TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grado en Ingeniería Informática de Gestión y Sistemas de Información

OBTENCIÓN Y GESTIÓN DE DATOS DE UN MONOPLAZA

Memoria

MARIPAZ CÓRCOLES BALLESTEROS
TUTOR: LÉONARD JANER GARCÍA

2021 - 2022

TecnoCampus
Mataró-Maresme

Dedicatoria

A toda mi familia por creer en mí, en especial a mis padres por darme la oportunidad de llegar hasta aquí y a mi pareja por haber estado siempre a mi lado durante el desarrollo del proyecto.

Agradecimientos

A mi tutor del trabajo de fin de grado, Léonard Janer García por su apoyo y guía durante estos meses y al equipo de *Formula Student* EUSS Motorsport por la experiencia y conocimientos que me ha permitido adquirir.

Abstract

This final degree project shows the data acquisition system from the *Formula Student* team EUSS Motorsport. For its realization, the main elements of the car, the knowledge needs of each department and the regulations of the competition have been considered. The main goal of this study is to ease the collection and analysis of the data generated during the competitions and tests in which the team takes part thanks to a scalable data acquisition model based on a MQTT protocol and the storage of the data into a MongoDB database.

Resum

Aquest treball de fi de grau mostra el sistema per a l'obtenció de dades de l'equip de *Formula Student* EUSS Motorsport. Per a la seva realització s'ha tingut en compte els elements principals del monoplaça, les necessitats de coneixement de cada departament i la normativa de la competició. Aquest estudi té com a principal objectiu facilitar la recollida i l'anàlisi de les dades generades durant les competicions i els test en què l'equip participi gràcies a un model d'adquisició de dades escalable basat en un protocol MQTT i l'emmagatzemament de dades en una base de dades MongoBD.

Resumen

Este trabajo de fin de grado muestra el sistema para la obtención de datos del equipo de *Formula Student* EUSS Motorsport. Para su realización se ha tenido en cuenta los elementos principales del monoplaza, las necesidades de conocimiento de cada departamento y la normativa de la competición. Este estudio tiene como principal objetivo facilitar la recogida y el análisis de los datos generados durante las competiciones y los test en los que el equipo participe gracias a un modelo de adquisición de datos escalable basado en un protocolo MQTT y el almacenamiento de datos en una base de datos MongoBD.

Índice

Índice de figuras

Índice de tablas

Glosario de términos

ABS *Anti-lock Braking System*

BOTS *Brake Over-Travel System*

BSPD *Brake System Plausibility Device*

CAN *Controller Area Network*

CLT *Coolant Liquid Temperature*

CV *Combustion Vehicle*

ECU *Engine Control Unit*

EM-0X EUSS Motorsport – 0X (versión de monoplaza)

FSG *Formula Student Germany*

GPS *Global Positioning System*

I/O *Input Output*

IDE *Integrated Development Environment*

IMU *Inertial Measurement Unit*

LiDAR *Laser Imaging Detection and Ranging*

LVS *Low Voltage System*

MAT *Multiple Air Temperature*

MQTT *Message Queuing Telemetry Transport*

MSL *MegaSquirt Log*

PCB *Printed Circuit Board*

RPM Revoluciones por minuto

TPS *Throttle Position Sensor*

1. Introducción

1.1. Formula Student

La *Formula Student* es una competición automovilística a nivel mundial entre equipos universitarios de todo el mundo regulada por la *Society of Automotive Engineers* (SAE) [1] regida por estrictas normativas al puro estilo de la competición real. Cada año se celebran múltiples certámenes en circuitos oficiales donde los equipos se enfrentan entre ellos en pruebas estáticas y dinámicas, con puntuaciones otorgadas por un jurado formado por profesionales del sector.

Los orígenes se remontan al 1981, cuando el programa Formula SAE vio la luz a manos de Ron Matthews, Mike Best, Robert Edwards y John Tellkamp. En su primera edición el programa contaba con cuatro pruebas dinámicas: aceleración, manejabilidad, *endurance* (sección 1.2.2.1) y consumo. Todas ellas puntuaban de la misma manera menos *endurance*, que puntuaba el doble. Desde sus inicios, el objetivo de este programa fue darles a los estudiantes de ingeniería la oportunidad de aprender y desarrollarse mientras realizan sus estudios.

El éxito de la primera edición fue muy reducido, por lo que se decidió añadir una nueva categoría para la segunda temporada en 1982, la BajaSAE [2]. En esta nueva modalidad, a diferencia de la principal donde se utilizaban motores de cuatro tiempos, se utilizaban motores pequeños y coches tipo *buggy*[1]. En esta segunda edición se incorporaron otros cambios importantes entre los que destacan la obligatoriedad del uso de suspensiones y la redefinición de la prueba de *endurance* pasando de uno a dos pilotos.

Otro de los grandes cambios fue la creación de la prueba estática de *cost report* en 1985, esta prueba se basa en la evaluación del coste inicial de un negocio que fabricaría mil unidades del coche de competición donde el coste de cada una se limita a 4.500$.

[1] Vehículo todoterreno de chasis ligero, carrocería sin techo rígido y ruedas descubiertas.

En 1986 la competición pasó de Texas a Michigan, y en 1987 nació otra de las pruebas dinámicas que sigue realizándose hoy en día, el *Skid Pad* (sección 1.2.2.1).

En el año 1998 la competición dio el salto a escala mundial en una competición entre monoplazas diseñados y fabricados por estudiantes estadounidenses y británicos después de que en 1997 participase el primer equipo europeo. Ese mismo año la Institución de Ingenieros Mecánicos (*Institution of Mechanical Engineers* [3]) dio el visto bueno a la gestión del proyecto en Europa en asociación con SAE y la competición ha tenido lugar al final de cada año académico desde entonces bajo el nombre de *Formula Student*.

1.2. *Formula Student* en la actualidad

Actualmente los coches que participan en la competición son monoplazas de cuatro ruedas descubiertas diseñados y construidos por equipos formados por estudiantes de ingeniería, con un presupuesto máximo de 21.000€, propulsados por un motor de combustión interna con brida de admisión o eléctrico.

El diseño se basa en una normativa, muy similar a la inicial de SAE, en la que cada año se introducen pequeños cambios basados en la experiencia adquirida por los jueces y escrutiñadores de la competición anterior.

1.2.1. Inspección técnica

En esta fase se realizan las validaciones técnicas previas donde cada monoplaza tiene que pasar por una serie de escrutinios. En estos escrutinios, realizados por un equipo de jueces que trabajan en el ámbito profesional del mundo de la automoción (ingenieros de grandes marcas de coches, ingenieros de Fórmula 1, ingenieros de IDIADA [4]…), se revisa que todo cumpla con la normativa vigente y que todo funcione de forma correcta antes de que el monoplaza pueda participar en las pruebas dinámicas.

Las pruebas de escrutinio son:

- Pre-escrutinio (*pre-scrutineering*): en esta prueba un juez evalúa el equipamiento tanto del propio piloto (guantes, mono, ropa interior ignifuga, bambas y casco) como del monoplaza (juegos de ruedas disponibles) y de los elementos de seguridad (extintores reglamentarios obligatorios, funcionamiento del *inertial switch*[2] y *emergency switch*[3], arneses y manillas). Durante esta prueba también se realiza el *driver egress test*, prueba en la que el piloto tiene que demostrar ser capaz de salir del coche en menos de cinco segundos con todos los sistemas de seguridad puestos y el coche en marcha[4].

