recibida por el receptor puede cambiar debido a los cambios en la distancia entre el transmisor y el receptor, lo que puede provocar una degradación en la calidad de la señal e incluso la pérdida de la conexión. Un ejemplo del funcionamiento del efecto Doppler se muestra en la figura 18.

Además, los monoplazas de carreras se mueven a altas velocidades, lo que significa que la señal WiFi tendría que ser transmitida y recibida a velocidades extremadamente altas para mantener una conexión constante y fiable. Esto puede ser difícil de lograr con la tecnología actual y, por lo tanto, se prefieren otros métodos de transmisión de datos en la industria automotriz, como el protocolo CAN o la transmisión de datos por radio de banda estrecha. Siendo esta última opción la adoptada en este proyecto a través de una radio LoRa (Long Range).

La tecnología LoRa (Long Range) es una tecnología de comunicación inalámbrica de largo alcance diseñada para transmitir datos a largas distancias con un bajo consumo de energía.

Esto permite a los equipos de carreras enviar y recibir datos desde el monoplaza incluso a largas distancias, lo que es especialmente útil en circuitos de carreras más grandes. Además, debido a que los radios LoRa pueden operar en una amplia variedad de frecuencias, los equipos pueden seleccionar la frecuencia óptima para minimizar los efectos del ruido y la interferencia electromagnética en la transmisión de datos.

A diferencia de las redes Wi-Fi y Bluetooth, que utilizan una amplia gama de frecuencias de radio para transmitir datos, LoRa utiliza una banda de frecuencia de radio específica que está reservada para su uso en sistemas de comunicación de baja potencia. Esto le da una ventaja significativa en términos de alcance, ya que puede transmitir datos a varios kilómetros de distancia en condiciones óptimas.

Otra ventaja importante de la tecnología LoRa es su capacidad para operar en entornos de alta densidad de señales sin verse afectada por la interferencia electromagnética y el ruido. Esto es especialmente importante en las carreras de monoplazas, donde las señales de radio de los distintos equipos pueden interferir con las señales de otros equipos.

Además, la tecnología LoRa también es altamente escalable y puede utilizarse en redes de diferentes tamaños, desde redes pequeñas de unos pocos nodos hasta grandes redes de miles de nodos.

Finalmente, también permite añadir seguridad a la transmisión ya que es altamente modulable y adaptable a las necesidades de cada equipo.

La transmisión por lo tanto será a través de una radio LoRa, la cual será controlada por un Arduino tanto en la transmisión desde el monoplaza hasta el servidor receptor, donde también estará controlado por un Arduino receptor.

El funcionamiento de dichos Arduino y del LoRa se especifica en la implementación del sistema.

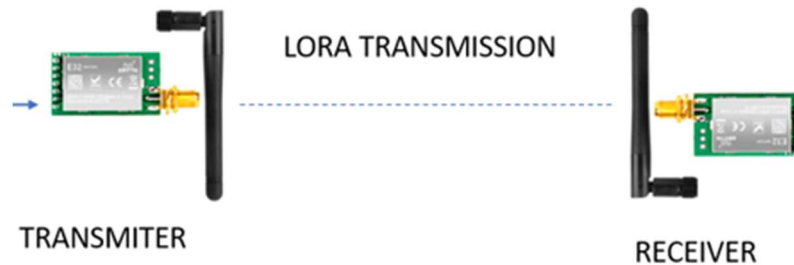


Figura 19: Módulos de transmisión LoRa

Cabe destacar que, aunque el enlace y la comunicación lo soporte, el intercambio de datos será unidireccional ya que se prohíbe por normativa el intercambio de datos por parte de los ingenieros al monoplaza pero no del monoplaza al exterior.

Una vez recibidos los datos, se procede a su tratamiento, almacenamiento y visualización. El almacenamiento será llevado a cabo a través de un servidor propio propiedad del equipo. Este servidor guardará los datos en una base de datos a la cual se podrá acceder desde cualquier punto si se le da un acceso a internet y se cifran los datos mediante usuario y contraseña.

Además del almacenamiento en remoto, o cloud, el receptor deberá mostrar los datos en tiempo real, imitando en primer lugar el *dashboard* del piloto para recrear su vista y, en segundo lugar y más importante, añadir toda la información posible y que no puede ser procesada por el piloto durante su conducción, como puede ser el valor de cada pack de baterías, su voltaje, temperatura, etc....

5.1.4 ANÁLISIS DEL LENGUAJE UTILIZADO PARA LA VISUALIZACIÓN DE LOS DATOS EN TIEMPO REAL

Ambos dashboards, tanto el interno del monoplaza como el del box, han sido realizados en JavaScript. Esta es la conclusión de un análisis y comparación entre diferentes lenguajes de programación. Los lenguajes propuestos, debido a su popularidad y eficacia han sido Python y Javascript.

Python es un lenguaje de programación de alto nivel y fácil de aprender que se utiliza en una gran variedad de aplicaciones, incluyendo la visualización de datos en tiempo real en carreras de monoplazas. La principal ventaja de Python es su amplia gama de bibliotecas y herramientas, lo que permite a los ingenieros y analistas de datos trabajar de manera más eficiente y efectiva.

Para la visualización de datos en tiempo real, se pueden utilizar diversas bibliotecas de Python, como Matplotlib [45], Seaborn [46] y Plotly [47]. Matplotlib es una biblioteca de trazado en 2D que permite crear gráficos de alta calidad con una gran variedad de estilos y formatos. Seaborn es otra biblioteca de visualización que se centra en la creación de gráficos estadísticos atractivos y elegantes. Plotly es una herramienta interactiva de visualización de datos que permite crear gráficos en tiempo real y compartirlos en línea.

Por otro lado, JavaScript es ampliamente utilizado en el desarrollo web y cuenta con una gran variedad de bibliotecas para la visualización de datos en tiempo real, como D3.js [48], Chart.js [49] y Plotly.js [50]. Estas bibliotecas permiten crear gráficos y visualizaciones en tiempo real, que pueden ser utilizados en aplicaciones web y dispositivos móviles. Además, el hecho de que JavaScript sea el lenguaje de programación principal en los navegadores web lo hace una opción natural para la visualización de datos en tiempo real en un entorno web.

Algunas de las ventajas de utilizar JavaScript frente a Python en el desarrollo de aplicaciones web incluyen:

- **Velocidad de ejecución:** JavaScript se ejecuta directamente en el navegador del usuario, lo que lo hace más rápido que Python en ciertas tareas, especialmente en aquellas que involucran interacciones con la interfaz de usuario.
- **Flexibilidad:** JavaScript es un lenguaje de programación muy flexible y se puede utilizar para una amplia variedad de tareas, desde animaciones hasta el desarrollo de aplicaciones web en tiempo real.
- **Integración con HTML y CSS:** JavaScript se integra muy bien con HTML y CSS, lo que hace que el desarrollo de aplicaciones web sea más fácil y eficiente.

- Mayor oferta de bibliotecas y *frameworks*: JavaScript cuenta con una gran cantidad de bibliotecas y *frameworks* disponibles, lo que hace que sea más fácil para los desarrolladores encontrar herramientas para sus proyectos.

5.2 DISEÑO DEL SISTEMA

El diseño del sistema ha sido realizado haciendo un seguimiento de la ruta de los datos desde los sensores de dentro del monoplaza y distribuidos mediante los dos cables de BUS HIGH y BUS LOW por todo el vehículo. Estos datos viajan por todo el vehículo a través de las dos líneas de transmisión a los respectivos sistemas necesitados de esos valores. Por ejemplo, los valores de presión del pedal de aceleración son, en un primer momento, leídos por un sensor conectado a la máster ECU delantera. Una vez la máster ECU delantera tiene los datos, procede a transmitirlos mediante el módulo MCP25-15 al inversor el cual tiene integrado en sí mismo una entrada BUS CAN [51]. Una vez recibido los datos por el inversor, realiza una conversión y manda la orden de aumentar o disminuir la potencia del motor dependiendo de los datos recibidos por el BUS CAN desde el pedal.



Figura 20: MCP25-15 CAN bus module

Cabe destacar que todas las señales que se utilizan y propagan dentro del monoplaza van a través de este sistema de dos cables haciendo uso del protocolo BUS CAN. Es por esta razón que decidimos introducir un esclavo más en la comunicación CAN que es nuestra TELEMETRY ECU, la cual no es una participante más del intercambio de datos dentro del monoplaza si no que es solamente un sistema a la escucha de todos los mensajes que se envían por el canal.

Es en este paso donde encontramos el primer elemento de nuestro sistema de telemetría.

5.2.1 ORDENADOR A BORDO

A la escucha de todos los valores que se comunican por la comunicación CAN tenemos el ordenador de a bordo del vehículo, el cual se encarga de dos funciones fundamentales:

- Actuar como *data logger*: se encarga de guardar todos los datos que recibe por parte de todos los emisores en formatos de texto diferentes por el id del mensaje. Esta funcionalidad provee al sistema de una caja negra ya que si en algún momento falla la comunicación por radio se siguen almacenando los datos independientemente de su conexión con el exterior. Cabe destacar que, para hacer uso de estos datos, se podrán acceder a ellos una vez el monoplaza se encuentre estacionado y habiendo terminado las pruebas.
- Display a bordo de los datos más relevantes: en segundo lugar, este ordenador de a bordo es el encargado de crear las gráficas que verá el piloto en el dashboard. Las principales ventajas de un Display a bordo son:
 - Monitoreo de datos: Los equipos de Formula Student utilizan sensores y sistemas de adquisición de datos para recopilar información en tiempo real sobre diferentes parámetros del vehículo, como la temperatura del motor, la presión de los neumáticos, las RPM del motor, las vibraciones, la temperatura de los frenos, entre otros. Estos datos son fundamentales para el mantenimiento predictivo.
 - Análisis de datos: Los datos recopilados se analizan utilizando herramientas de análisis de datos y técnicas de aprendizaje automático para identificar patrones, tendencias y anomalías. Se pueden aplicar algoritmos y modelos estadísticos para predecir el desgaste, el rendimiento y la posible aparición de fallos en los componentes del vehículo.

- Planificación del mantenimiento: El análisis de datos permite a los equipos de Formula Student planificar el mantenimiento de manera más precisa y eficiente.

Para el diseño de la pantalla, se hicieron sesiones de feedback con los diferentes pilotos para ver tanto la distribución como los valores más críticos que deberían mostrarse y actualizarse en tiempo real. Estos valores deben ser pocos, muy reveladores e útiles ya que las dimensiones de la pantalla a bordo son muy limitadas.

Además de las sesiones de feedback de los pilotos, que se pueden ver reflejadas en la figura 21, se tuvieron reuniones con los distintos jefes de departamento para estudiar y mostrar también valores críticos los cuales deberían prevenir roturas, desgaste o incluso estrangulamiento térmico. El estrangulamiento térmico es un concepto relacionado con el mundo de la electrónica donde debido a la subida de temperatura interna o temperatura del medio, los equipos electrónicos comienzan a presentar fallos.

Además de este tipo de problemas, al ser un coche de propulsión a baterías de Litio, o de propulsión eléctrica, la salud de las baterías es vital para, en primer lugar, un buen desarrollo y descarga de las celdas de batería. Cabe destacar que además de la salud de estas para un buen desarrollo del monoplaça, la salud de las baterías es fundamental por dos motivos:

- Seguridad al piloto: Las baterías de litio utilizan compuestos químicos altamente reactivos para almacenar y liberar energía. Bajo condiciones normales de uso y mantenimiento adecuado, las baterías de litio son seguras. Sin embargo, existen situaciones en las que las baterías de litio pueden experimentar problemas, como cortocircuitos internos o sobrecalentamiento *thermal runaway* (descontrol térmico). Durante el *thermal runaway*, la batería puede liberar calor de manera descontrolada y producir gases inflamables. Si la temperatura aumenta lo suficiente, los materiales internos de la batería pueden entrar en ignición y causar un incendio. Estos incendios pueden ser difíciles de extinguir y generar humo tóxico.

- Coste de reparación: además de los posibles daños físicos al piloto, en condiciones extremas, pueden darse roturas o corrosiones donde la química de la celda se vea afectada.

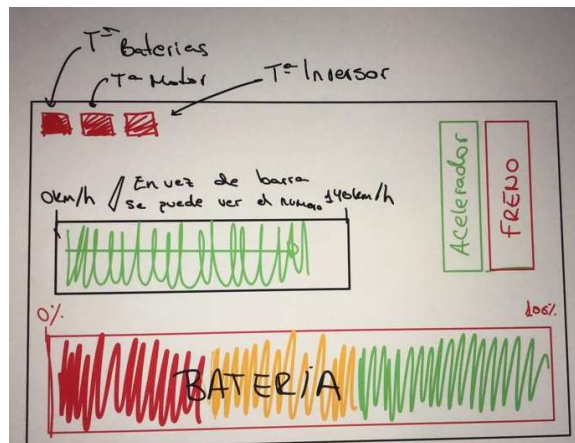


Figura 21: Feedback recibido por parte de los pilotos

Una vez hecho el diseño previo, se procede a la medida y encaje de la pantalla dentro del ángulo de visión del piloto. En un primer momento el diseño se planteó como una pantalla integrada dentro del mismo volante, similar a un formula 1 o cualquier simulador, pero por falta de espacio y presupuesto, se decidió que el diseño del volante y de la pantalla fuesen por separado. Esto añade un punto más de adaptabilidad y modularidad al sistema ya que no se precisa de una pantalla en específico ni de un diseño único de volante que implemente una pantalla o Display específico.

Una vez conscientes de las necesidades y especificaciones además de sus limitaciones, se procede al diseño e integración de la pantalla dentro del chasis. La implementación por parte del software y la solución planteada se describe en la implementación de la misma.

5.2.2 ENVIÓ Y RECEPCIÓN DE DATOS

Uno de los pilares del sistema planteado es la lectura y análisis en tiempo real de las señales leídas por los sensores en el vehículo. Estas señales deben ser procesadas, guardadas y visualizadas de manera óptima para los ingenieros de pista al mismo tiempo que el monoplaza está siendo utilizado en test o carrera. Por lo tanto, para cumplir esta especificación debemos contar con un sistema de transmisión y recepción de datos que pueda soportar tanto las especificaciones que debe cumplir, como el medio en el que va a ser utilizada.

Las especificaciones más importantes a la hora de diseñar el intercambio de datos y la radio son las de optimizar:

- Latencia: tiempo que tarda un paquete de datos en viajar desde su origen hasta su destino a través de una conexión inalámbrica. Depende de factores como la distancia entre el transmisor y el receptor, la calidad de la señal, la congestión en la red, la tecnología utilizada y otros factores ambientales. Esta no puede ser eliminada en transmisiones inalámbricas ya que solo por la propagación en un medio, la señal ya tiene un retraso temporal. No obstante, siempre que esta se mantenga baja, no supone un problema para el intercambio de datos en tiempo real.
- Ancho de banda: cantidad de datos que se pueden transmitir durante el intercambio de datos entre transmisor y receptor. Para transmisiones en tiempo real, queremos que el intercambio de datos sea alto, pero se priorizará la velocidad de transmisión o la latencia ya que los valores no son de un tamaño elevado (pocos bits por lectura) pero si necesitamos el valor en el menor tiempo posible.
- Baja tasa de errores e interferencias: dependiendo de la tecnología utilizada y la frecuencia de funcionamiento, podemos encontrar diferentes comportamientos en la radio. Por ejemplo, cuanto menor sea la frecuencia, mayor ancho de banda y a la

inversa, mayor frecuencia menor ancho de banda. Otras limitaciones, errores o interferencias pueden ser debidas a:

- Errores de bit: Son errores que afectan a los bits individuales de los datos transmitidos. Pueden ser causados por ruido en la señal, interferencias electromagnéticas, atenuación de la señal debido a obstáculos físicos, fluctuaciones en la potencia de la señal, entre otros factores. Estos errores pueden provocar la corrupción de la información transmitida, lo que puede afectar la calidad de la comunicación.
- Interferencia co-canal: Ocurre cuando dos o más transmisores utilizan la misma frecuencia o canal de transmisión. Esto puede causar interferencias mutuas y afectar la calidad de la señal. La interferencia co-canal puede ser causada por transmisores cercanos o por la presencia de otras fuentes de radiofrecuencia en el entorno.
- Interferencia de canal adyacente: Ocurre cuando la señal de transmisión se superpone o se solapa con señales en frecuencias cercanas. Esto puede ocurrir debido a la falta de separación adecuada entre los canales de transmisión o a la presencia de otras transmisiones cercanas. La interferencia de canal adyacente puede provocar distorsión en la señal y aumentar la tasa de errores.
- Interferencia de múltiples trayectos: Es causada por la reflexión, difracción y dispersión de la señal en el entorno. Estas señales reflejadas pueden llegar al receptor con un retardo y una fase diferente, lo que puede interferir con la señal original y causar distorsiones. La interferencia de múltiples trayectos es común en entornos urbanos o con obstáculos físicos.

En el caso de nuestro sistema, podemos llegar a encontrar con todas las limitaciones descritas.

- Alcance: el alcance de la señal, como su propio nombre indica, nos muestra el espacio máximo al cual podemos transmitir la señal. Existen varias tecnologías, protocolos y herramientas de hardware para la transmisión inalámbrica que nos proporcionan alcances desde pocos centímetros hasta miles de kilómetros. Una vez más, deberemos adaptar las necesidades del proyecto a la selección de la tecnología, protocolo y hardware seleccionado. Dependiendo de la tecnología, la antena y frecuencia, el alcance se limita en distancia. En este caso, y tras haber hecho un estudio estimado del alcance máximo necesitado, necesitamos un sistema de transmisión seguro, fiable y rápido para una distancia no superior a 2 kilómetros. Querer tener mayor alcance sería ineficiente ya que no se pondría en práctica y utilizaríamos más energía de la necesaria y hardware más costoso.
- Limitaciones del hardware: al utilizarse dentro de un monoplaza de Formula Student, los recursos son limitados y el espacio también, por lo que, el valor computacional de los sistemas de dentro del vehículo será menor que el que se tenga en un ordenador o servidor.

Además de todas estas especificaciones “típicas” para las transmisiones inalámbricas, al ser en un monoplaza de competición, encontramos una limitación más: la transmisión de los datos cuando el emisor se está moviendo a gran velocidad. Esta limitación es clave para la comunicación ya que nos impide usar señales y tecnologías que empleen frecuencias bajas. Esto ocurre ya que, como se ha mencionado previamente, existe el efecto Doppler.

Una vez decidido la tecnología y el protocolo, el diseño es sencillo e intuitivo.

Para la transmisión queremos recibir y todos los valores que circulen por el BUS Can, por lo tanto, lo que deberemos realizar es, al igual que con el ordenador de a bordo, un esclavo más de la comunicación que no participe en la misma sino que simplemente se dedique a transmitir todo lo que recibe.

En un primer momento, el diseño era una radio controlada por el ordenador de a bordo, pero la idea fue descartada ya que añadía complejidad y lentitud de procesamiento al propio ordenador ya que tenía que hacer más operaciones. Esta complejidad aportaba retrasos considerables a la visualización de los datos en tiempo real al piloto, lo cual era inasumible. El cambio a realizar en ese diseño sencillamente añade un receptor más del BUS Can y un Arduino que controle la radio, por lo que la alternativa nos aporta una velocidad muy superior (al utilizar C en el Arduino) y un coste mínimo del Arduino y otro receptor MCP2515.

En la recepción, la antena al igual que en emisión, será controlada por un Arduino ya que nos proporciona gran velocidad y robustez en la tecnología utilizada, siendo la misma que en transmisión.

Por último, la recepción finaliza cuando los datos son recibidos por el servidor u ordenador que procesa los datos en la llegada, que guarda los valores en el servidor interno y hace el Display con la información de los sensores de manera más amplia y detallada que la realizada al piloto.

Esta última recepción se hace mediante la escritura de los datos recibidos por la radio a un puerto del ordenador o servidor. Esta conexión puede ser USB, Ethernet o incluso fibra óptica. En este caso, la conexión utilizada ha sido USB ya que no existen diferencias con el resto.

5.2.3 DISEÑO DEL TRATAMIENTO DE LOS DATOS EN LA RECEPCIÓN, ALMACENADO Y VISUALIZACIÓN EN TIEMPO REAL.

Una vez recibidos los datos emitidos desde el coche, el servidor u ordenador receptor debe poder leerlos del puerto en el que se están escribiendo y poder así realizar todas las tareas que se necesitan para el correcto funcionamiento del sistema de telemetría.

El diseño de esta última etapa está dividido en tres partes fundamentales:

- Diseño de la recepción y lectura de los datos

- Diseño de la visualización
- Diseño del almacenamiento de los datos

5.2.3.1 Diseño de la recepción y lectura de los datos

La recepción de los datos se realiza de la misma manera que la emisión: un Arduino controla una antena y usa la misma frecuencia que la utilizada en la emisión. Esto se realiza de esta manera para aportar unicidad y simpleza para el desarrollador y velocidad en la transmisión y lectura. La unicidad resulta útil ya que no hay dos partes desarrolladas en lenguajes diferentes o en sistemas o hardware distintos. Todo lo relacionado con la transmisión se expresa en C ya que es el lenguaje “popular” más veloz.

Para la recepción de los datos por parte del servidor u ordenador final de la recepción, en un primer momento se realizó un diseño completo en Python, el cual cumplía todas las especificaciones que necesitaba en dicho momento.

Las especificaciones iniciales que tiene la recepción son la de poder leer los valores que se reciben en uno de los puertos del dispositivo y almacenarlos de manera local. Con estas especificaciones en un primer momento se utilizó el lenguaje más simple y popular actualmente como lo es Python. Este primer script recibía los datos y almacenaba los valores en archivos de texto plano (.txt).

Ya con una primera versión funcional pero limitada se deciden implementar mejoras en la velocidad de lectura del puerto, velocidad de guardado, mejora en el guardado optimizándolo en bases de datos y realizar una visualización en tiempo real de los valores.

5.2.3.2 Cambio entre el backend y almacenamiento en Python a recepción y peticiones en JavaScript

Viendo las nuevas especificaciones se decide comenzar de cero el diseño de la lectura de los datos y hacerlo mediante el uso de JavaScript.

Este cambio de lenguaje se debe a dos motivos fundamentales: mayor velocidad y lenguaje enfocado a la visualización web.

Además, JavaScript ofrece programación del lado del cliente, lo que significa que se ejecuta en el navegador web del usuario. Es especialmente útil para agregar interactividad a las páginas web, como animaciones, efectos visuales y respuestas en tiempo real además de quitarle complejidad al servidor. También ofrece una amplia gama de APIs para manipular el Document Object Model (DOM) de una página web. Esto permite realizar cambios dinámicos en los elementos de la página, como agregar o eliminar elementos, modificar estilos y contenido, y responder a eventos del usuario.

Otro punto fundamental de JavaScript es la existencia de *frameworks* y bibliotecas: dedicados a la visualización de datos en la web, como D3.js, Chart.js y Three.js. Estas herramientas ofrecen una gran flexibilidad y capacidad para crear visualizaciones personalizadas y de alta calidad. Cabe destacar que en este proyecto se ha utilizado Chart.js.

Además, JavaScript cuenta con Node.js, un framework ampliamente utilizado por la comunidad el cual aporta gran versatilidad al proyecto. Se considera un entorno de tiempo de ejecución de JavaScript del lado del servidor que permite ejecutar código JavaScript fuera de un navegador web. A diferencia de JavaScript en el navegador, que se utiliza principalmente para interactuar con el DOM y crear interacciones en el cliente, Node.js está diseñado para ejecutar código JavaScript en el servidor y facilitar la creación de aplicaciones web y de red. Node.js utiliza el motor V8 de Google Chrome, que es un motor de ejecución de JavaScript de código abierto y altamente eficiente, lo cual permite que se ejecute código de manera rápida y eficiente.

Node.js se basa en un modelo de programación no bloqueante y basado en eventos. Esto significa que puede manejar múltiples solicitudes simultáneamente sin bloquear el flujo de ejecución. Esto lo logra utilizando devoluciones de llamada (callbacks) y promesas para manejar las operaciones asíncronas, lo que permite una escalabilidad y rendimiento superiores en aplicaciones web. también cuenta con módulos que permiten organizar y reutilizar código de manera efectiva. Además, Node.js se integra con el Administrador de paquetes de Node (NPM), que es un repositorio público de paquetes de código abierto que permite a los desarrolladores compartir, descargar e instalar fácilmente módulos y bibliotecas adicionales.

Gracias a estos módulos se simplifica enormemente la programación, gestión de puertos y llamadas desde la recepción de los datos.

Finalmente, la utilización de Node.js nos permite el desarrollo de aplicaciones web y de red. Con Node.js, se pueden construir servidores web, APIs, servicios de *backend*, aplicaciones en tiempo real y mucho más. Además, se utiliza un framework como Express.js para simplificar el desarrollo de la aplicación web con Node.js y así simplificar la visualización de los datos.

5.2.3.3 Diseño de la visualización de datos

Para la visualización de los datos se ha hecho uso de los mismos módulos Node.js utilizados previamente en la visualización *on-board* con la excepción de que en la comunicación por radio perdemos el protocolo original de transmisión y recepción de mensajes BUS Can. Ahora, sin embargo, utilizamos un Arduino que escribe los valores en un puerto del servidor u ordenador.

Para su lectura de datos también se hace uso de los módulos Node los cuales nos permiten abrir uno o varios de los puertos del receptor desde una interfaz controlada por un lenguaje basado en JavaScript.

Estos puertos nos permiten intercambiar información, en forma de objetos en JavaScript los cuales contienen en sus valores los datos que se quieren mostrar.

Una vez establecida la conexión del módulo o puerto receptor de datos desde el Arduino, se diseña una página HTML la cual mostrara dichos valores por pantalla en tiempo real. Una vez más, todas estas conexiones se realizan gracias a Node.js.

Una vez con los datos accesibles desde un entorno, se procede al diseño de las gráficas y disposición en página de los valores a mostrar y su manera de ser representados. Por ejemplo, los valores críticos de temperatura se mostrarán como un valor único el cual modificará su color e intensidad conforme vayan siendo más críticos.

Además, en esta etapa de diseño de la visualización, debemos contemplar la tasa de refresco y la fluidez que queremos obtener en nuestras gráficas ya que dependiendo del número de gráficas a actualizar y la fluidez deseada, se utilizará una tasa de refresco u otra.

5.2.4 DISEÑO DEL MÓDULO DE PREDICTIVE MAINTENANCE

Como valor añadido al proyecto, se incluye un módulo de *predictive maintenance* o mantenimiento predictivo. El mantenimiento predictivo es una estrategia de mantenimiento que se basa en la monitorización continua de los equipos y sistemas para detectar posibles fallos o problemas antes de que ocurran. Se utiliza la recopilación de datos en tiempo real, así como técnicas de análisis y pronóstico, para predecir el comportamiento futuro de los equipos y tomar acciones preventivas o correctivas de manera proactiva.

A diferencia del mantenimiento reactivo (reparar una falla después de que ocurre) o el mantenimiento preventivo (realizar tareas de mantenimiento de manera periódica sin considerar el estado real del equipo), el mantenimiento predictivo se centra en la gestión eficiente de los recursos y la reducción de costos asociados a paradas inesperadas o reparaciones innecesarias.

Se basa en la idea de que los equipos y sistemas muestran signos tempranos de deterioro o comportamiento anormal antes de fallar. Al utilizar sensores, instrumentación y sistemas de monitorización, se recopilan datos sobre variables relevantes, como vibración, temperatura, consumo de energía, presión, entre otros, para realizar un seguimiento continuo del estado de los equipos.

Estos datos se analizan mediante técnicas de análisis de datos y algoritmos de aprendizaje automático para detectar patrones, tendencias o anomalías que puedan indicar un fallo inminente. Con base en esta información, se pueden tomar acciones preventivas, como realizar un mantenimiento programado o reemplazar componentes antes de que ocurra una falla crítica.

Ayuda a maximizar la disponibilidad y confiabilidad de los equipos, reducir los tiempos de inactividad no planificados, minimizar los costos de mantenimiento y prolongar la vida útil de los activos. Además, al predecir y programar el mantenimiento de manera eficiente, se evita la interrupción innecesaria de la producción o servicios.

En este caso, se presenta el diseño de un módulo de mantenimiento predictivo para las baterías del monoplaza. En el mantenimiento predictivo de baterías, se utilizan varios datos para monitorear y predecir su rendimiento. Algunos de los datos comunes que se consideran incluyen:

- **Voltaje:** El voltaje de la batería es un indicador clave de su estado de carga y rendimiento. La medición periódica del voltaje puede ayudar a identificar cambios significativos en la capacidad de la batería.
- **Corriente:** La corriente que fluye dentro y fuera de la batería durante la carga y descarga también proporciona información valiosa. Los patrones de corriente anormales pueden indicar problemas o desgaste en la batería.
- **Capacidad:** La capacidad de la batería se refiere a la cantidad de energía que puede almacenar y suministrar. La medición regular de la capacidad puede revelar cambios en el rendimiento a lo largo del tiempo.
- **Temperatura:** La temperatura de la batería es un factor crítico que puede influir en su rendimiento y vida útil. El monitoreo de la temperatura puede ayudar a detectar condiciones de funcionamiento inadecuadas o anomalías térmicas.
- **Ciclos de carga y descarga:** Registrar el número de ciclos de carga y descarga que ha experimentado la batería es importante para evaluar su desgaste y capacidad residual.

Las baterías tienen una vida útil limitada, y el seguimiento de los ciclos puede ayudar a predecir cuándo es probable que se degrade.