Ilustración 1 Piloto realizando el *egress test* durante el pre-escrutinio del monoplaza EM-02

[2] Actuador capaz de parar el coche de forma automática en caso de impacto.

[3] Actuador capaz de parar el coche de forma manual en caso de emergencia. El monoplaza cuenta con tres *emergency switch*, uno en el *dashboard* para el piloto y dos laterales accesibles desde fuera.

[4] Fuente de las ilustraciones: EUSS Motorsport.

- Escrutinio (*scrutineering*): prueba principal donde dos o más jueces especializados en normativa comprueban que el monoplaza sea seguro para competir. En caso de no serlo el equipo debe hacer las modificaciones pertinentes en la misma carpa de escrutinio (en caso de ser pequeñas) o en su box. En este último caso el monoplaza deberá volver a pasar por escrutinio para poder avanzar hacia la siguiente prueba.

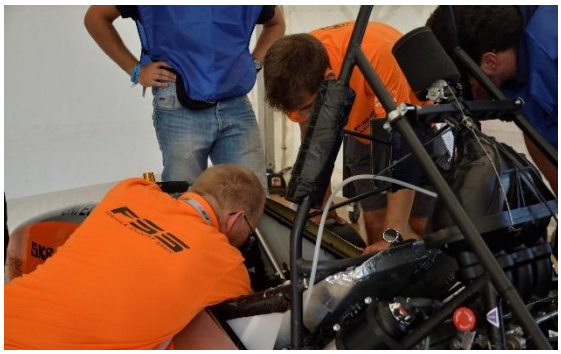


Ilustración 2 Jueces validando el fondo plano del monoplaza EM-02 durante escrutinio

- Prueba de frenada (brake test): prueba donde el monoplaza tiene que circular utilizando la segunda marcha en un circuito recto y frenar bloqueando las cuatro ruedas a la vez en la zona final delimitada por conos.

Ilustración 3 Monoplaza EM-02 a la salida del *brake test*

- Prueba de sonido e interruptor general (*noise test* y *Master Switch*): prueba donde se comprueba el nivel de ruido que emite el motor en ralentí y en corte. Para poder pasar la prueba no se pueden superar los límites de 103 y 110 dB respectivamente. Las mediciones se realizan en una zona sin obstrucciones, normalmente situada entre el

tubo de escape y la rueda. En esta prueba también se comprueba que el *Master Switch* desconecta todos los elementos y sistemas del coche[5].

Ilustración 4 Resultado del sonómetro para el monoplaza EM-03 durante el *noise test*

- Prueba de volcado (*tilt-test*): última prueba del escrutinio. Consiste en colocar el monoplaza en una rampa elevadora capaz de inclinar coche junto con el piloto más alto hasta 60º laterales para comprobar que en caso de coger una curva crítica no hay peligro de vuelque ni de pérdida de contacto de las ruedas con el suelo. Se hace una primera comprobación a 30º para asegurar que no haya ningún elemento que pierda líquido. En todo el momento tanto el monoplaza como el piloto están asegurados con una cinta que une el *main hoop*[6] con el elevador. Al final de esta prueba se pesa el coche sin piloto y repostado.

Ilustración 5 Monoplaza EM-02 en la prueba de tilt-test con una inclinación de 30º

[5] Fuente de la ilustración: Formula Student Spain.

[6] Arco principal y más alto del monoplaza.

Una vez pasado cada uno de los cinco escrutinios, se engancha en la parte frontal del monoplaza una pegatina que certifica el cumplimiento de la normativa. El motor solo puede ponerse en marcha una vez que se han conseguido las pegatinas de pre-escrutinio y escrutinio.

1.2.2. Pruebas

1.2.2.1. Pruebas dinámicas

- Aceleración (*acceleration*): prueba en solitario donde los monoplazas arrancan desde parado y recorren 75 metros en el menor tiempo posible.

- *Autocross* (*AutoX*): prueba donde los monoplazas corren en solitario en un circuito de conos enrevesado de menos de 1,5 Km de largo y tres metros de ancho. El objetivo es completar el circuito en poco tiempo tirando el menor número de conos posibles ya que estos representan una penalización en el tiempo total.

- Resistencia (*endurance and fuel consumption*): en esta prueba de 22 Km en un circuito enrevesado diferente del de *AutoX* los equipos compiten entre sí para demostrar que su monoplaza es más rápido, fiable y eficiente que el resto. Hay un cambio de piloto a mitad de la carrera, donde el coche tiene que estar parado tres minutos. La puntuación se otorga en función de si se completa o no, el tiempo marcado y el consumo de combustible.

- Circuito en ocho (*skid pad*): en esta prueba los monoplazas se enfrentan a cuatro carreras cortas en un circuito simétrico en forma de ocho. Es la única prueba donde el circuito siempre es el mismo[7].

Ilustración 6 Circuito Skid Pad

1.2.2.2. Pruebas estáticas

- Presentación (*business plan*): dos miembros del departamento de organización del equipo presentan un plan de negocio a los jueces e intentan vender el proyecto haciendo hincapié en los puntos fuertes del monoplaza. La intención es que los jueces decidan hacer una inversión inicial en el equipo.

- Análisis de diseño (*design event*): un jurado visita el box del equipo donde los coordinadores de cada departamento ponen a prueba su conocimiento técnico defendiendo el diseño de ese año. Durante 30 minutos se explica en qué se basan, qué han tenido en cuenta y qué mejoras se han aplicado respecto al monoplaza anterior.

- Análisis de coste (*cost report*): los equipos presentan una lista detallada de los precios y cantidades de todos los materiales y procesos utilizados en la construcción del monoplaza para demostrar que el proyecto se ajusta a los 21.000€ de presupuesto establecidos por la competición.

[7] Fuente de la ilustración: equipo de *Formula Student* ESTACA.

1.3. Equipo

EUSS Motorsport [5] es el equipo de *Formula Student* formado casi en su totalidad por estudiantes de la Escola Universitària Salesiana de Sarrià (EUSS) [6] fundado en marzo de 2016. En él hay como integrantes estudiantes de diferentes ingenierías: mecánica, electrónica, doble grado en ingeniería mecánica y electrónica, aeronáutica, automoción, informática y organización industrial.

La selección de integrantes del equipo pasa por diferentes fases:

- Fase inicial donde todo aquel interesado en el proyecto pasa por una entrevista, unas formaciones y unos exámenes de validación de conocimientos y aptitudes. Durante esta etapa el número de participantes se sitúa alrededor de 80 estudiantes.

- Fase intermedia donde los nuevos integrantes junto con los antiguos empiezan la fase de diseño y construcción del monoplaza. En esta etapa se siguen haciendo formaciones, de carácter general, impartidas por los coordinadores actuales y miembros *alumni*, y cuentan con pruebas evaluativas. Si un miembro del equipo falta a más de tres formaciones sin motivo justificado puede ser expulsado del equipo. Durante esta etapa el número de participantes es menor a 60.

- Fase final o precompetición. Esta es la etapa más dura para los miembros del equipo debido a que se pone a prueba todo aquello que han ido aprendiendo y desarrollando a lo largo del año, empieza con el primer test del monoplaza y acaba con la última competición. Es la etapa donde más miembros abandona debido a la presión y la carga de trabajo y la que menos miembros tiene, normalmente alrededor de veinte.