- **Tiempo de funcionamiento:** Registrar el tiempo de uso de la batería puede proporcionar información sobre su vida útil y patrones de consumo de energía. Esto es especialmente relevante para baterías que se utilizan en dispositivos portátiles o sistemas autónomos.

Además de estos datos básicos, también se pueden considerar otros parámetros específicos según el tipo de batería y la aplicación. Algunos ejemplos adicionales incluyen la resistencia interna de la batería, el nivel de electrólito o electrolito, la densidad de energía, entre otros.

Para el caso del mantenimiento predictivo de baterías se pueden encontrar diversos estudios y proyectos como uno realizado por la Universidad de Cambridge [52], el cual, relaciona una duración de hasta diez veces más a aquellas celdas de baterías que han sido previamente monitorizadas y con mantenimiento predictivo por parte de inteligencias artificiales que aquellas que no lo habían sido. Es interesante destacar que aquellos sistemas que habían sido tratados mediante un mantenimiento predictivo, sus ahorros eran de aproximadamente de un 30% a 40% mientras que con mantenimiento preventivo, esta cifra disminuye a entre 8% y 12% [53].

Si se eligen los sistemas de mantenimiento predictivo para la protección de baterías, se sabrá exactamente cuándo llegará el final de la vida útil del paquete de baterías y se tendrá tiempo suficiente para sustituirlo. Estas son algunas de las principales ventajas de adoptar este sistema de protección de baterías:

1. Se requiere un menor tiempo de inactividad en el mantenimiento de los equipos
2. Uso eficaz y óptimo de la batería hasta el EOL (End-Of-Life)
3. Eliminación de celdas defectuosas.
4. Mejora del progreso y la sostenibilidad general de la batería.
5. Pérdida mínima de horas productivas

6. Reducción de costes en mantenimiento no deseado y sustitución de piezas de repuesto

Y los pasos a seguir antes de realizar un mantenimiento predictivo son:

- Identificar los componentes críticos del pack de baterías que requieren someterse a una comprobación de mantenimiento.
- Mantener un registro sistemático, conservando un historial de todas las actividades detectadas por el sistema de gestión de baterías.
- Evaluar las anomalías actuales en el conjunto existente de actividades de la batería, como las fluctuaciones de temperatura y tensión en estado de reposo y durante la carga.
- Realizar auditorías relacionadas con el análisis modal de fallos y efectos del proceso (PFMEA).
- Clasificar diferentes sistemas de mantenimiento predictivo para seleccionar la tecnología de mantenimiento predictivo que se ajuste a los requisitos.

Una vez se cumplen todos los requisitos previos del sistema, se puede proceder a la elaboración del sistema o módulo del mantenimiento predictivo.

Para la elaboración del módulo del mantenimiento predictivo, hacemos uso del artículo recientemente publicado de la universidad de Chicago, la cual realiza un estudio estadístico de predicción del mantenimiento de 292 baterías después de almacenar sus valores más de dos años [54].

Normalmente, existen dos tipos de métodos para la detección de fallos y el mantenimiento de las baterías. El primero de ellos, la inspección manual, es cuando los expertos realizan pruebas in situ para deducir el estado de salud y la vida útil restante de las baterías. El *discharge test* es la forma más directa y precisa de detectar la capacidad de la batería, pero requiere complicados procedimientos de medición. Este método sólo puede proporcionar una evaluación aproximada del estado de la batería, ya que la conductividad es sólo uno de los numerosos factores que pueden afectar a su estado.

Otro tipo es la detección automática en línea, es decir, se utiliza un sistema automático, como un instrumento de comprobación de baterías con algún algoritmo diseñado, para detectar el estado de la batería.

Algunas pruebas que se realizan para este tipo de detección son: *Tensión de circuito abierto* que es una indicación relativamente precisa de la capacidad restante. Pero la batería debe separarse del sistema para medir la tensión en circuito abierto o *Resistencia interna* demostración que existe una relación directa entre la resistencia interna y la vida útil restante de las baterías VRLA. Sin embargo, la resistencia interna varía entre lotes de baterías y fabricantes, lo que dificulta la formulación del método.

Entre estos métodos de control, las investigaciones sobre el estado de salud (SOH) (Shahriari y Farrokhi (2013))[55] tienen como objetivo averiguar cuántas veces se puede cargar y descargar una batería mientras suministra la energía necesaria. Asimismo, la predicción de la vida útil restante (RUL, por sus siglas en inglés) (Ren et al. (2018); Gregory W. Ratcliff (2019)) es otro tema de interés.

En el modelo propuesto, se recogen aproximadamente 240.000.000 de datos de 292 baterías monitoreadas durante más de 30 meses. Se almacenan los tiempos de medición, el número de serie de la celda y el número de serie del pack y de la batería además de los atributos electrónicos previamente explicados como el voltaje, la temperatura, la resistencia interna o la corriente.

Tras este almacenado masivo de los datos, se procede a un etiquetado de los mismo, Una batería se clasifica como defectuosa cuando se produce un evento que conduce a su sustitución; a este evento lo denominamos Evento de Interés (EoI).

Un EoI suele ser de los siguientes tipos:

- Envejecimiento natural: La resistencia óhmica de la batería aumenta suavemente con el envejecimiento de la batería, causando la atenuación de la capacidad de la batería. De acuerdo con las normas de la industria, se recomienda separar una

- batería del sistema cuando su resistencia alcance los 5 mΩ. Este fallo se puede ver reflejado en la figura 22 (a)
- Fallo interno. Un fallo interno puede provocar un deterioro drástico del estado de la batería, lo que pone en peligro no sólo el pack de baterías, sino también la seguridad. En la figura 22 (b) vemos este segundo tipo.

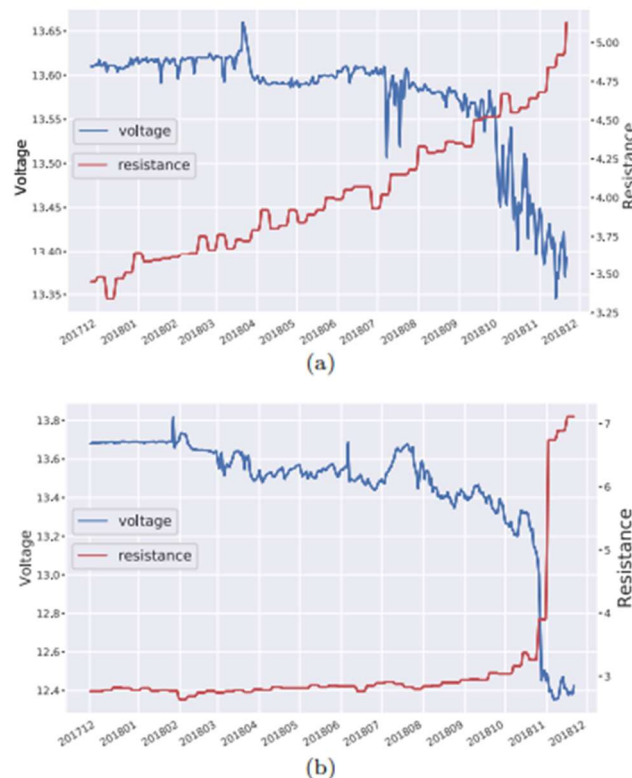


Figura 22: Tipos de error en baterías

Para el caso del Envejecimiento Natural cuando resistencia aumenta suavemente (alcanza los 5 mΩ). Para el caso de Fallo interno, la tensión de flotación de la batería disminuirá y fluctuará drásticamente en la fase inicial de fallo, lo que es indicativo de una sustitución de la batería.

En primer lugar, se debe definir un indicador de disminución de la tensión de flotación:

$$D_t = \sqrt{\sum_{i=1}^W w_i \cdot \min\{V_{t-i} - \text{Median}[V_{(t-2M):(t-M)}], 0\}^2}$$

Figura 23: Definición de la tensión de flotación

A partir de la figura 23, encontramos la definición de dicha fórmula, donde la M y W son tiempos, *Median*, es la mediana de $\{V_i | i=t-2M, t-2M+1, \dots, t-M-1\}$ y w son los pesos de cada uno de los valores, siendo la suma de total de ellos 1. Debemos definir el punto de cambio que será el encargado de indicarnos cuando es el momento óptimo de reemplazo de baterías. Este viene definido por:

$$t_c = \min_t \{t | D_t \geq 3\sigma_e\}$$

Figura 24: definición del punto de cambio

La definición del punto de cambio, viene definida en la figura 24, donde D_t es la disminución de la tensión de flotación, previamente explicada, y σ_e , la media de la desviación estándar empírica de los valores de la tensión de flotación de todas las muestras consideradas como sanas u óptimas.

El punto de cambio, por lo tanto, será cuando el indicador de la tensión de flotación exceda tres veces la desviación estándar mínima.



Figura 25: Funcionamiento de la predicción del punto de cambio

En la figura 25, podemos ver como el punto de cambio se sitúa en días previos a la degradación y malfuncionamiento de la celda en el estudio realizado en el artículo.

Una vez que podemos determinar el punto de cambio, ya hemos dotado a nuestro sistema de un mantenimiento predictivo que ayuda al ahorro, buen funcionamiento y optimización y eficiencia de las celdas de batería.

Ahora tenemos una herramienta de clasificación de datos que puede ejecutarse en tiempo real gracias a los valores previamente almacenados y calculados.

Es importante tener en cuenta que los datos recopilados deben ser analizados y comparados a lo largo del tiempo para identificar tendencias y patrones que indiquen cambios en el rendimiento de la batería. El uso de sistemas de monitoreo continuo y la aplicación de técnicas de análisis de datos avanzadas pueden ser beneficiosos para obtener información más precisa y realizar pronósticos más confiables.

5.3 IMPLEMENTACIÓN

Desde el punto de vista de la implementación, también se puede hacer la misma división hecha en el apartado del diseño siguiendo la ruta cronológica de los datos desde su origen, los sensores del vehículo hasta su visualización en los *displays* del piloto y del garaje o ingenieros.

No obstante, previo a la implementación y sus diferentes partes y especificaciones, debemos remarcar en qué condiciones se realiza la implementación del diseño del proyecto.

5.3.1 LIMITACIONES EN LA IMPLEMENTACIÓN EN EL MONOPLAZA DE LA UNIVERSIDAD DE ILLINOIS

El vehículo de competición eléctrica de la universidad de Illinois es relativamente nuevo por lo que cuentan con falta de experiencia en el campo de la Formula Student electric. Debido a esto, el prototipo de este año, 2023, cuenta con grandes problemas de aislamiento eléctrico. Esto viene originado por la migración de un solo motor eléctrico a cuatro. Los principales retos de la migración de uno a dos o cuatro motores es el aislamiento eléctrico y electrónico para que los cables de alta tensión que van dirigidos a cada una de las ruedas o motores no

interfieran con los cables de baja tensión por los que el coche comunica su telemetría entre tantos valores.

Cuando un cable, o conexión transporta alto voltaje, sus campos magnéticos que se crean debido a estos voltajes pueden interferir con los cables de baja tensión comprometiendo así los valores que circulan por los mismos. Este problema provocó que el sistema de telemetría aquí propuesto no pudiese llevarse a cabo ya que la comunicación interna es muy limitada.

La solución propuesta para este proyecto fue la creación de un entorno simulado para recrear valores pseudo-reales para poder desarrollar interfaces gráficas, conexiones a bases de datos, etc...

5.3.2 DESARROLLO ENTORNO DE PRUEBAS

Como previamente se ha explicado, para la correcta realización y el correcto desarrollo del proyecto, se ha tenido que desarrollar un entorno de pruebas simulado el cual genere los datos simulados del vehículo.

Como no se pueden estimar a que frecuencia se enviarán los datos de manera interna dentro del vehículo, deberemos suponer siempre la velocidad más alta. Para conseguirlo, se hace uso del hardware comercial más rápido: ARDUINO[56]. Este hardware, ya descrito en el apartado de descripción de las tecnologías, es el óptimo para la generación y transmisión a gran velocidad de datos al BUS Can.

Para la creación del entorno simulado, primero debemos saber que datos debemos simular y como se van a transmitir a los diferentes ordenadores, tanto el de abordo como el receptor de la señal.

Los datos se generarán de manera aleatoria en un primer lugar. Más tarde se modifican a valores cercanos a los reales o estimados. En segundo lugar, todos estos valores deben ir asociados a un sensor como previamente se ha explicado. Todos los valores por lo tanto

tendrán un identificador único que será utilizado para ser transmitida y leído dentro de la comunicación Can. En la siguiente figura, se muestra el reparto inicial de los identificadores de los sensores.

tensión inversor	0x100	Inversor	0x100-0x109
temperatura motor	0x101	Refri	0x110-0x119
temperatura inversor	0x102	Baterias	0x120-0x129
temperatura refri in	0x110	Dinámica	0x130-0x139
temperatura refri out	0x111		
presión aero refri	0x112		
temperatura baterías	0x120		
acelerometro x	0x130		
acelerometro y	0x131		
acelerometro z	0x132		
aceleración pedal	0x133		
presión frenos	0x134		

Tabla 1: Reparto provisional de identificadores de sensor

Todos estos sensores serán simulados y sus datos generados de manera aleatoria serán la base sobre la cual se diseñarán la visualización, transmisión y almacenamiento de los datos.

Finalmente, el entorno de pruebas deberá simular también la escritura en los diferentes protocolos y transmisión dentro de los diferentes protocolos. Por lo tanto, en el caso del ordenador de a bordo, los mensajes y valores deberán transmitirse mediante una conexión BUS Can, haciendo uso de los módulos MCP25-15 mientras que en el caso de la recepción y transmisión al ordenador receptor solo hará falta la escritura en uno de los puertos mediante los serial port print.

En el caso de la conexión con el protocolo BUS Can, se deberá llevar a cabo la conexión el módulo MCP25-15. Para ello se consulta el *pinout* del Arduino Zero, y su conexión con el módulo MCP25-15 y su manual de uso, para ver los comandos y librerías necesarias para la propagación de los mensajes[57].

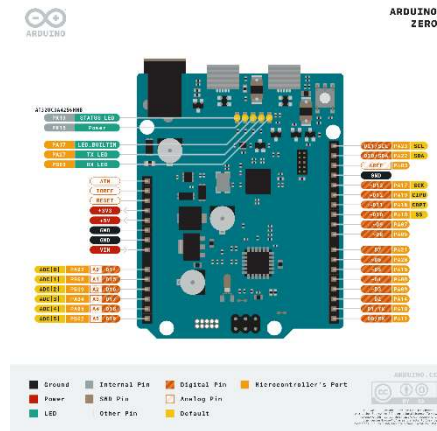


Figura 26: Pinout Arduino Zero

Una vez comprendida la comunicación con el resto de los elementos, debemos entender que los datos se generan en un bucle continuo y se asignan a todos los identificadores. Una vez los mensajes generados, solo es necesario hacer uso de la comunicación y protocolo para la transmisión del mismo.

Destacar también que previo a su generación y transmisión, se deben inicializar la comunicación escogida, en el caso de la comunicación por BUS Can, deberán incluirse las librerías *spi.h* y *mcp25-2* e inicializar el *bitrate* a 125kbps[57]. En el caso de la escritura en un puerto del ordenador, deberemos inicializar la velocidad de escritura a 115200 baudios. Este será el entorno de simulado que se utilizará a lo largo de todo el desarrollo del proyecto.

5.3.3 IMPLEMENTACIÓN SIMULADA DEL *DASHBOARD* DEL PILOTO

Para la implementación previamente diseñada en el capítulo anterior, se realiza la conexión del entorno de pruebas con el ordenador de a bordo. Este ordenador, como ya se ha visto en el capítulo de diseño, debe cumplir las funciones de almacenamiento y visualización de los datos en tiempo real.

En el desarrollo del ordenador de a bordo, primero se debe hacer la conexión del ordenador del piloto al resto de sistemas que operan en el vehículo. Esta conexión se realiza mediante

el protocolo previamente descrito: BUS Can. Todos los mensajes del vehículo circulan a través de este protocolo por lo que el entorno de pruebas también deberá transmitir los mensajes a través del BUS Can.

El hardware seleccionado para cumplir con las especificaciones previamente descritas en una RaspberryPi ya que reúne tanto velocidad en lectura y escritura como gran capacidad de generar un entorno gráfico [58], imprescindible para la pantalla del piloto que debe actualizarse en tiempo real.

Para cumplir ambas funciones, se desarrollan dos scripts que trabajarán en paralelo. El primero de ellos estará desarrollado en Python y su función principal es la lectura del puerto y escritura en archivos de texto (formato “ID.txt”)

Cada valor que llegue al ordenador de a bordo, este script será el encargado de leer y almacenar los datos en su archivo de texto correspondiente. Este script se ejecutará nada más encenderse el coche. El funcionamiento del script se resume en: cuando un nuevo dato llega al ordenador de a bordo, es decir, se transmite por el BUS Can, este primero de todo revisa su identificador único de sensor y comprueba que no exista ningún archivo con dicho nombre de identificador, si es así, lo crea y registra el valor con la fecha y hora de llegada. En caso contrario, accede al archivo en modo lectura y escritura y añade al final del archivo una línea más donde figura el valor con el tiempo de escritura. Cada valor se registra en su archivo con el nombre único de su identificador y se registra con el tiempo en el que se ha escrito.

Esto nos permite tener o generar una especie de caja negra o *data logger* dentro del propio vehículo, el cual registrara todos los datos en s ordenador de a bordo y una vez finalizado el rodaje o las pruebas, se accederá al ordenador mediante una conexión wifi haciendo uso del protocolo *putty*, y descargarán los datos en caso de que haya fallado la comunicación inalámbrica o se quiera ver si dicha comunicación ha tenido fallos.

En segundo lugar, se realiza un script el cual es capaz de generar un entorno gráfico que actualice en tiempo real el estado del vehículo al piloto y puede dar de manera clara e intuitiva los valores necesarios para una conducción eficiente y segura del monoplaza.

En un primer momento, se decide realizar este script haciendo uso de Python y las librerías Dash, propiedad de Plotly, extremadamente útiles para la visualización de datos en tiempo real[59]. Esta decisión pretendía aportar uniformidad al software ya que estaría desarrollado todo en el mismo lenguaje.

Una vez finalizado el diseño y la implementación dentro del monoplaza, los tiempos de respuesta y la definición de las gráficas obligaron a la modificación del script. Debido al limitado hardware del ordenador de a bordo: RaspberryPi Zero, la generación de las gráficas suponía un problema ya que, aunque estas funcionasen de manera perfecta con hardware más avanzado (RaspberryPi 3 en adelante) en la placa actual no conseguían generarse a la velocidad necesaria para un correcto funcionamiento.

Una vez reconocido el problema, se estudian otras soluciones como la visualización de datos únicamente en vez de graficas o la migración a otro lenguaje de programación más optimizado y enfocado a la generación de entornos gráficos. La primera de las ideas se pone en práctica con un script provisional para ver si podía ser de utilidad y ser la solución buscada pero al no ser intuitivo y con valores con cambios constantes, se decide la migración al lenguaje de programación más enfocado a la generación del entorno gráfico JavaScript en vez de Python [60].

Una vez decidido el cambio, se lleva a cabo la migración del entorno gráfico con todo lo que ello suponía. Se debían ejecutar en lugar de un solo script en Python, dos scripts.

El segundo de ellos consiste en la generación del entorno gráfico optimizado para los recursos hardware disponibles. Para ello, se necesita de la instalación de los recursos software para la lectura y procesado de las gráficas en tiempo real. Para ello se hacen uso de los módulos Node, previamente descritos en el capítulo previo.

Gracias a Node.js se realiza de manera más sencilla y optimizada la lectura del bus can conectado a la RaspberryPi Zero [61]. Esta lectura permite acceder desde JavaScript a los datos que llegan desde el resto de los sistemas del vehículo y así poder generar el entorno gráfico.

Una vez ya accedidos a los datos provenientes del resto de sistemas, se busca realizar una estructura de *backend* y *frontend*, el cual procesa en un primer momento los datos que llegan desde la lectura del puerto del BUS Can y dependiendo de los identificadores que lleguen, decide enviarlos al *frontend* para su actualización y en tiempo real.

La creación del sistema de *back* y *frontend* se realiza con el uso de otro modulo Node.js el cual permite crear dicha conexión, así como un sistema de página HTML la cual se muestra cuando se accede al puerto donde se está ejecutando la conexión entre servidor y *frontend*.

Para el correcto funcionamiento del sistema, se tendrá que hacer uso de los módulos *express* [62] y *socket.io* [63] los cuales permiten la visualización de datos y graficas accediendo al puerto de ejecución y la conexión entre *frontend* y *backend* respectivamente.

En el primer archivo, *server.js*, que actuará como servidor o *backend*, se requerirán todos los módulos que se necesiten para el correcto funcionamiento del software. Entre estos destacan los previamente mencionados *express* y *socket.io* como el módulo de lectura de datos del BUS Can, *socketcan* [64].

Cuando ya han sido importados o requeridos (en la jerga de Node) se inicializa un objeto con todos los parámetros que se quieren intercambiar con la visualización de datos. Este objeto, dentro de este proyecto, recibe el nombre de *carInfo*. El objeto *carInfo* tendrá los valores de velocidad, porcentaje de batería, freno, aceleración, temperaturas de batería, inversor y motor, así como el valor de par que está ejerciendo el motor.

Ya con el objeto creado e inicializado a cero, crearemos las dependencias de para la página HTML de la visualización de los datos e indicaremos la ruta a seguir para encontrar los módulos que se utilizan, así como el HTML y JavaScript que debe ejecutarse en la visualización.

Una vez creadas la dependencias y la ruta de archivos para la lectura, comenzamos con el tratamiento de los datos leídos desde el puerto BUS Can previamente inicializado con la creación de la ruta de datos mediante el comando: `var channel = can.createRawChannel("can0", true);`. Con la instrucción creamos el canal que

posteriormente y mediante un *listener*, nos avisará de la entrada de un nuevo dato al ordenador.

Con la llegada del nuevo dato, se realiza un filtrado de todos los identificadores que son relevantes para la visualización, es decir, todos los identificadores que describan cualquiera de los parámetros inicializados a cero en el objeto *carInfo*.

Tras el filtrado, si el identificador del mensaje entrante coincide con alguno esperado, se modifica el valor asociado del objeto. Por lo tanto, cada vez que llega un nuevo mensaje del BUS Can, se realiza un filtrado y se actualiza en caso de que sea necesario el objeto que se va a visualizar.

Mientras se va realizando el filtrado de cada nuevo mensaje, se realiza paralelamente la emisión o transmisión del objeto *carInfo* al *frontend* para su visualización. Esta emisión la realiza el servidor cada 50 milisegundos para no crear apenas retraso de la llegada del actual valor a la visualización. El servidor además determina que puerto será el destinado no a la lectura de los datos si no al intercambio y generación de la visualización de datos y avisos al piloto.

Mientras tanto, en el *frontend*, solamente se debe especificar el puerto en el que figura abierto y en el que esta realizando el intercambio de datos. Esto nos permitirá acceder relativamente fácil a los datos que provenientes del vehículo y por lo tanto del *backend*.

Una vez los datos se encuentran en el *frontend*, se procede a la visualización de los mismos. Para la visualización también se deben realizar dos archivos, el primero de ellos el HTML que estructura y lista todos los elementos, y en segundo lugar, un JavaScript encargado de la actualización y animación de los datos.

El HTML hará uso de un framework de visualización de datos, el cual se populariza por su fácil uso y rápida respuesta. Bootstrap es una herramienta de visualización y organización de páginas web[65]. En este proyecto se utiliza principalmente para la organización de la página web y la visualización de las barras de progreso del pedal de acelerador y freno, así como el porcentaje restante de batería del vehículo.

Una vez creados los elementos y la organización de la visualización, se procede al desarrollo del JavaScript encargado de la gestión de los elementos que figuran en el HTML previamente descrito.

Para la gestión de la actualización de tiempo real, primero deberemos conectar el *frontend* con el *backend* haciendo uso del puerto de intercambio abierto en el *backend*. La conexión se realiza mediante la instrucción: `“var socket = io.connect('localhost:3000');”`. La variable `socket`, siempre que llegue una nueva actualización o dato proveniente del *backend*, actualizará todo el *frontend* con los valores del objeto `carInfo`.

Finalmente, deberemos especificar como se inicia la RaspberryPi y la ejecución de los dos scripts a la vez mientras que a su vez se inicia el entorno gráfico.

Para llevar a cabo la secuencia de arranque de la RaspberryPi, simplemente deberemos ejecutar un archivo `“.sh”` el cual se ejecutará siempre que se inicie el ordenador. Esto se debe a que dicho archivo ha sido configurado en el `boot.config`.

5.3.4 IMPLEMENTACIÓN SIMULADA DE LA RECEPCIÓN

Para la creación del módulo de recepción de los datos, debemos generar una salida del entorno simulado similar o igual a la salida esperada de la recepción de los datos por medio de un módulo LoRa. Para ello recreamos la salida del formato de los datos del Arduino y comenzamos la escritura en el puerto del ordenador o servidor receptor.

En un primer momento, al igual que el ordenador de a bordo, se realizó el script en Python para aportar uniformidad al código de todo el proyecto pero por motivos de fluidez en la visualización [60] y ya habiendo modificado el lenguaje en el ordenador de a bordo se cambia a JavaScript también.

Las especificaciones o requisitos que debe cumplir el ordenador receptor son dos, la visualización de los datos en tiempo real de manera aumentada y más completa que la

generada dentro del vehículo y el almacenamiento optimizado de los datos, proporcionando un sistema de almacenado óptimo y su fácil acceso desde internet o intranet (en caso de haberse hecho solo para red local).

El script desarrollado en Python en un primer momento realizaba las tareas requeridas, pero contaba con grandes limitaciones a la hora de la visualización en tiempo real ya que el procesado de los datos no era lo suficientemente veloz. Por ello, se decide la migración a JavaScript. Hay que destacar que la visualización en Python al igual que en el caso del ordenador de a bordo, se había realizado a través las librerías de Dash, propiedad de Plotly.

Una vez decidido el cambio, deben realizarse dos modificaciones, la creación de nuevo del script de visualización de los datos en tiempo real y la creación de la base de datos al mismo tiempo que un gestor para la misma para poder hacer los accesos de manera independiente.

Para comenzar, el script de la visualización de los datos reutiliza mucha de la estructura explicada e implementada en el apartado anterior, aunque debe modificarse la lectura de los datos por parte del ordenador, ya que la lectura en este caso, no se hace por medio del BUS Can, si no por la lectura de un puerto serial del ordenador receptor. Esta modificación es relativamente sencilla ya que solamente se sustituye el módulo Node encargado de dicha función y se utiliza: *serialport*. Este módulo permite, especificando el número de puerto (en este caso *COM9*) y la velocidad a la que debe operar su lectura (*115200 baudios*) el acceso a los datos por parte de JavaScript, al igual que en el caso del ordenador *on-board*.

Una vez se posibilita el acceso a los datos, se realiza la misma conexión *backend* y *frontend* realizada previamente, aunque el intercambio en este caso no es solamente de los datos más críticos del vehículo si no que se intercambian todos los datos posibles con el mismo y ya en el *frontend* se decide cuáles de todos los disponibles se muestran en tiempo real, pudiendo hacerse modificaciones en el mismo momento, ampliando el número de graficas o la visualización de las mismas.

Para la visualización de los datos, tras haberse probado diferentes *frameworks* y librerías de JavaScript, finalmente se hace uso de las librerías *d3 Charts* ya que son este caso las que mejor respuesta y con mayor velocidad además de gran adaptabilidad proporcionan.

El principal reto del visionado y modificación de gráficas en tiempo real es la velocidad de renderización y el hacerlo varias veces por segundo. Mientras que existen infinidad de librerías que permiten la generación de estas gráficas, pocas de ellas permiten su uso adaptado a un análisis en tiempo real. Como se ha visto en el apartado de diseño, hay librerías cuyo uso para un análisis en tiempo real cuestan sus licencias más de mil euros.

Por lo tanto, a través de este proyecto, y siguiendo la especificación principal del mismo, la realización de un sistema de telemetría completo y bajo coste, no se puede hacer uso de este tipo de productos. Es en este momento y tras un estudio sobre las posibles soluciones donde se opta por la realización del visionado de las gráficas en tiempo real con las librerías de *d3* [66].

Estas librerías permiten la actualización en tiempo real, así como su renderización. Además, cuentan con respuestas al uso, es decir, si se sitúa el cursor sobre la gráfica, esta muestra información del dato que está mostrando en dicho momento, añadiendo la hora y el valor específico.

Para el renderizado de las gráficas, primero debemos especificar en que posición del documento HTML se sitúa, mediante la descripción del mismo en dicho documento. Es imprescindible asociar un id único ya que será mediante él, que se hará el acceso y el renderizado de la gráfica. Una vez situadas todas las gráficas con sus tamaños y anchuras correspondientes dentro del documento web, se procede en el JavaScript del *frontend* a la descripción de su funcionamiento y renderizado.

El renderizado de las gráficas funciona de manera similar al de los datos *on-board*. Cada vez que un nuevo dato llega al *frontend*, previamente procesado por el *backend*, se procede a la actualización de todos los valores. Esto añade la sensación de tiempo real ya que, si no ha

habido modificaciones en los valores, se vuelve a mostrar el valor pasado mientras que si sí lo ha habido, se muestra el nuevo.

Primero, se debe especificar cuantos valores queremos que esté mostrando la gráfica, por defecto lo definimos a 20, para que se muestren los últimos 20 valores de cada identificador de sensor. Si fuesen necesarios más o menos valores, es tan sencillo como modificar este valor. En segundo lugar, se crean las listas donde se almacenarán dichos valores. Cada gráfica tiene asociada una lista única de valores propios. Además, se crea el elemento CanvasJs.Chart, que crea la gráfica y define su tipo: de puntos, línea, etc...