A lo largo de todas las etapas el equipo cuenta con un tutor o *faculty advisor* por parte de la universidad. Desde el nacimiento del equipo este rol lo ha representado el Dr. Ignasi Florensa Ferrando[8].

[8] https://www.researchgate.net/profile/Ignasi-Florensa-Ferrando

1.3.1. Estadísticas del equipo

A continuación, se muestran tres gráficas: una para la fase precompetición donde quedan los miembros finales de la temporada, una para la distribución de los diferentes estudios de grado, y una última para la distribución de sexos.

1.3.1.1. Fase competición

Al acabar la temporada 2020-2021 el equipo contaba con un total de 23 integrantes de tres universidades diferentes. De estos 23 integrantes, veintiuno eran estudiantes propios de la EUSS (91,30% de los miembros), uno era estudiante de la UPC de Terrassa (4,35% de los miembros) y uno era estudiante del Tecnocampus Mataró – UPF (4,35% de los miembros).

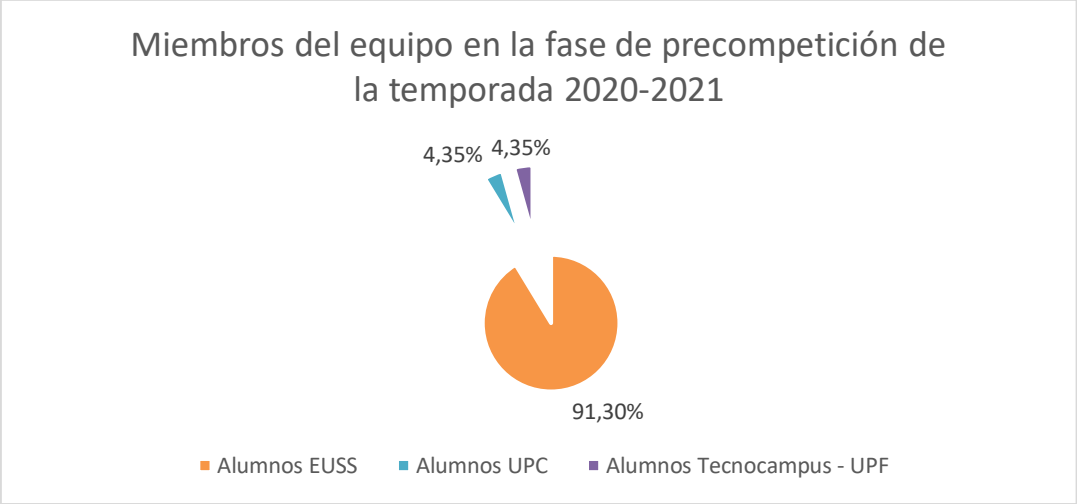


Ilustración 7 Miembros del equipo en la fase de precompetición (Temporada 2020-2021)

1.3.1.2. Distribución de estudios de grado

El equipo cuenta en su mayoría con perfiles técnicos provenientes de seis grados de ingeniería diferentes. De los 23 estudiantes, cinco son de ingeniería mecánica, cuatro de doble ingeniería mecánica y electrónica, tres de ingeniería electrónica, tres de ingeniería de automoción, uno de ingeniería aeroespacial y uno de ingeniería informática. Esto constituye un 73,91% del equipo.

El 26,09% restante proviene de los cuatro estudiantes de ingeniería de organización industrial, encargados de la parte de operaciones y negocios del equipo.

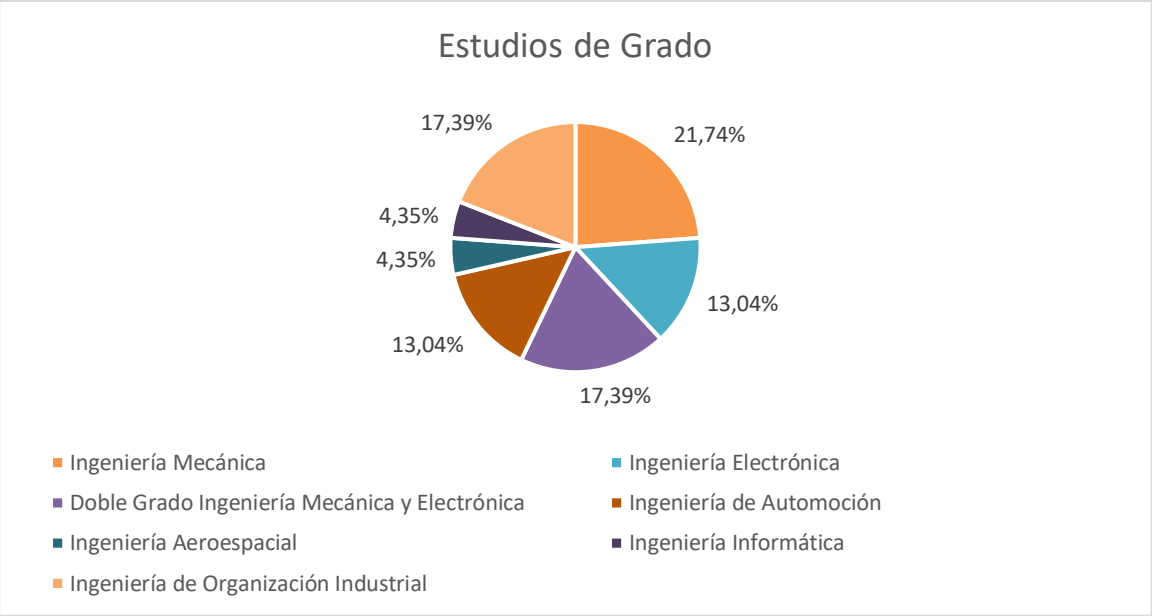


Ilustración 8 Estudios de Grado

1.3.1.3. Distribución de sexos

En el equipo, como en la gran mayoría de las ingenierías, el género predominante es el masculino. De los 23 integrantes, diecisiete eran hombres representando un 73,91% del total, y seis eran mujeres representando el 26,09% restante.

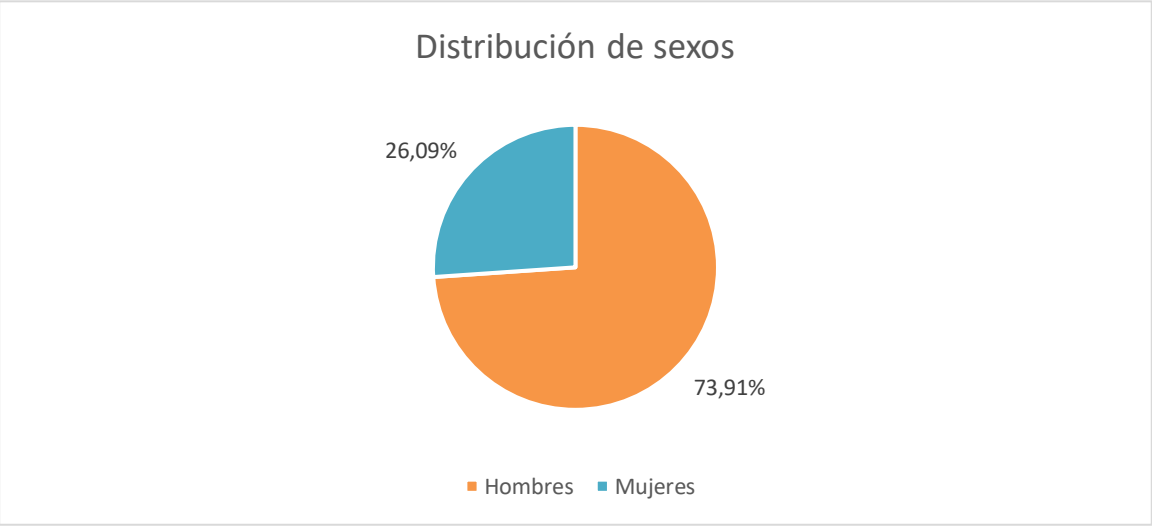


Ilustración 9 Distribución de sexos

1.3.2. Competiciones

El equipo ha competido tres temporadas a lo largo de su historia, tres veces en España, una vez en Italia y una última en Austria, con tres monoplazas diferentes basados en la evolución del diseño original: EM-01, EM-02 y EM-03. Actualmente se está trabajando a nivel conceptual en el nuevo modelo EM-04.