Una vez creados los dos valores principales de la gráfica, se inicia con un primer valor a cero y se define la función de renderizado o actualización de la misma para que cuando llegue un nuevo valor, esta lo añada a la lista y cree de nuevo la imagen con los datos. Para cada valor nuevo que llega, se utiliza el comando *push* a la lista, el cual añade los elementos al final. En ese caso, si la lista está llena con los 20 elementos (la dimensión máxima) se procede a la eliminación del primer valor, recreando una estructura FIFO (first in, first out). Una vez con la lista ya adaptada al formato requerido, se procede a su renderizado.

```
var dataLength = 20; // number of dataPoints visible at any point

//Controle tempBat chart
var dpsBatTemp = []; // dataPoints
var chartBatTemp = new CanvasJS.Chart("tempBat", {
  data: [{
    type: "line",
    dataPoints: dpsBatTemp
  }]
});
var xValBatTemp = 0;
var yValBatTemp = 0;
var updateChartBatTemp = function (count) {
  count = count || 1;
  //console.log(carInfo.tbatt);
  for (var j = 0; j < count; j++) {
    yValBatTemp = parseInt(carInfo.tbatt);
    dpsBatTemp.push({
      x: xValBatTemp,
      y: yValBatTemp
    });
    xValBatTemp++;
  }
  if (dpsBatTemp.length > dataLength) {
```



```

        dpsBatTemp.shift ();
    }
    chartBatTemp.render ();
};

```

Código de gestión de una gráfica en el frontend

Además de la generación de gráficas, el ordenador receptor de los datos debe almacenarlos en una base de datos. Para ello, se implementa una base de datos controlada por *pgAdmin v4* y conectada a través de Springboot a la red.

En un primer lugar, debemos especificar como se realiza la creación de la base de datos. Primero debemos especificar en que puerto interno del ordenador o servidor queremos que se ejecute a la vez que un usuario y contraseña. Esto se debe a que cuando queramos hacer inserciones en la misma en línea, deberemos especificar el puerto de ejecución de la base así como el usuario y contraseña que hace la inserción. El puerto destinado en este caso es el 5432 y el usuario y contraseña postgres y postgres [67] respectivamente (desatacar que son usuarios y contraseñas por defecto, para mayor seguridad es recomendado modificarlo).

Una vez especificados el puerto, usuario y contraseña, debemos crear la base de datos y sus tablas asociadas. En este caso, solo se contará con una tabla que será la que procese y almacene todos los datos provenientes del vehículo. Dicha tabla, deberá tener como campos la hora de llegada del dato, valor de su identificador único de sensor y el valor actual del mismo. Además, para el correcto funcionamiento de la tabla, deberemos especificar también una clave para cada dato, por lo tanto, creamos un identificador único por dato que será el valor o número de paquete dentro de la secuencia. Es decir, el primer dato recibido, su clave será uno, el segundo su clave será dos, etc...

36	36	100	254	100
37	37	100	102	100
38	38	100	101	100
39	39	100	100	100

Tabla 2: Almacenado de los datos en la base

Una vez creada la tabla y la base de datos, se procede a la implementación de la misma mediante el uso de Springboot, para hacerla accesible desde la red, tanto para inserciones como para extraer los datos.

Springboot es un popular framework, previamente explicado más detenidamente en el apartado de descripción de las tecnologías, que se utiliza en el marco de trabajo empresarial de código abierto que sirve para crear aplicaciones autónomas de producción que se ejecutan en una máquina virtual Java (JVM) [68].

En este caso, lo hemos utilizado para la creación de una aplicación de gestión y extracción de datos mediante llamadas a una API. Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos [69].

En primer lugar, deberemos crear un proyecto con las dependencias necesarias para el correcto funcionamiento de Springboot y de la aplicación que queremos desarrollar. Para ello hacemos uso de Spring initializr el cual permite añadir las dependencias que deseemos y empezar el proyecto con estas ya cargadas [70]. Una vez, creado el proyecto, deberemos primero de todo, conectar el proyecto con la base de datos, especificando el puerto, usuario y contraseña previamente especificados en la descripción de la base de datos.

Ya conectada la base de datos, se procede a su gestión mediante las definiciones de llamadas y guardado de los datos. Hay que destacar que debemos definir qué sentencias deben ejecutarse y como para la inserción de los datos.

El proyecto dentro de SpringBoot se divide en los diferentes gestores de petición, implementación y conexión con la base de datos. El lenguaje utilizado en este proyecto de gestión y procesado de peticiones es Java.

Para el correcto funcionamiento encontramos diferentes partes dentro del servidor. Entre las cuales destacan: un controlador de peticiones al servidor o aplicación, una definición de objeto *valor*, que serán el tipo de objeto que se almacenará en la base de datos, un controlador

del repositorio o tabla, encargado de generar las *queries*, y un módulo de implementación donde se realiza la conexión entre el repositorio y el controlador de peticiones.

El primer elemento o clase a describir es la que se corresponde con los servicios que interactúan con la base de datos ya que primero de todo debemos saber si la conexión está bien hecha. Para ello, se hace uso de los módulos previamente cargados dentro del proyecto de Springboot que permiten la generación de una clase *ServiceImplementation*. Esta clase deberá sobrescribir algunos métodos como el de actualizar valor, crear valor o extraer valor por Id. Todos estos métodos hacen referencia a la clase repository que es la encargada de ejecutar las *queries* para la base de datos.

La clase repository no necesita de grandes definiciones ya que utiliza otro modulo previamente cargado en la generación del proyecto: *extends CrudRepository*. Esto nos permite tener las llamadas a la base de datos previamente configuradas.

En el controlador de peticiones, *ValueController*, se especifica la ruta y como deben ser las peticiones a la base de datos. Por ejemplo, para ejecutar una secuencia de extracción de datos, deberemos introducir en la búsqueda, *localhost:8080/api/values*. Cuando se detecta un acceso del tipo *get* a dicha dirección, se ejecuta la función *retrieveValues* del módulo que implementa la conexión entre el repositorio y el controlador y devuelve la salida de la ejecución de la query “*SELECT * FROM VALORES*”.

La función más determinante de la base de datos es la del almacenamiento de cada valor. Para ello, se realiza la descripción de la gestión por parte del controlador de las peticiones de *POST* a la misma ya que debe insertar en valor. Cada vez que un nuevo valor llega, se le asigna un nuevo identificador único, y se ejecuta la query que inserta un nuevo valor a la base de datos.

Esta función o query, será ejecutada cada vez que se haga una petición de *POST* al servidor. Estas peticiones deberán hacerse cada vez que llegue un nuevo dato proveniente del coche. Por lo tanto, el encargado de generar estas peticiones será el JavaScript del *backend*, que además de intercambiar datos con el *frontend*, deberá mandar los mismos a la base de datos.

Esta solución planteada se realiza dentro del conocimiento de la necesidad de la base de datos offline o local. Si se necesitase una base de datos de acceso online, la modificación correspondiente sería habilitar dentro de las funciones de red del router del servidor, la gestión interna de peticiones por parte de solicitudes externas a la red propia.

Una vez habilitada la gestión de peticiones, deberá conectarse la aplicación con un dominio web propiedad del equipo o particular. Esto debe realizarse a través de la propagación del nombre o *url* y de la *ip* asociada que tiene mediante el uso de *DNSs*.

Ya con la gestión de red terminada, solamente deberemos modificar dentro del JavaScript que las peticiones de *POST* también llamados *fetch* en la jerga de JavaScript, no es a un puerto interno con una url específica si no a un dominio público.

En este último caso, resultaría conveniente añadir alguna capa de seguridad para no comprometer los datos y su lectura por parte de equipos rivales o competencia, así como por integridad del propio sistema del servidor y su gestión de peticiones. Cada petición de almacenado de dato deberá en este caso, llevar un usuario y contraseña asociado y la base de datos además de almacenar los valores propios del sistema, deberá como valor añadido registrar que usuario hace el acceso y cuando.

Esto se debe a que cuando se publica un sitio web propio, siempre es susceptible de poder ser atacado o comprometido por usuarios externos a la red interna.

Finalmente, destacar, que no se ha podido realizar la implementación del módulo de mantenimiento predictivo ya que al no haberse podido realizar la implementación real del sistema de telemetría, no se han podido obtener los datos suficientes para una simulación. En el capítulo de trabajos futuros se especifica la importancia de este módulo y de su implementación.

Capítulo 6. ANÁLISIS DE RESULTADOS

A continuación, se muestran los resultados obtenidos más relevantes del desarrollo del proyecto, que confirman la capacidad del sistema y sus posibilidades de contribuir a todos los equipos de la competición. Se aporta prevención de errores, así como la visualización, transmisión y almacenamiento de los valores vitales del vehículo.

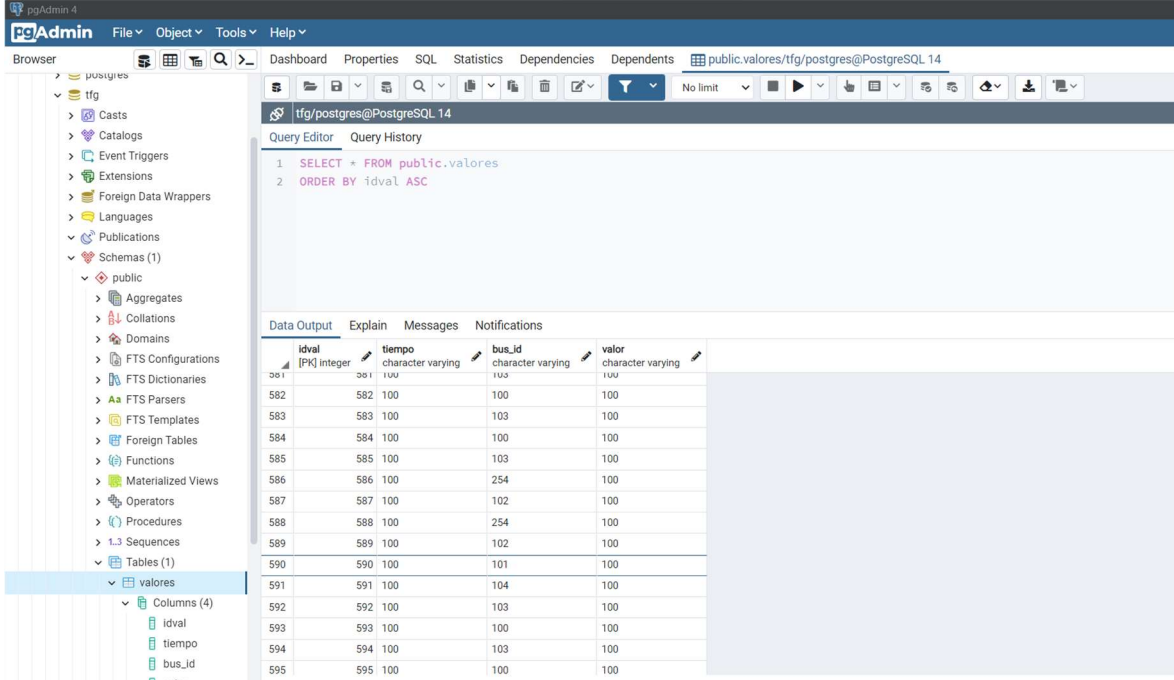
6.1 CORRECTO FUNCIONAMIENTO DE PETICIONES Y ALMACENAMIENTO INSTANTÁNEO DE LOS VALORES DEL VEHÍCULO

Uno de los pilares fundamentales del proyecto es la capacidad de almacenamiento instantáneo o pseudo-instantaneo para cada valor que llega al servidor receptor de los datos provenientes del vehículo.

Para ello, se implementa en paralelo la visualización en tiempo real de los datos, así como su transmisión a la base donde se almacenan. El principal reto es el correcto funcionamiento de ambos, proporcionando máxima velocidad a ambos procesos. Por lo tanto, y como se muestra en las imágenes, debemos comprender como una solución para dicho reto es el empleo de funciones asíncronas.

La asincronía de JavaScript permite realizar largas solicitudes o gran cantidad de ellas sin bloquear el hilo principal. Esto se debe a que dicho lenguaje fue diseñado en su origen para ser ejecutado en navegadores, trabajar con peticiones sobre la red y procesar las interacciones de usuario, al tiempo que mantiene una interfaz fluida [71].

En las siguientes capturas, se muestra como las peticiones llegan al servidor de manera continua, figura 29, manteniendo un flujo aparentemente sin interrupciones ni saltos asíncronos entre llamadas, pero una vez realizada la visualización de la escritura de los datos en la base de datos, figura 28, se aprecia el valor y la utilidad de las funciones asíncronas.



Query Editor Query History

```
1 SELECT * FROM public.valores
2 ORDER BY idval ASC
```

idval	tiempo	bus_id	valor
581	100	100	100
582	100	100	100
583	100	103	100
584	100	100	100
585	100	103	100
586	100	254	100
587	100	102	100
588	100	254	100
589	100	102	100
590	100	101	100
591	100	104	100
592	100	103	100
593	100	100	100
594	100	103	100
595	100	100	100

Figura 27: Almacenamiento instantáneo en la base de datos db2 controlada por pgAdmin

```
performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2023-06-12 11:40:47.799 INFO 5428 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-06-12 11:40:47.810 INFO 5428 --- [main] com.example.DemoApplication : Started DemoApplication in 5.479 seconds (JVM running for 6.007)
2023-06-12 11:45:23.488 INFO 5428 --- [nio-8080-exec-1] o.a.c.c.c.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-06-12 11:45:23.489 INFO 5428 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2023-06-12 11:45:23.490 INFO 5428 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2023-06-12 11:45:23.868 DEBUG 5428 --- [nio-8080-exec-1] o.s.jdbc.core.JdbcTemplate : Executing SQL statement [INSERT INTO valores (IDVAL, tiempo, BUS_ID, VALOR) VALUES ('1', '100', '0', '100');]
2023-06-12 11:45:23.868 DEBUG 5428 --- [nio-8080-exec-4] o.s.jdbc.core.JdbcTemplate : Executing SQL statement [INSERT INTO valores (IDVAL, tiempo, BUS_ID, VALOR) VALUES ('4', '100', '0', '100');]
2023-06-12 11:45:23.868 DEBUG 5428 --- [nio-8080-exec-5] o.s.jdbc.core.JdbcTemplate : Executing SQL statement [INSERT INTO valores (IDVAL, tiempo, BUS_ID, VALOR) VALUES ('5', '100', '0', '100');]
2023-06-12 11:45:23.868 DEBUG 5428 --- [nio-8080-exec-2] o.s.jdbc.core.JdbcTemplate : Executing SQL statement [INSERT INTO valores (IDVAL, tiempo, BUS_ID, VALOR) VALUES ('3', '100', '0', '100');]
2023-06-12 11:45:23.868 DEBUG 5428 --- [nio-8080-exec-3] o.s.jdbc.core.JdbcTemplate : Executing SQL statement [INSERT INTO valores (IDVAL, tiempo, BUS_ID, VALOR) VALUES ('2', '100', '0', '100');]
2023-06-12 11:45:23.963 DEBUG 5428 --- [nio-8080-exec-6] o.s.jdbc.core.JdbcTemplate : Executing SQL statement [INSERT INTO valores (IDVAL, tiempo, BUS_ID, VALOR) VALUES ('6', '100', '0', '100');]
```

Figura 28: Correcta conexión entre base de datos y frontend

6.2 VISIONADO EN TIEMPO REAL DE LOS DATOS Y RENDERIZADO DE LAS GRÁFICAS

Otro pilar indispensable del proyecto es el correcto visionado de los datos generados por el vehículo, así como el correcto renderizado de las gráficas, mostrando y actualizando sus valores en tiempo real.