2. Objeto del proyecto

El objeto de este proyecto es la conceptualización, el dimensionado y el desarrollo de un sistema de obtención y procesado de datos para un monoplaza de competición basándose en la normativa FSG [7] vigente para la temporada 2021-2022.

Ilustración 10 Monoplazas EM-01, EM-02 y EM-03 en competición

La motivación de este proyecto reside en ayudar al equipo de *Formula Student* EUSS Motorsport realizando un proyecto integral viable en el que se tomen en cuenta las necesidades del equipo, los recursos disponibles y los requerimientos físicos del monoplaza.

En la línea temporal del equipo, el proyecto se implementará en paralelo con la construcción del monoplaza EM-04.

En el alcance de este proyecto se contemplan todas las fases de diseño, implementación y desarrollo conceptual del sistema.

3. Descripción del monoplaza

Para empezar a desarrollar el proyecto hay que tener en cuenta el punto del que se parte y lo que están haciendo otros equipos, y más importante, porqué lo están haciendo así.

Actualmente el monoplaza del cual se parte, el EM-03, cuenta con los siguientes sensores:

- ECU: unidad de control que regula el motor dentro de un sistema electrónico compuesto por sensores y actuadores. Los sensores informan a la unidad central, y ésta envía la orden necesaria a los actuadores tras transformar la información inicial [8].

- Sensor de temperatura del agua refrigerante (ECT): sensor de temperatura que sirve tanto para ajustar la cantidad de combustible a inyectar como para conectar el ventilador del radiador. En el primer caso, si el motor esta frio se deberá inyectar una mayor cantidad de combustible para contrarrestar las pérdidas por condensación en el colector de admisión y calentar más rápido el motor. En el segundo, si el sensor detecta que la temperatura es superior a 90ºC, la ECU mandará una señal al ventilador para que se encienda y refrigere el motor gracias al flujo de aire [9].

- Sensor de oxígeno o sonda lambda: sensor que mide la cantidad residual de oxígeno en los gases de escape. Si hay mucho oxígeno en los gases de escape la mezcla quemada es pobre (tiene exceso de aire) y si hay poco oxígeno la mezcla es rica [10].

- Sensor de posición del cigüeñal o *Crankshaft Position Sensor* (CRANK): sensor magnético que genera voltaje usando un sensor y una rueda con dientes perdidos solidaria al cigüeñal. Envía información a la ECU relativa a la posición de los pistones en un ciclo de motor. Con esta información la unidad central del motor es capaz de calcular las RPM's y la secuencia de inyección [11].

- Sensor del árbol de levas o *Camshaft Position Sensor* (CAM): sensor, normalmente magnético, que controla la posición de las válvulas de admisión y escape. Se encuentra

en la rampa de admisión que conecta la bomba de gasolina al motor, y su función es sincronizar el inyector y la secuencia de disparo de la bobina en los inyectores [12]. El sensor CAM trabaja alineado con el sensor CRANK. La información recogida sirve para saber cuándo el pistón está lo suficientemente arriba para una potencial inyección y confirma que las válvulas están alineadas para la entrada de aire y combustible [13].

- Sensor de posición del acelerador o *Throttle Position Sensor* (TPS): sensor que forma parte del sistema de gestión del combustible asegurando que la mezcla de aire y gasolina que le llega al motor es precisa en relación con la posición del acelerador [14].

- Sensor de presión del aceite o *Oil Pressure Sensor* (OPS): sensor encargado de medir a distancia la presión del aceite del motor. Su funcionamiento se basa en un filamento capaz de regirse por el principio de presión atmosférica, haciendo que cuando el flujo de aceite ejerce fuerza en la cabeza del sensor, se envía un voltaje a la ECU gracias a un conector de pines que hay en el otro extremo del sensor [15].

- Dispositivo de plausibilidad del sistema de frenos o *Brake System Plausibility Device* (BSPD): placa PCB no programable utilizada para monitorizar electrónicamente el control del acelerador. Su funcionamiento se basa en que se cumplan las siguientes condiciones:

    - De forma conjunta:

        - Frenada fuerte (>0.8 G's de deceleración sin la necesidad de bloquear las cuatro ruedas).

        - La mariposa de admisión del acelerador[9] esté un 10% abierta.

    - Pérdida de señal del sensor de freno durante más de 100msec.

[9] Válvula típica de los motores térmicos con inyección de combustible. Forma parte de su sistema de admisión y regula mecánicamente el flujo de aire que entra al motor y formará parte del proceso de combustión, aumentando o disminuyendo el paso de éste. Complementa al TPS**Fuente especificada no válida.**.

- Pérdida de señal del TPS durante más de 100msec.
- Pérdida de alimentación en el circuito de la BSPD. En caso de cumplirse alguna de las condiciones anteriores, la BSPD debe cortar la corriente del acelerador electrónico, cortar el flujo de combustible y posicionar en neutro la mariposa del acelerador [16].

- Sensor inercial o *Inertial Measurement Unit* (IMU): componente diseñado para obtener la posición, orientación y velocidad de cualquier dispositivo donde sea utilizado. Está compuesto por tres sensores:
  - Giroscopio encargado de medir los giros realizados.
  - Acelerómetro para medir la aceleración lineal en cualquier dirección.
  - Magnetómetro para obtener información sobre el norte magnético y poder recrear el recorrido del monoplaza gracias a un sistema por coordenadas [17].[10]

[10] Fuente de las ilustraciones: programa de modelado 3D SolidWorks.

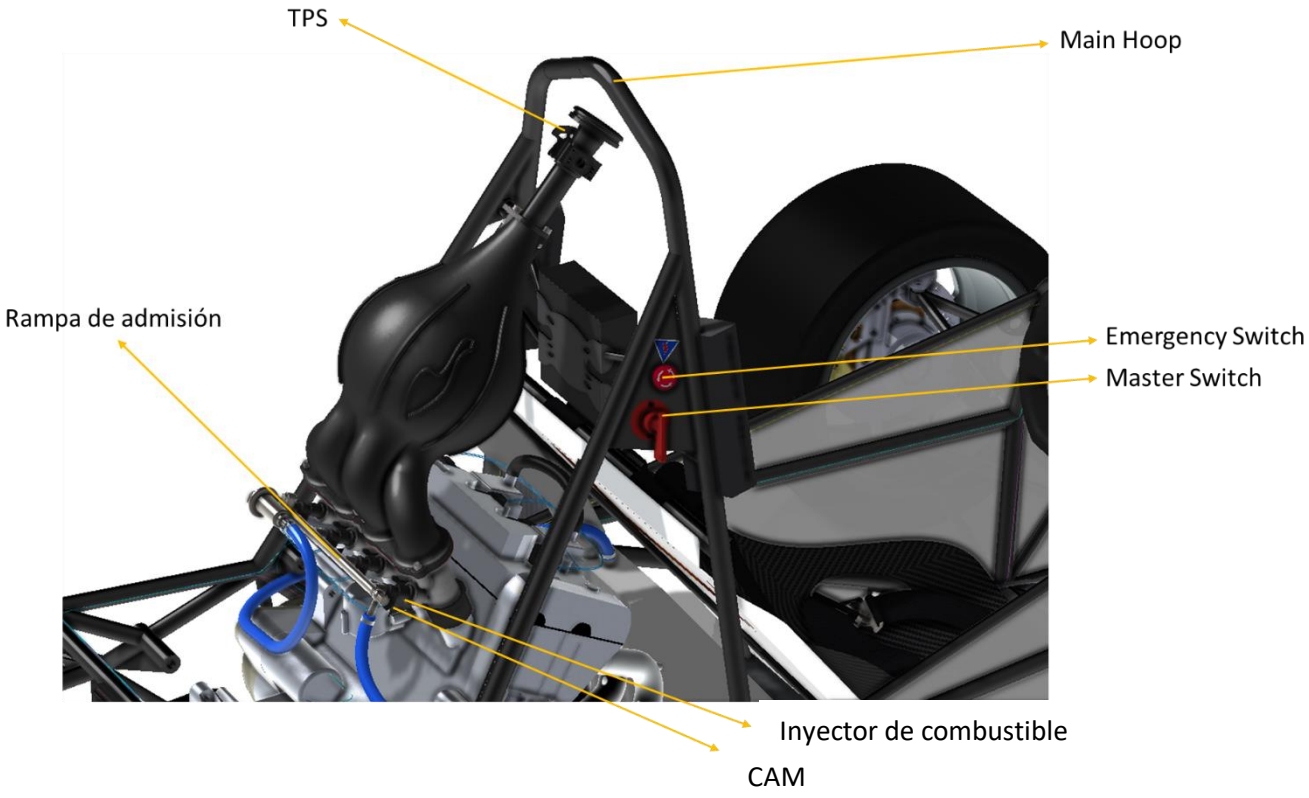


Ilustración 11 Disposición de elementos en el monoplaza

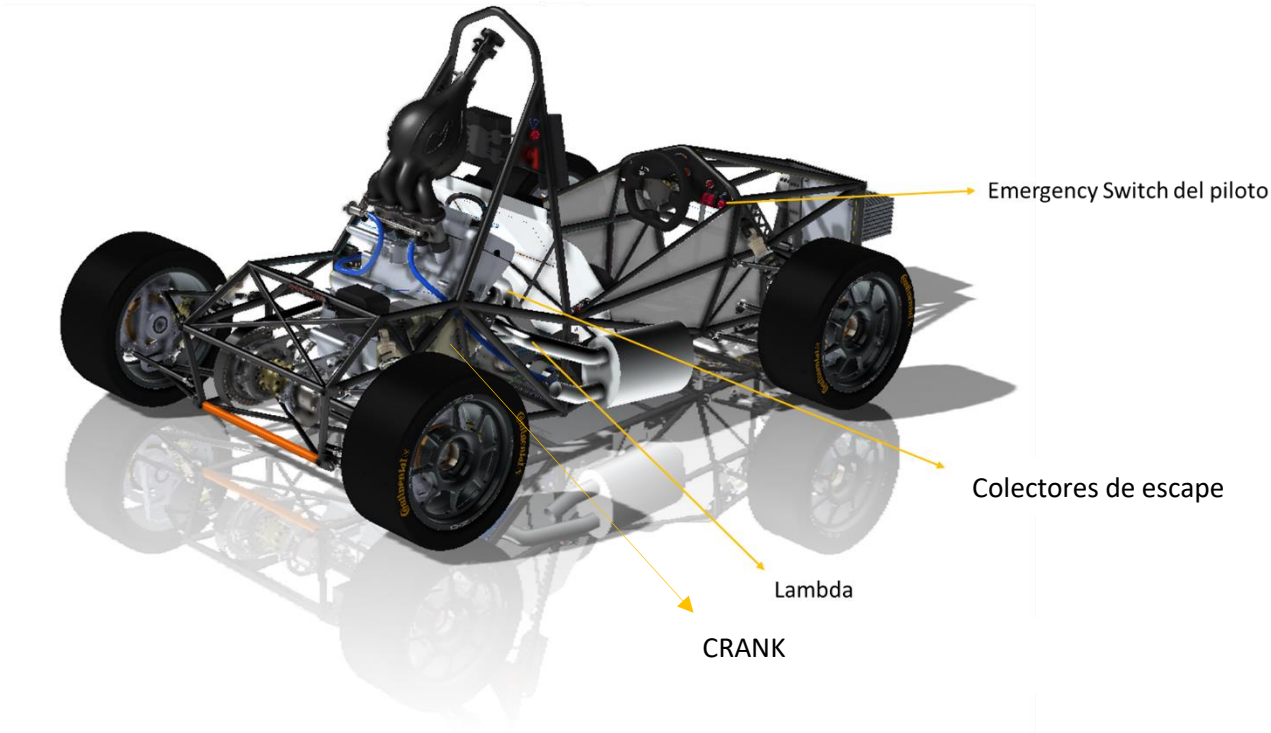


Ilustración 12 Disposición de elementos en el monoplaza

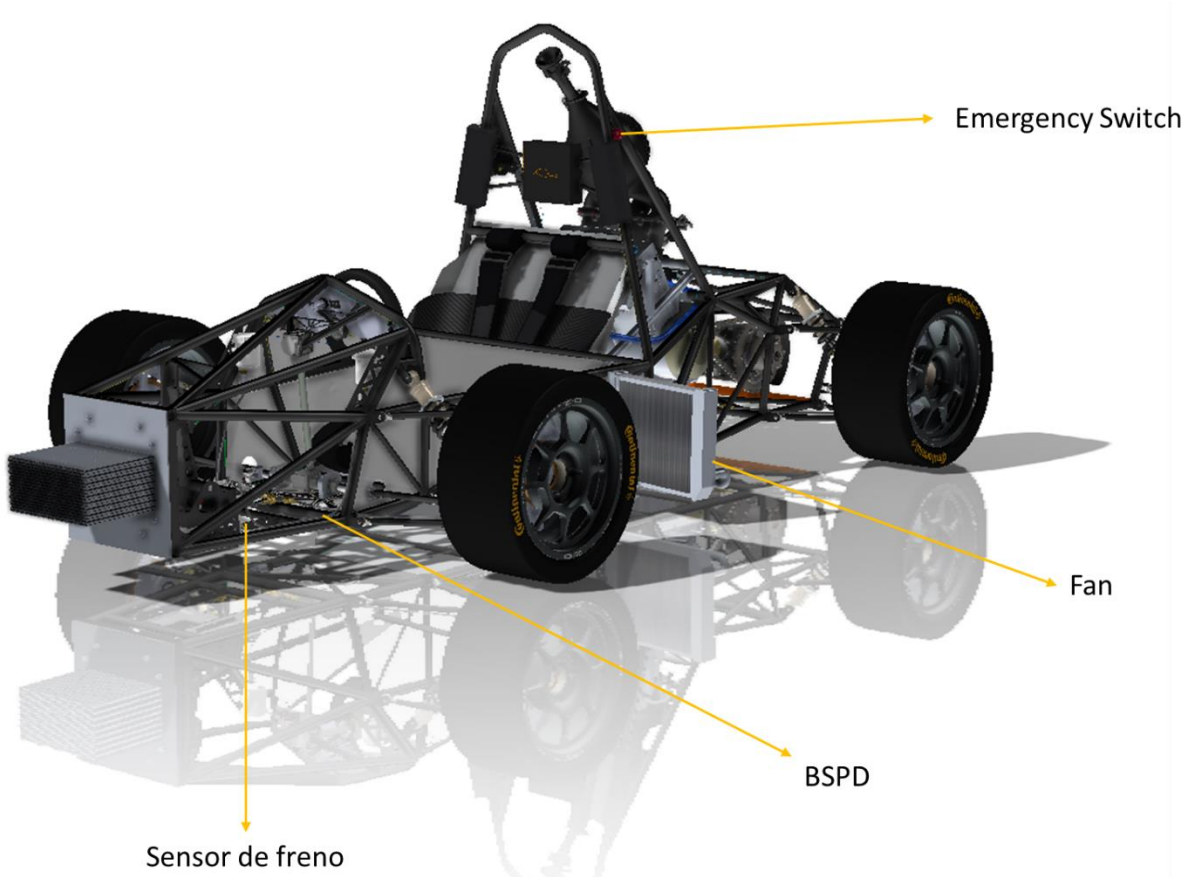


Ilustración 13 Disposición de elementos en el monoplaza

Referente a la adquisición de datos, hay instaladas en el monoplaza una Raspberry Pi 3B+ que almacena los datos proporcionados por la ECU y un Arduino, y un Arduino Nano que envía los datos recopilados por el sensor IMU a la Raspberry.

Si lo comparamos con otros equipos, en concreto aquellos con más años en la competición, ellos cuentan con un gran catálogo de sensores (aceite, velocidad para las ruedas y el diferencial, GPS, galgas extensiométricas, recorrido de suspensión, LiDAR, *data loggers*…) y un sistema de telemetría a distancia que les permite conocer el estado del coche desde cualquier lugar de la pista. Los datos recopilados los analizan con softwares de cálculo especializados como Matlab u OptimumLap.

La integración del conjunto de sensores y el sistema de postproceso de datos les permite aplicar mejoras *in situ* en caso de ser necesario en caso de sufrir una avería o querer optimizar las características del monoplaza para una prueba dinámica en concreto.

Este proyecto depende de la adquisición de una serie de elementos entre los que se incluye un *data logger* y diferentes placas, diseñadas específicamente para cada sensor. Estos elementos necesarios ya fueron comprados con el presupuesto de la temporada anterior, aunque a nivel de hardware todavía faltan algunos circuitos de cableado para poder implementarlo.

A nivel de software se necesita una IDE donde poder programar el funcionamiento del sistema *CAN bus* y los programas de cálculo específico para cada departamento.

En lo referente a fuentes de información son necesarios *papers* y documentos internos del equipo, así como un trabajo de *benchmarking* de otros equipos para delimitar las posibilidades dentro del ámbito de la *Formula Student*.

La documentación necesaria para el desarrollo del proyecto se centra en cuatro puntos: el sistema de comunicación de *CAN bus*, la diferencia entre comunicación *Serial*, *CAN bus* y *profiNET IO*, comunicación por radiofrecuencia y conexiones con servidor para el acceso remoto de los datos.

Durante la temporada 2020-2021 el departamento de electrónica del equipo hizo un prototipo de un sistema basado en la comunicación *CAN bus* con el propósito de conectar un *data logger* y la ECU a una Raspberry que actuase como cerebro principal.

Finalmente, no llegó a implementarse por falta de tiempo y problemas de configuración del data *logger*.

4. Objetivos y alcance

Las decisiones tomadas en este trabajo son del propio equipo, donde la autora del TFG tiene la responsabilidad de desarrollar todo el sistema de adquisición de datos junto con los miembros del departamento de electrónica y de acuerdo con las exigencias del resto del equipo de diseño de EUSS Motorsport.

4.1. Objetivos