Se ha hecho uso de frameworks como Bootstrap, el cual como podemos ver en las visualizaciones adjuntadas a continuación, nos sirven como representación perfecta, su barra de progreso, de la presión sobre los pedales de freno y aceleración, así como el porcentaje de batería restante del vehículo. El resto de valores mostrados son simples elementos de texto de página HTML, los cuales se actualizan y no requieren de renderizado. Este visionado de datos, se puede ver reflejado en la figura 30.

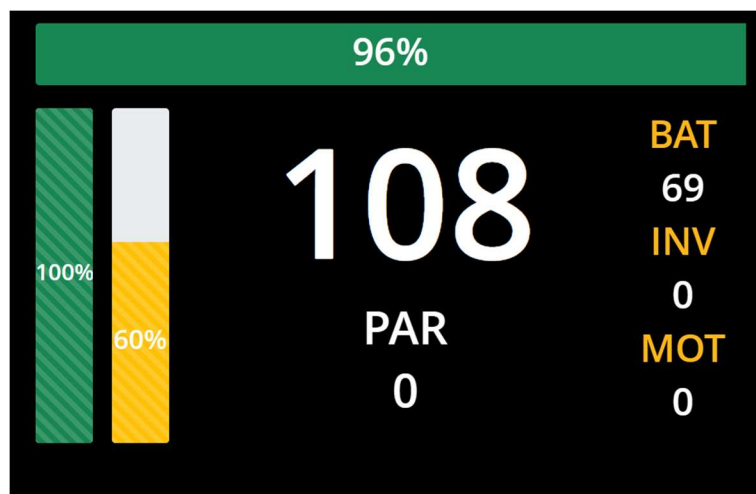


Figura 29: Valores modificados en tiempo real

En segundo lugar, encontramos la visualización de las gráficas, figura 31, valor fundamental del proyecto. Como podemos comprobar, la visualización es completa y detallada en todos los casos.

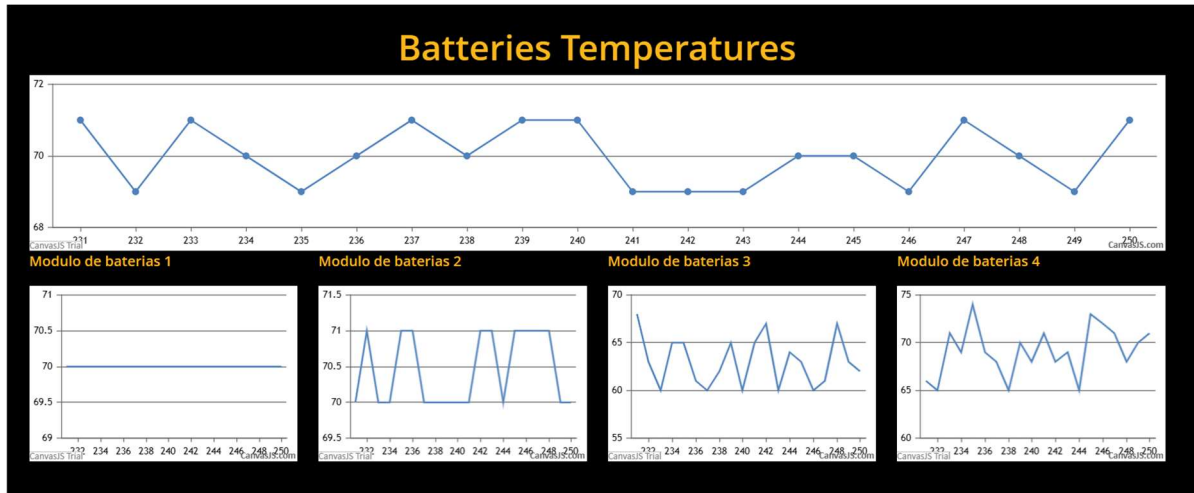
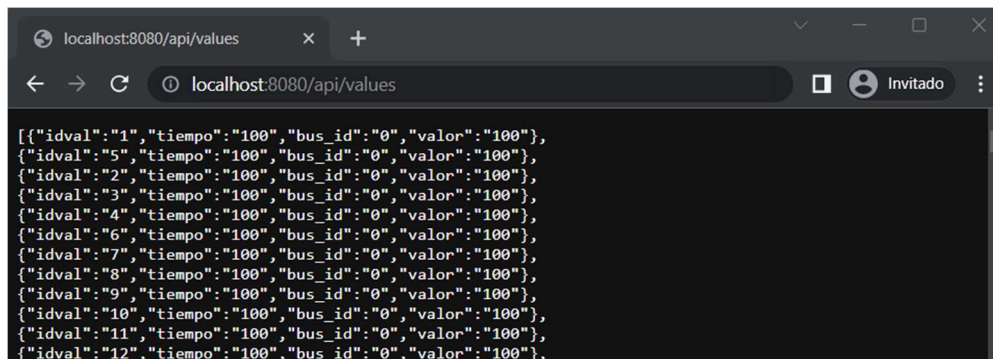


Figura 30: Generación de graficas en tiempo real

6.3 ACCESO DE LOS DATOS DESDE LA RED INTERNA DEL SERVIDOR RECEPTOR DE DATOS POR PARTE DE CUALQUIER USUARIO

Finalmente, otro valor fundamental es el acceso a los datos por parte de los ingenieros o responsables del vehículo para poder utilizar herramientas propias de confirmación de correcto funcionamiento.

A continuación, se adjunta la visualización, figura 32, de los datos mediante un acceso externo al ordenador o servidor, pero desde dentro de la propia red local.



```
localhost:8080/api/valores
localhost:8080/api/valores
Invitado
[{"idval": "1", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "5", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "2", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "3", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "4", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "6", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "7", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "8", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "9", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "10", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "11", "tiempo": "100", "bus_id": "0", "valor": "100"},
{"idval": "12", "tiempo": "100", "bus_id": "0", "valor": "100"}]
```

Figura 31: Acceso remoto a través de la red local o Internet

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

El sistema de telemetría diseñado cumple con todas las especificaciones previamente planteadas en el anexo B y las necesarias para el equipo de formula Student. Las principales especificaciones y diferencias frente al resto de soluciones en el mercado o en otros equipos han sido desarrolladas aportando un valor real al modelo.

Con el proyecto, se pretende aportar una herramienta de análisis y seguimiento en tiempo real de los valores del monoplaça y su visionado de manera intuitiva y rápida, resultando útil y eficaz para aquellos ingenieros que decidan cambios de última hora o modificaciones importantes una vez visto el desempeño del vehículo.

Los objetivos principales del proyecto son la transmisión de los datos en tiempo real, su almacenamiento, su visionado en tiempo real y el uso de gráficas, valores o herramientas visuales las cuales proporcionen el máximo número de información en el menor tiempo y espacio posible. Además de cumplir con todas estas especificaciones y estos objetivos, se han implementado nuevas mejoras y modificaciones los cuales permiten un acceso inmediato a la base de datos de manera remota por parte de cualquier persona conectada a la red (caso de base de datos en local) o de cualquier persona que esté conectada a internet (caso de base de datos a través de web).

También, se dota al sistema de otro objetivo añadido más tarde como una introducción de un modelo de *predictive maintenance*, o mantenimiento predictivo, el cual por falta de datos y de un sistema real no ha podido llegar a finalizarse.

No obstante, y como ya se ha mencionado en diversos capítulos de la memoria, ha habido limitaciones y cambios en el desarrollo. A continuación, se listan las principales limitaciones y futuros trabajos que pueden mejorar el desempeño del sistema.

7.1 LIMITACIONES EN LA IMPLEMENTACIÓN

El proyecto, como se ha especificado en el apartado de implementación no ha podido integrarse en el monoplaza por cuestiones ajenas al sistema por lo que todas las conclusiones y resultados han sido obtenidos en un entorno simulado. Cabe destacar que la simulación, por cuestiones prácticas ha sido realizada desde la recepción de los datos. Por lo tanto, han sido simulados los valores en la recepción y como sus respectivos id o identificadores de sensor.

Todas estas simulaciones y patrones de comportamiento contemplados en el entorno recreado han seguido en un inicio valores aleatorios y conforme el desarrollo tomaba envergadura, se han aproximado los valores ficticios a valores reales medidos en competiciones previas.

No obstante, toda la simulación supone una limitación a la hora de ser implementado en un monoplaza real ya que deben revisarse todos los valores que se van a recibir y ajustar sus identificadores para que a la hora de ser almacenados como mostrados y procesados en tiempo real sean los datos correctos.

7.1.1 FORMATO REAL DEL DATO

Además, una limitación extra debida a la falta de implementación es la falta de realidad en la sensorización y de los valores. Dependiendo de los formatos y necesidad de precisión en el dato que extraemos de la sensorización deberemos ajustar nuestro sistema propuesto para poder trabajar con el grado de detalle y precisión que requieren este tipo de competiciones.

7.1.2 FALTA DE SENSORIZACIÓN REAL

Otro punto de conflicto o limitación del proceso es al no contar con los sensores y diferentes sistemas o electrónicas generadoras de datos y valores, no podemos obtener los datos definitivos. Es decir, gran parte de los sensores electrónicos y sistemas internos que forman

parte de la comunicación BUS Can, intercambian datos sin formatear o procesar, es decir, intercambian valores medidos en resistencias o impedancias. Esto implica que los valores intercambiados son de voltajes u otros indicativos de medición. En la telemetría definitiva por lo tanto se deberán aplicar los coeficientes de transformación o conversión al valor natural para ser transmitidos o recibidos ya en un formato adecuado para su lectura. Todos o gran parte de los sensores, cuentan en su *datasheet*, con los factores de conversión necesarios para su correcta lectura. Una vez organizados todos los identificadores de los sensores (cabe destacar que todos los identificadores son únicos para cada sensor) se relacionan con su factor de conversión. Esta relación y transformación del dato, para no cargar con computaciones innecesarias al limitado equipo electrónico dentro del monoplaza, es recomendable, que se realice una vez ha sido realizada la transmisión de los datos sin convertir.

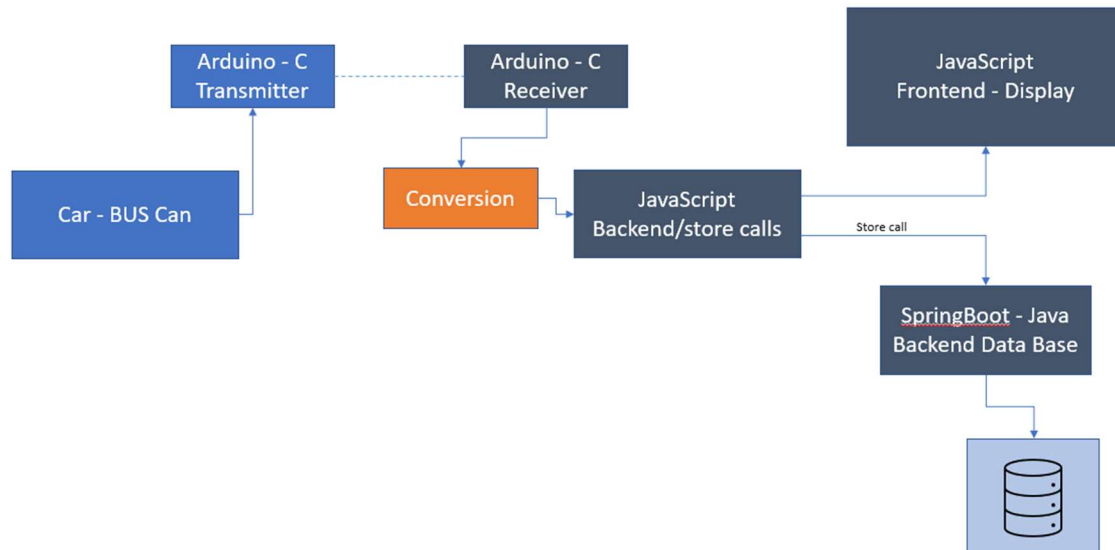


Figura 32: Sistema propuesto

7.1.3 IDENTIFICACIÓN DE LOS SENSORES

Para el correcto desempeño del sistema de telemetría planteado, requiere que todos los sensores estén monitorizados y controlados mediante sus identificadores únicos dentro de la

red. Una limitación dentro de la implementación en la vida real es la disparidad y diferentes criterios que se utilizan en los diferentes equipos. Una vez implementado el sistema, deberá dotársele de la información de los identificadores de cada sensor y su descripción.

7.1.4 TASAS DE ACTUALIZACIÓN DE LOS DATOS

Finalmente, la última limitación de la no aplicación en la vida real del proyecto es la falta de conocimiento de las velocidades de refresco o actualización del dato de los diferentes sensores. En la simulación, siempre se ha supuesto el caso más extremo, tener que actualizar todos los datos a una frecuencia muy elevada. Si se hace este ajuste, deberán determinarse que valores son aquellos que no hace falta modificar, leer o que ni siquiera están siendo modificados por el vehículo y solo actualizarlos cuando lo hagan en el monoplaza.

7.1.5 LIMITACIÓN DE RANGO

Una limitación real que existe dentro de todas las comunicaciones inalámbricas son las limitaciones de rango o alcance, bandas de frecuencia y la latencia.

Las comunicaciones inalámbricas presentan varias limitaciones en términos de alcance o rango. Entre otras se incluyen:

- **Distancia:** distancia física entre los dispositivos emisor y receptor. Cuanto mayor sea la distancia, mayor será la atenuación de la señal y más débil será la intensidad de la señal recibida. Esto puede resultar en una pérdida de calidad de la comunicación e incluso en la interrupción de la señal.
- **Obstáculos físicos:** Los obstáculos físicos, como paredes, edificios, árboles y otros objetos sólidos, pueden interferir con la propagación de las ondas de radio utilizadas en las comunicaciones. Estos obstáculos pueden bloquear, reflejar o difractar la señal, lo que lleva a una pérdida de intensidad de la señal y a una disminución del alcance efectivo.
- **Interferencia:** estas pueden sufrir interferencias de otras fuentes de señales electromagnéticas, como dispositivos electrónicos cercanos, electrodomésticos, sistemas de comunicación de radio o televisión, e incluso otros dispositivos

inalámbricos. Esta interferencia puede degradar la calidad de la señal y afectar el alcance de la comunicación.

En base a las bandas de frecuencia utilizadas, las principales limitaciones que existen son:

- **Propagación de ondas:** Las ondas electromagnéticas utilizadas en las comunicaciones inalámbricas se propagan de manera diferente en diferentes frecuencias. Por ejemplo, las ondas de alta frecuencia, como las microondas y las ondas milimétricas, tienen dificultades para atravesar obstáculos físicos y tienden a propagarse en línea recta. Por otro lado, las ondas de baja frecuencia, como las ondas de radio, pueden propagarse más lejos y tienen una mejor capacidad para atravesar obstáculos, pero su ancho de banda es más limitado.
- **Ancho de banda limitado:** El espectro electromagnético utilizado para las comunicaciones inalámbricas es limitado y se divide en diferentes bandas de frecuencia asignadas para diferentes servicios y aplicaciones. A medida que aumenta la demanda de ancho de banda para transmitir datos, el espectro disponible puede volverse congestionado, lo que lleva a una competencia por el uso de frecuencias y una disminución en el rendimiento de la comunicación.
- **Regulaciones y licencias:** Algunas bandas de frecuencia están reguladas y requieren licencias para su uso. Estas regulaciones varían según el país y pueden limitar el acceso a ciertas frecuencias para usuarios específicos o requerir un pago por el uso del espectro. Esto puede ser una limitación para los usuarios que deseen utilizar ciertas frecuencias y puede influir en la disponibilidad y el alcance de las comunicaciones inalámbricas.
- **Atenuación y pérdida de señal:** A medida que aumenta la frecuencia de las ondas electromagnéticas, estas tienden a sufrir mayor atenuación y pérdida de señal al propagarse a través del espacio y al encontrarse con obstáculos físicos. Esto puede limitar el alcance efectivo de las comunicaciones inalámbricas en frecuencias más altas y requerir una mayor densidad de estaciones base o repetidores para mantener una buena cobertura.

Finalmente, en base a la latencia, las limitaciones son:

- **Retardo de propagación:** El tiempo de propagación de la señal inalámbrica puede ser significativo, especialmente en distancias largas. La velocidad de la luz es rápida, pero en distancias considerables, el tiempo de viaje puede sumar una cantidad apreciable de latencia. Esto es especialmente notable en aplicaciones que requieren una comunicación en tiempo real, como juegos en línea, videoconferencias o transmisiones en vivo.
- **Interferencia y congestión de red:** En entornos con alta densidad de dispositivos inalámbricos o en redes inalámbricas congestionadas, puede haber interferencia y congestión en el canal de comunicación. Esto puede provocar retrasos en la entrega de paquetes de datos, aumentando la latencia general de la comunicación.
- **Retransmisiones y errores:** En las comunicaciones inalámbricas, las señales pueden sufrir interferencias o atenuaciones debido a obstáculos físicos o interferencias electromagnéticas. Esto puede llevar a la pérdida de paquetes de datos y errores de transmisión. Cuando se producen errores, se requiere una retransmisión de los datos, lo que aumenta la latencia.
- **Capacidad limitada del canal:** El ancho de banda disponible en las comunicaciones inalámbricas es finito y se comparte entre múltiples usuarios y aplicaciones. Si se excede la capacidad del canal, puede producirse congestión y aumentar la latencia. Esto es especialmente relevante en entornos donde hay una gran cantidad de usuarios intentando acceder al canal al mismo tiempo, como en áreas densamente pobladas o en eventos masivos.

7.1.6 LIMITACIÓN COMPUTACIONAL

Un sistema de telemetría puede enfrentar varias limitaciones computacionales que pueden afectar su rendimiento y funcionalidad. Algunas de las limitaciones comunes incluyen:

- **Capacidad de procesamiento:** Un sistema de telemetría necesita procesar y analizar grandes volúmenes de datos en tiempo real. La capacidad de procesamiento del sistema puede ser limitada si los recursos de hardware, como la velocidad de la CPU

o la memoria, no son suficientes para manejar la carga de trabajo. Esto puede resultar en retrasos en el procesamiento de los datos y una disminución en la capacidad de respuesta del sistema.

- **Ancho de banda limitado:** La transmisión de datos de telemetría requiere un ancho de banda adecuado para garantizar la transferencia eficiente de datos entre los dispositivos de adquisición y el sistema central. Si el ancho de banda disponible es limitado, puede haber congestión en la red y una reducción en la velocidad de transferencia de datos, lo que afecta la capacidad del sistema para recibir y procesar información en tiempo real.

Almacenamiento de datos: Los sistemas de telemetría generan una gran cantidad de datos que deben ser almacenados para su posterior análisis. La capacidad de almacenamiento puede ser limitada si no se cuenta con suficiente espacio de almacenamiento o si los recursos de almacenamiento no son lo suficientemente rápidos para escribir y recuperar datos de manera eficiente. Esto puede afectar la capacidad del sistema para retener y acceder a datos históricos de telemetría.

- **Seguridad y privacidad:** Los sistemas de telemetría a menudo manejan datos sensibles y críticos, por lo que la seguridad y privacidad de esos datos son fundamentales. La implementación de medidas de seguridad adecuadas, como el cifrado de datos y el control de acceso, puede requerir recursos computacionales adicionales y, en algunos casos, puede limitar el rendimiento general del sistema.
 - **Escalabilidad:** A medida que un sistema de telemetría crece en términos de cantidad de dispositivos de adquisición y datos generados, puede enfrentar desafíos de escalabilidad. La capacidad del sistema para manejar eficientemente un aumento en la cantidad de datos y dispositivos conectados puede verse limitada si no se diseñó teniendo en cuenta la escalabilidad desde el principio.
- Integridad de los datos:** Los sistemas de telemetría deben garantizar la integridad de los datos recibidos y transmitidos. Esto implica la validación de los datos, la detección y corrección de errores, y la gestión de datos perdidos o corruptos. Estos procesos computacionales pueden requerir recursos adicionales y afectar el rendimiento del sistema.

Es importante considerar estas limitaciones computacionales al diseñar e implementar un sistema de telemetría, para asegurarse de que los recursos computacionales sean suficientes para manejar la carga de trabajo esperada y cumplir con los requisitos de rendimiento y funcionalidad del sistema.

7.2 TRABAJOS FUTUROS

La descripción del proyecto abre y plantea nuevas líneas de mejora del sistema planteado para poder maximizar el rendimiento tanto del vehículo como del propio sistema de telemetría. Entre todas las posibles líneas de mejora o futuros trabajos, destacan la creación de una interfaz de usuario para garantizar un acceso seguro y cifrado a la base de datos, la integración en un monoplaza, la integración con la industria actual y la creación del módulo descrito previamente de *predictive maintenance* para las baterías y posteriormente para el resto de los elementos que puedan aplicarlos.

7.2.1 CREACIÓN DE INTERFAZ DE USUARIO PARA ACCESO SEGURO Y CIFRADO EN LA BASE DE DATOS

La creación de una interfaz propia a un usuario, mediante acceso de usuario y contraseña, facilitaría los accesos mediante internet y acceder de manera descentralizada. El sistema aquí propuesto, solo contempla el acceso a los datos de manera local o mediante el uso de VPN o redes virtuales.

Con una interfaz de usuario, los ingenieros y resto de personas que quieran y puedan acceder a los datos, podrán hacerlo de manera segura y cifrada desde la interfaz alojada en internet.

7.2.2 INTEGRACIÓN EN UN MONOPLAZA

Como se ha mostrado previamente, el sistema planteado no se ha podido integrar en un monoplaza real. Esto supone que un trabajo futuro o una línea de investigación futura son los pasos a realizar para poder adaptar e integrar completamente el sistema en un vehículo.

Si bien es cierto, que con la descripción de limitaciones por la no integración ya tenemos los puntos que debemos modificar y cumplir, también se deberá hacer un futuro estudio de la integración del sistema del monoplaça. Desde su viabilidad dentro del monoplaça hasta su consumo eléctrico deberá ser estudiado caso por caso por cada equipo.

7.2.3 INTEGRACIÓN EN LA INDUSTRIA

Muchos de los vehículos y fórmulas de la competición no cuentan con sistemas de telemetría completos, aunque puede que si cuentan con *data loggers*, encargados de almacenar todas las comunicaciones o incluso con una visualización local dentro del monoplaça para algunos de los valores.

Como trabajo futuro o nueva línea de investigación, resultaría de gran interés la integración de este proyecto con estructuras o herramientas ya existentes y sin resultar invasivas para sus usuarios.

La adaptabilidad del sistema es muy alta, pero requiere de una comprensión completa del mismo. Esta nueva línea debería proporcionar al usuario, sin conocer al completo el sistema, la posibilidad de conseguir un sistema completo de telemetría con pequeños cambios en su monoplaça.

7.2.4 CREACIÓN DEL MÓDULO DE PREDICTIVE MAINTENANCE A TRAVÉS DE VALORES REALES

Finalmente, y como ya se ha mencionado durante todo el proyecto, la principal línea de mejora o investigación, resultando muy novedosa en el sector es la creación de módulos de *predictive maintenance*. Estos módulos serán capaces de predecir y avisar de mantenimientos necesarios con el fin de evitar pérdidas económicas o degradación de los materiales.

Para poder llevarse a cabo un módulo de mantenimiento predictivo debe haber una combinación entre hardware, software y técnicas analíticas.

En primer lugar, se requieren sensores que puedan recopilar datos sobre el estado y rendimiento del vehículo. Estos sensores pueden incluir dispositivos para medir vibración, temperatura, presión, corriente y otros parámetros relevantes. La función principal de estos sensores es capturar información en tiempo real sobre diversas partes del vehículo y su funcionamiento.

Además de los sensores, es necesario contar con una unidad de control que pueda procesar y analizar los datos recopilados. Esta unidad de control puede estar basada en microcontroladores o microprocesadores especializados en el manejo de datos y algoritmos de análisis. Su papel es fundamental para interpretar los datos de los sensores y llevar a cabo análisis posteriores.

La conectividad es otro elemento esencial en el desarrollo de un módulo de mantenimiento predictivo. Es necesario contar con una conexión de red que permita la transferencia de datos en tiempo real desde el vehículo a un sistema centralizado. Esto puede incluir tecnologías como Wi-Fi, Bluetooth o incluso una conexión a Internet móvil. La conectividad permite la transmisión continua de datos y facilita la comunicación entre el vehículo y el sistema de monitoreo y análisis.

Una vez que los datos son recopilados y transmitidos, se utilizan algoritmos de análisis avanzados y técnicas de aprendizaje automático para procesarlos. Estos algoritmos ayudan a identificar patrones, tendencias y anomalías en los datos capturados. A través del análisis de estos patrones, es posible predecir fallas o problemas futuros en el vehículo y tomar medidas preventivas de mantenimiento.

Los modelos de predicción son una parte integral del módulo de mantenimiento predictivo. Estos modelos se basan en los datos analizados y en los resultados de los algoritmos de aprendizaje automático. Utilizando técnicas estadísticas y matemáticas, los modelos pueden predecir el tiempo de vida útil de los componentes del vehículo, lo que permite planificar el mantenimiento de manera más eficiente y evitar averías inesperadas.

Finalmente, es necesario contar con un software de gestión que procese y presente la información relacionada con el mantenimiento predictivo. Este software puede mostrar alertas de mantenimiento, recomendaciones basadas en los resultados del análisis y generar informes detallados. Además, el módulo de mantenimiento predictivo debe integrarse con el sistema electrónico del vehículo para acceder a la información relevante, como el kilometraje, el historial de mantenimiento y otros parámetros de rendimiento necesarios para una evaluación completa.

Un módulo de mantenimiento predictivo en un vehículo requiere una combinación de sensores, unidad de control, conectividad, algoritmos de análisis, modelos de predicción, software de gestión e integración con el sistema electrónico del vehículo. Estos componentes trabajan en conjunto para recopilar datos, analizarlos, predecir fallos y generar recomendaciones de mantenimiento con el objetivo de mejorar la eficiencia y la confiabilidad del vehículo.

Capítulo 8. BIBLIOGRAFÍA

Referencias

- [1] <https://www.facebook.com/DriveSmart.ES>, «DriveSmart | La telemetría en la F1. Clave para el ingeniero, trampa para el piloto», *Website de :DriveSmart*. <https://drive-smart.com/es/blog/2015/07/22/la-telemetria-en-la-f1-clave-para-el-ingeniero-trampa-para-el-piloto/> (accedido 14 de junio de 2023).
- [2] strange_v, «Car Telemetry». 11 de abril de 2022. Accedido: 14 de junio de 2023. [En línea]. Disponible en: <https://github.com/strange-v/car-telemetry>
- [3] «Especial. Estado del coche autónomo: dónde estamos y hacia dónde vamos», *forococheelectricos*, 16 de septiembre de 2018. <https://forococheelectricos.com/2018/09/estado-del-coche-autonomo-donde-estamos-y-hacia-donde-vamos-i.html> (accedido 14 de junio de 2023).
- [4] «CAN bus: componentes, ventajas y fallas en el sistema». <https://blog.reparacion-vehiculos.es/can-bus-ventajas-fallas-en-el-sistema> (accedido 23 de junio de 2023).
- [5] «Conocer la interfaz de bus CAN | Winstar». <https://www.winstar.com.tw/es/can-bus-interface-communication> (accedido 23 de junio de 2023).
- [6] «NodeJS». <https://desarrolloweb.com/home/nodejs> (accedido 19 de junio de 2023).
- [7] «Arduino + Link2fs». <https://www.voovirtual.com/t45510-arduino-link2fs> (accedido 19 de junio de 2023).
- [8] «367725.pdf». Accedido: 19 de junio de 2023. [En línea]. Disponible en: <https://repositorio.unican.es/xmlui/bitstream/handle/10902/5011/367725.pdf?sequence=1>
- [9] J. G. Cobo, «Crean un simulador de coches con una placa de Arduino», *Hardware libre*, 27 de junio de 2016. <https://www.hwlibre.com/crean-simulador-coches-una-placa-arduino/> (accedido 19 de junio de 2023).