El objetivo principal del proyecto se basa en el desarrollo de un sistema de adquisición de datos para el monoplaza EM-04 del equipo de *Formula Student* EUSS Motorsport que sea capaz de centralizar la información recogida por los diferentes sensores en un único documento. La finalidad de ese documento es proporcionarles a los diferentes departamentos información suficiente para poder mejorar sus diseños e implementaciones.

Este objetivo se puede dividir en una serie de objetivos más concretos:

- Garantizar una comunicación fiable entre la ECU, los sensores y las placas Raspberry y Arduino mediante el uso de un protocolo de mensajería MQTT.
- Crear un sistema donde se puedan incorporar nuevos sensores al conjunto sin un aumento de la complejidad de gestión.
- Preparar el sistema de adquisición de datos para el paso a motor eléctrico e integración con una nueva ECU.
- Preparar el monoplaza para el salto al coche autónomo en la temporada 2022-2023 mediante el uso de un sensor LiDAR.
- Reducir las incidencias en el departamento de electrónica en un 25% mediante la creación de un sistema de alertas para lecturas fuera de rango que avise en caso de avería de cualquier componente.
- Agilizar el traspaso de los archivos generados en los *runs* para poder hacer modificaciones in situ en los test.

4.2. Alcance del trabajo

En este trabajo se realizan todas las tareas necesarias para el desarrollo de un sistema de adquisición de datos para un monoplaza de *Formula Student* siguiendo unas normas y procedimientos que permitan evitar incoherencias, irregularidades o fallos en el cumplimiento de la normativa. Las tareas que se incluyen son:

- Análisis del estado del sistema de adquisición de datos actual con el objetivo de ver puntos no cubiertos.
- Análisis de los datos extraídos en anteriores test y competiciones con el fin de definir la estructura del archivo que se quiere implementar.

5. Metodología

5.1. Herramienta de control

Durante el ciclo de vida del proyecto se realizan reuniones quincenales con el tutor con el fin de ver el estado en el que se encuentra el proyecto y los pasos a implementar. De estas reuniones se redacta un acta con la fecha, asistentes, temas tratados y siguientes puntos. Después de cada reunión se genera una nueva versión del documento, así se tiene un historial de todo en lo que se ha trabajo hasta la fecha.

Existe un diagrama de Gantt como apoyo a estas reuniones. En él están señalizadas fechas importantes como las entregas y hay definidos varios *sprints* relacionados con los pasos a implementar.

Todos los documentos y enlaces de interés se encuentran en una carpeta de Drive donde tanto el tutor como la autora del TFG tienen permisos de lectura y edición.

Antes de cada reunión el tutor hace una revisión del contenido añadido y , en caso de ver conveniente, realiza sugerencias de mejora.

5.2. División por tareas

En el proyecto participan cuatro perfiles: analista, electricista, diseñador y administrativo.

- El perfil de analista se encarga de la programación del centralizado de datos, de los sensores y de las comunicaciones entre los principales módulos. Estos módulos son la ECU, la Raspberry Pi y los Arduino.

- El perfil de electricista se encarga de realizar un estudio de las conexiones del prototipo y de su integración con los circuitos existentes del monoplaza. También se encarga del diseño de los esquemas de cableado y su representación en 3D.