- [10] «¿Qué es Java Spring Boot? | IBM». <https://www.ibm.com/es-es/topics/java-spring-boot> (accedido 23 de junio de 2023).
- [11] S. Borges, «¿Qué es PostgreSQL? - Para qué sirve, Características e Instalación», *Infranetworking*, 19 de noviembre de 2019. <https://blog.infranetworking.com/servidor-postgresql/> (accedido 23 de junio de 2023).
- [12] A. Morales, «Descubre el nuevo pgAdmin 4 para trabajar con PostGIS», *MappingGIS*, 15 de noviembre de 2017. <https://mappinggis.com/2017/11/descubre-el-nuevo-pgadmin-4-para-trabajar-con-postgis/> (accedido 23 de junio de 2023).
- [13] P. R. de los Santos, «Breve historia de Internet de las cosas (IoT)», *Think Big*, 22 de septiembre de 2020. <https://empresas.blogthinkbig.com/breve-historia-de-internet-de-las-cosas-iot/> (accedido 23 de junio de 2023).
- [14] «Project Mercury - A Chronology. Part 2 (B)». <https://history.nasa.gov/SP-4001/p2b.htm> (accedido 23 de junio de 2023).
- [15] «PubMed Central Full Text PDF». Accedido: 23 de junio de 2023. [En línea]. Disponible en: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6235649/pdf/cureus-0010-00000003300.pdf>
- [16] «What was the earliest invention of race car telemetry actually used in a race and shown on International television?», *Quora*. <https://www.quora.com/What-was-the-earliest-invention-of-race-car-telemetry-actually-used-in-a-race-and-shown-on-International-television> (accedido 23 de junio de 2023).
- [17] MIE, «Then and Now – Telemetry in F1», *The Parc Fermé*, 23 de abril de 2014. <https://theparcferme.com/then-and-now-telemetry-in-f1/> (accedido 23 de junio de 2023).
- [18] «First use of a computer in an F1 car? - The Nostalgia Forum», *The Autosport Forums*. <https://forums.autosport.com/topic/83815-first-use-of-a-computer-in-an-f1-car/> (accedido 25 de junio de 2023).
- [19] «PFC_ABRAHAM_DIAZ_CAMPO_PINILLA.pdf». Accedido: 25 de junio de 2023. [En línea]. Disponible en: https://oa.upm.es/48663/1/PFC_ABRAHAM_DIAZ_CAMPO_PINILLA.pdf

- [20] «What's the wireless data transfer technology used in Formula One cars?», *Quora*. <https://www.quora.com/Whats-the-wireless-data-transfer-technology-used-in-Formula-One-cars> (accedido 25 de junio de 2023).
- [21] «History of Formula One regulations», *Wikipedia*. 24 de junio de 2023. Accedido: 25 de junio de 2023. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=History_of_Formula_One_regulations&oldid=1161694260
- [22] «[F1] Un recorrido por las prohibiciones de la FIA (y VI)». <https://es.charla.motor.narkive.com/Wz9PwWeS/fl-un-recorrido-por-las-prohibiciones-de-la-fia-y-vi> (accedido 25 de junio de 2023).
- [23] «Formula 1 ECU - McLaren Applied». <https://mclarenapplied.com/solutions/formula-1-ecu> (accedido 25 de junio de 2023).
- [24] «What was the earliest invention of race car telemetry actually used in a race and shown on International television?», *Quora*. <https://www.quora.com/What-was-the-earliest-invention-of-race-car-telemetry-actually-used-in-a-race-and-shown-on-International-television> (accedido 23 de junio de 2023).
- [25] G. Lara, «CAN Bus: la forma de transmitir información en el automóvil», *Motorpasion*, 24 de enero de 2013. <https://www.motorpasion.com/coches-hibridos-alternativos/can-bus-como-gestionar-toda-la-electronica-del-automovil> (accedido 25 de junio de 2023).
- [26] «qué es canbus can bus bus can». https://www.equipotaller.es/es/blog/post/29_can-bus-que-es-como-funciona-y-cuales-son-sus-ventajas.html (accedido 25 de junio de 2023).
- [27] «ER-Soft - Noticias». <http://www.er-soft.com/es/noticias/ventajas-y-usos-del-bus-can-controller-area-network> (accedido 25 de junio de 2023).
- [28] «A Few of the Unique Sensors used in Formula 1 – Formula Bharat». <https://www.formulabharat.com/blog/a-few-of-the-unique-sensors-used-in-formula-1/> (accedido 25 de junio de 2023).

- [29] «Critical Electronic Components in Formula 1 Race Cars», *Arrow.com*.
<https://www.arrow.com/en/research-and-events/articles/critical-electronic-components-in-a-formula-1-race-cars> (accedido 25 de junio de 2023).
- [30] «Formula 1: How sensor technology is changing the race | ZDNET».
<https://www.zdnet.com/article/formula-1-how-sensor-technology-is-changing-the-race/>
(accedido 25 de junio de 2023).
- [31] «LoRa», *Wikipedia*. 22 de mayo de 2023. Accedido: 25 de junio de 2023. [En línea].
Disponible en: <https://en.wikipedia.org/w/index.php?title=LoRa&oldid=1156430910>
- [32] «LoRa. Tecnología LoRa: características y ventajas», *Redexia*.
<https://www.redexia.com/red-lora/> (accedido 25 de junio de 2023).
- [33] N. Carpentiers, «F1 telemetry: The data race», *F1i.com*, 27 de septiembre de 2016.
<https://f1i.com/magazine/73067-f1-telemetry-data-race.html> (accedido 26 de junio de 2023).
- [34] «Porsche's 1990 IndyCar Prints its Telemetry on a Piece of Receipt Paper», *Road & Track*, 3 de octubre de 2018.
<https://www.roadandtrack.com/motorsports/a23581225/march-porsche-1990-indycar-teo-fabi-patrick-long/> (accedido 26 de junio de 2023).
- [35] «Realtime Telemetry Data Visualization in Formula One», *SciChart*.
<https://www.scichart.com/blog/realtime-telemetry-datavisualisation-formulaone-motorsport/> (accedido 26 de junio de 2023).
- [36] «Charting toolkit for motorsports - Lightning-fast charts for high-performance», 13 de enero de 2021. <https://lightningchart.com/charts-motorsports/> (accedido 26 de junio de 2023).
- [37] «Real Time JavaScript Chart with Chart.js», *CodePen*.
<https://codepen.io/ztrayner/details/VeJMRL> (accedido 26 de junio de 2023).
- [38] «Podium – Autosport Labs», 18 de marzo de 2023.
<https://www.autosportlabs.com/category/podium/> (accedido 26 de junio de 2023).
- [39] «PodiumConnect MK2 – Autosport Labs».
<https://www.autosportlabs.com/product/podiumconnect-live-stream-motorsport-real->

- time-telemetry-from-aim-motec-race-technology-or-other-data-acquisition-systems-to-podium/ (accedido 26 de junio de 2023).
- [40] «Pi Toolbox», *RaceDataSystems*. <https://www.racedatasystems.com/pages/analysis-software-cosworth-pi-research-toolbox-html> (accedido 26 de junio de 2023).
- [41] «Beginner’s Guide to Telemetry Analysis (MoTec)», *Trinacria Simracing*, 14 de julio de 2021. <https://trinacriasimracing.wordpress.com/beginners-guide-to-telemetry-analysis-motec/> (accedido 23 de junio de 2023).
- [42] «DASH2 PRO Manual V 1.2.pdf». Accedido: 23 de junio de 2023. [En línea]. Disponible en: <http://www.race-technology.com/upload/PDF%20manuals/DASH2%20PRO%20Manual%20V%201.2.pdf>
- [43] «“El incendio del coche no ha hecho más que motivarnos para seguir trabajando” | El Comercio: Diario de Asturias». <https://www.elcomercio.es/gijon/gijon-universidad-escuela-politecnica-ingenieria-incendio-coche-motivarnos-20210909001921-ntvo.html> (accedido 19 de junio de 2023).
- [44] «Bus CAN», *Wikipedia, la enciclopedia libre*. 22 de abril de 2022. Accedido: 23 de junio de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Bus_CAN&oldid=143065112
- [45] «Real time plotting with Matplotlib in Python», *CodersLegacy*. <https://coderslegacy.com/python/matplotlib-real-time-plotting-graphs/> (accedido 23 de junio de 2023).
- [46] «Visualization with Seaborn | Python Data Science Handbook». <https://jakevdp.github.io/PythonDataScienceHandbook/04.14-visualization-with-seaborn.html> (accedido 23 de junio de 2023).
- [47] «Plot real-time data», *Plotly Community Forum*, 18 de noviembre de 2019. <https://community.plotly.com/t/plot-real-time-data/31467> (accedido 23 de junio de 2023).
- [48] «Realtime Chart in d3.js – Wil Yegelwel – Machine Learning and Distributed Systems Engineer». <https://wil.yegelwel.com/d3-realtime/> (accedido 23 de junio de 2023).

- [49] «How To Build Realtime Charts With JavaScript and WebSocket – PieSocket Blog», 21 de abril de 2022. <https://www.piesocket.com/blog/building-realtime-charts-in-javascript> (accedido 23 de junio de 2023).
- [50] A. Matteson, «Real-Time Charts with Plotly.js in Angular», *Medium*, 19 de noviembre de 2021. <https://javascript.plainenglish.io/real-time-charts-with-plotly-js-in-angular-4a46c4c56eb5> (accedido 23 de junio de 2023).
- [51] alldatasheet.com, «MCP2515 Datasheet(PDF) - Microchip Technology». <https://www.alldatasheet.com/datasheet-pdf/pdf/166906/MICROCHIP/MCP2515.html> (accedido 23 de junio de 2023).
- [52] «AI techniques used to improve battery health and safety», *University of Cambridge*, 6 de abril de 2020. <https://www.cam.ac.uk/research/news/ai-techniques-used-to-improve-battery-health-and-safety> (accedido 22 de junio de 2023).
- [53] «Adopt Predictive Maintenance Systems for Battery Protection», 18 de mayo de 2022. <https://www.bacancytechnology.com/blog/predictive-maintenance-systems-for-battery-protection> (accedido 22 de junio de 2023).
- [54] J.-X. Tang, J.-H. Du, L. Yiting, y Q.-S. Jia, «Predictive Maintenance of VRLA Batteries in UPS towards Reliable Data Centers», jul. 2020.
- [55] J. Olarte, J. Ilarduya, E. Zulueta, R. Ferret, U. Fernandez-Gamiz, y J. Lopez-Guede, «A Battery Management System with EIS Monitoring of Life Expectancy for Lead-Acid Batteries», *Electronics*, vol. 10, p. 1228, may 2021, doi: 10.3390/electronics10111228.
- [56] «Arduino vs Raspberry Pi una comparativa HETPRO TUTORIALES», *HeTPro-Tutoriales*, 7 de diciembre de 2017. <https://hetpro-store.com/TUTORIALES/arduino-vs-raspberry-pi/> (accedido 14 de junio de 2023).
- [57] «Tutorial modulo CAN MCP2515 arduino», *Talos Electronics*, 24:04 de 500. <https://www.taloselectronics.com/blogs/tutoriales/modulo-can-mcp2515> (accedido 14 de junio de 2023).
- [58] Y. Fernández, «Arduino y Raspberry Pi: qué son y cuáles son sus diferencias», *Xataka*, 28 de agosto de 2018. <https://www.xataka.com/basics/arduino-raspberry-pi-que-cuales-sus-diferencias> (accedido 15 de junio de 2023).

- [59] «Live Updates | Dash for Python Documentation | Plotly». <https://dash.plotly.com/live-updates> (accedido 15 de junio de 2023).
- [60] «Why is JavaScript so good at visualizations over say python?», *Quora*. <https://www.quora.com/Why-is-JavaScript-so-good-at-visualizations-over-say-python> (accedido 15 de junio de 2023).
- [61] «Canbus», *npm.io*. <https://npm.io/> (accedido 15 de junio de 2023).
- [62] E. Sunday, «Data visualization with D3.js and Node.js», *LogRocket Blog*, 20 de diciembre de 2021. <https://blog.logrocket.com/data-visualization-d3-js-node-js/> (accedido 15 de junio de 2023).
- [63] «Real-Time Communication Between Frontend And Backend», *DEV Community*, 9 de enero de 2023. <https://dev.to/jszutkowski/real-time-two-way-communication-between-frontend-and-backend-using-sockets-5ghc> (accedido 15 de junio de 2023).
- [64] «socketcan», *npm*, 13 de febrero de 2023. <https://www.npmjs.com/package/socketcan> (accedido 15 de junio de 2023).
- [65] «Bootstrap: ¿qué es, para qué sirve y cómo instalarlo?», *Rock Content - ES*, 12 de abril de 2020. <https://rockcontent.com/es/blog/bootstrap/> (accedido 15 de junio de 2023).
- [66] «D3 Real Time Chart with Multiple Data Streams», *Gist*. <https://gist.github.com/boeric/6a83de20f780b42fadb9> (accedido 14 de junio de 2023).
- [67] «How to Set the Default User Password in PostgreSQL», *Chartio*. <https://chartio.com/resources/tutorials/how-to-set-the-default-user-password-in-postgresql/> (accedido 17 de junio de 2023).
- [68] «¿Qué es Java Spring Boot? | IBM». <https://www.ibm.com/es-es/topics/java-spring-boot> (accedido 17 de junio de 2023).
- [69] «¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS», *Amazon Web Services, Inc.* <https://aws.amazon.com/es/what-is/api/> (accedido 17 de junio de 2023).
- [70] «Spring Initializr», *Spring Initializr*. <https://start.spring.io> (accedido 17 de junio de 2023).

- [71] «JavaScript Asíncrono jonmircha», 15 de abril de 2020.
<https://jonmircha.com/javascript-asincrono> (accedido 19 de junio de 2023).

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

Alinear un sistema de telemetría para un equipo de Formula Student con los Objetivos de Desarrollo Sostenible (ODS) implica asegurarse de que las acciones y prácticas relacionadas con el sistema estén en consonancia con los objetivos de sostenibilidad establecidos por las Naciones Unidas.

En primer lugar, el sistema de telemetría se enmarca dentro del ODS 9, que busca promover la industria, la innovación y la infraestructura sostenibles. La telemetría es una herramienta innovadora utilizada en el campo de la ingeniería automotriz, y al desarrollar el sistema de manera eficiente y efectiva, se contribuye al avance de la industria de manera sostenible.

Además, el equipo puede considerar el ODS 11, que se enfoca en ciudades y comunidades sostenibles. Durante el desarrollo del sistema de telemetría, se pueden involucrar a la comunidad local, fomentando la colaboración y el intercambio de conocimientos. También se pueden implementar prácticas sostenibles en el diseño y la producción del sistema, minimizando su impacto ambiental.

El ODS 12, que busca la producción y el consumo responsables, también es relevante. El equipo puede enfocarse en la eficiencia energética del sistema de telemetría, optimizando el consumo de energía y minimizando el uso de materiales no renovables en su construcción. Esto contribuirá a una producción más responsable y a un consumo consciente de recursos.

En términos de acción por el clima (ODS 13), el sistema de telemetría puede recopilar datos sobre el rendimiento del vehículo, permitiendo un análisis y una optimización de su eficiencia energética. Al utilizar estos datos para mejorar el diseño y el rendimiento del automóvil, el equipo puede contribuir a la mitigación del cambio climático.

Por último, la alineación con el ODS 17, que se centra en las alianzas para lograr los objetivos, implica fomentar la colaboración y la cooperación con otros equipos, universidades, industrias y organizaciones relevantes. Compartir conocimientos y experiencias en relación con el desarrollo y la implementación de sistemas de telemetría puede contribuir al avance conjunto hacia los ODS.

ANEXO II

Todo el código fuente utilizado para la realización de este proyecto, se encuentra en el github:

<https://github.com/GonzaloBarcelo>