- El perfil de diseñador se encarga del dibujo de CAD, posicionando la representación de los cables hecha por el electricista en el monoplaza. Ayuda al electricista a colocar los cables en el monoplaza de forma física.

- El perfil administrativo se encarga de documentar todo lo que hacen los perfiles anteriores. Se asegura de que se siguen los formatos y de que no falte nada.

5.3. Búsqueda de información

La información documentada en este proyecto se recoge de *papers*, libros especializados, charlas formativas, documentación interna del equipo y páginas web certificadas.

6. *CAN bus* y *profiNET*

6.1. CAN bus

CAN bus o *Controller Area Network bus* [18] es un protocolo de comunicación en serie desarrollado por Bosch para el intercambio de información entre unidades de control electrónicas del automóvil que permite transportar una gran cantidad de información entre las unidades de control del sistema, lo que provoca una reducción importante tanto del número de sensores utilizados como de la cantidad de cables que componen la instalación eléctrica.

6.1.1. Funcionamiento

El funcionamiento de un sistema *CAN bus* en el mundo de la automoción se basa en una serie de sensores, pertenecientes o no a un mismo sistema, conectados en línea a las unidades del motor, del ABS, del cambio automático en caso de existir, y el sistema de confort del habitáculo.

El sistema de comunicación que se utiliza está orientado hacia el mensaje y no el destinatario, por lo que la información en línea es transmitida en forma de mensajes estructurados en la que una parte de este es un identificador que indica la clase de dato que contiene. Las unidades de control reciben el mensaje, lo filtran, y solo las que necesitan el dato gestionan el mensaje.

Cuando el bus está libre cualquier unidad conectada puede empezar a transmitir un nuevo mensaje, y en caso de que más de una unidad intente introducir un mensaje al mismo tiempo, lo hará la que tenga mayor prioridad. Esta prioridad viene indicada en el identificador.

6.1.2. Especificaciones técnicas

Los sistemas basados en *CAN bus* tienen dos líneas, la línea de alta velocidad y la de baja velocidad. Cuando no se está transmitiendo ningún mensaje por el bus, las dos líneas

mantienen el mismo voltaje (2.5V), en cambio cuando se envía un mensaje se genera una diferencia de voltaje entre ellas de 1.5V, donde *CAN bus* de alta velocidad tiene un voltaje de 3,75V y el de baja intensidad de 1.25V. De esta forma cuando un dispositivo quiera enviar un mensaje este podrá saber en qué estado se encuentra el bus.

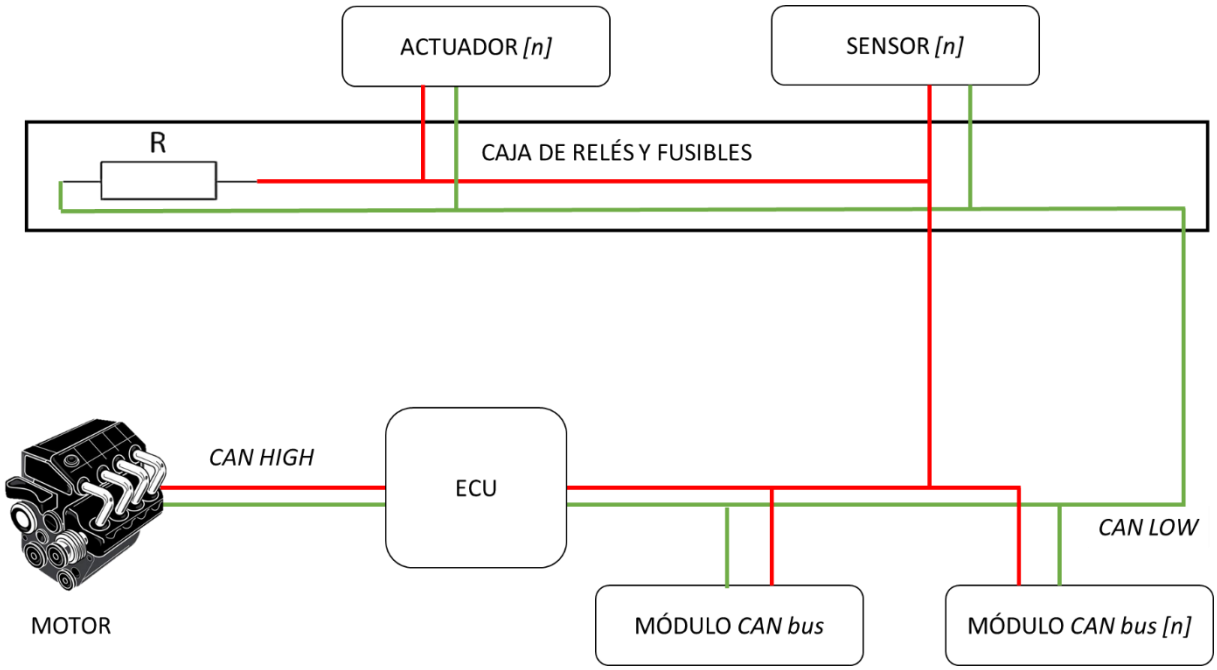


Ilustración 14 Esquema de un sistema *CAN bus*

6.1.3. Características

- Prioridad de mensajes: prioriza los mensajes (en la trama de comunicación se especifica la prioridad del mensaje).
- Garantía de tiempos: cumple los tiempos de transmisión de los mensajes dicho por el fabricante.
- Flexibilidad en la configuración: se puede configurar de tal manera que al usuario le sea más fas fácil.
- Sistema robusto en cuanto a consistencia de datos: no se pierden datos en la transmisión del mensaje.

- Sistema multimaestro: no hay un maestro que controla el bus, sinó que es un protocolo multimaestro donde todos son maestros.
- Detección y señalización de errores: el mismo protocolo avisa de errores de transmisión de datos y de trama.
- Retransmisión automática de tramas erróneas: si se ha perdido algún mensaje en la transmisión de la trama el propio sistema vuelve a lanzar el mensaje.

6.1.4. Ventajas y desventajas

- Ventajas
    - Minimización de la cantidad de cables en el vehículo. Lo que simplifica el cableado y le da menor peso total al coche.
    - Menos sensores, ya que el mismo sensor da la misma información para varios sistemas electrónicos.
    - Menos conexiones entre las unidades de control.
    - Mejor rendimiento de los componentes.
    - Medición de todos los aspectos electrónicos del automóvil.
    - Diagnóstico integrado y aviso de fallas.
- Desventajas
    - Entre más larga sea la distancia, la velocidad se reduce (de transmisión) a partir de 40 m.
    - Coste del software.
    - Interacciones indeseadas más probables.
    - Peligro por mala programación.

6.2. profiNET

profiNET I/O [18] es un mecanismo para intercambiar datos entre controladores y dispositivos. Los controladores pueden ser PLCs (controladores lógicos programables), DCSs (sistemas de control distribuido) o PACs (controladores de automatización programables). Los dispositivos pueden ser bloques I/O, sistemas de visión, lectores RFID, drives, instrumentos de proceso, proxis o incluso otros controladores.

6.2.1. Funcionamiento

profiNET usa la transferencia de datos cíclicos para intercambiar datos con controladores programables a través de Ethernet. Un controlador programable y un dispositivo deben tener una comprensión previa de la estructura y el significado de los datos. En ambos sistemas los datos se organizan como ranuras que contienen módulos con el número total de puntos I/O para un sistema y la suma de los puntos I/O para los módulos individuales.

profiNET utiliza tres canales de comunicación diferentes para intercambiar datos con controladores programables y otros dispositivos:

- Canal TCP/IP estándar se utiliza para la parametrización, configuración y operaciones de lectura/escrituras acíclicas.

- Canal RT o *Real Rime* se utiliza para la transferencia de datos y alarmas cíclicas estándar. Este tipo de comunicaciones omiten la interfaz TCP/IP estándar para acelerar el intercambio de datos con controladores programables.

- Canal IRT o *isochronous real time* se utiliza para aplicaciones de control de movimiento debido a la velocidad con la que trabaja.

6.2.2. Características

- *profiNET* I/O ofrece funcionamiento en “tiempo real” para datos de E/S cíclicos. Tiempo real significa programar/organizar el intercambio cíclico con cada esclavo, con alta prioridad y tiempos fijos.

- *profiNET* I/O es a veces llamado *profiNET* -RT (*Real Time*).
- Se pueden utilizar los cables y *switches* estándar de *Ethernet*.
- Sistema Maestro-Esclavo.
- Se configura como una red de campo.
- Los dispositivos ya no se direccionan mediante número de nodo, sino mediante un nombre.
- Comunicación fácil, rápida, flexible y abierta.

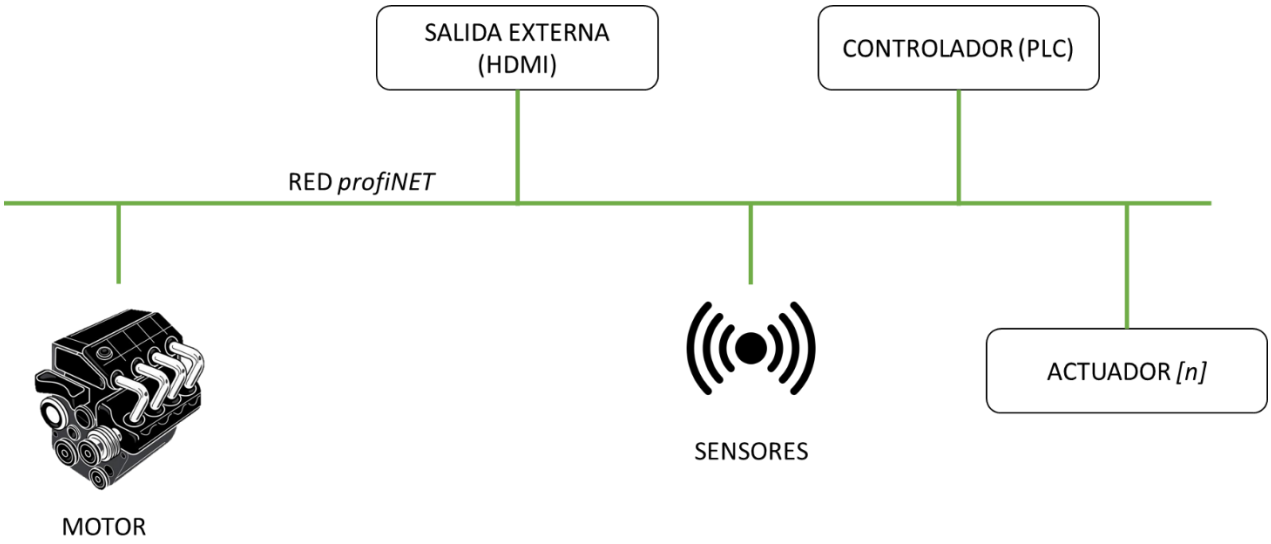


Ilustración 15 Esquema de un sistema *profiNET*

6.2.3. Ventajas y desventajas

- Ventajas
  - Mejora la escalabilidad en las infraestructuras.
  - Acceso a los dispositivos de campo a través de la red. *profiNET* al ser un protocolo que utiliza Ethernet en su comunicación facilita acceder a dispositivos de campo desde otras redes de una forma más fácil.

  - Ejecución de tareas de mantenimiento y prestación de servicio desde cualquier lugar. Es posible acceder a dispositivos de campo mediante conexiones seguras como por ejemplo VPN para realizar mantenimientos remotos.
  - Ofrece un acceso transparente a la planta con servicios estándares remotos incluyendo Mantenimiento, Diagnostico y parametrización, puesta en marcha y cambios de programa.
  - Tiempos de parada menores y aumento de la eficiencia en el mantenimiento.
  - Permite la conexión de más nodos.
  - Rendimiento hasta cien veces mejor en control de movimiento.
  - Acceso a los datos con herramientas estándares de oficina.
  - Comunicación sin cable con Wireless LAN Industrial.

- Desventajas
  - Si por cualquier motivo el repetidor o *switch* central llega a fallar, se pierde la señal completamente en todas las áreas de la conexión.
  - Es costosa debido a que requiere mayor cantidad de cables a diferencia de topologías en bus o anillo.

6.3. Por qué CAN bus

Después de ver las características, ventajas y desventajas de las dos tecnologías, se ha decidido utilizar un sistema basado en *CAN bus* debido a que para el uso concreto en el que se va a utilizar *CAN bus* aporta mayor seguridad y estabilidad que *profiNET*, apuesta más arriesgada por el tipo de proyecto y el tiempo del que se dispone.

Requerimiento	**Peso del requerimiento**	***CAN bus***	***profiNET***
Fácil de configurar	10	Sí	No
Rapidez de transmisión	10	Sí	Sí
Minimización de cableado	7	Sí	No
Escalable	8	Sí	Sí
Acceso remoto a los datos	10	Sí	Sí
Robustez	9	Sí	No
Fácil de mantener	10	Sí	Sí
Aviso de fallas	10	Sí	Sí
TOTAL	**74**	**74**	**48**

Tabla 1 Comparativa *CAN bus* - *profiNET*

7. MQTT

MQTT o *Message Queuing Telemetry Transport* es un protocolo de mensajería ligero utilizado en clientes que necesitan una huella de código pequeña, que están conectados a redes no fiables o con recursos limitados en ancho de banda. Principalmente se aplica a comunicaciones de máquina a máquina o en conexiones de internet de las cosas [19].

7.1. Origen del protocolo

El protocolo MQTT fue creado por el Dr Andy Stanford-Clark y Arlen Nipper en el año 1999. Su propósito original era permitir que los dispositivos de monitoreo utilizados en la industria del petróleo y gas enviaran sus datos a servidores remotos situados en ubicaciones remotas donde el establecimiento de línea fija, conexión por cable o enlace de transmisión de radio sería imposible. En ese caso, la única opción viable eran las comunicaciones por satélite, costosas debido a que se facturaban en base de la cantidad de datos utilizada. En ese momento nace MQTT, proporcionando un sistema de comunicación y transmisión de datos fiable empleando un ancho de banda mínimo.

MQTT fue estandarizado como código abierto en 2013 gracias a la organización para el avance de estándares de información estructurada u *Organization for the Advancement of Structured Information Standards* (OASIS[11]).

7.2. Arquitectura

MQTT se ejecuta sobre TCP/IP usando una tipología PUSH/SUBSCRIBE. Para esta arquitectura se establecen dos sistemas diferenciados, los clientes y los bróker. Los clientes (que pueden tener rol de editor, suscriptor o ambos a la vez) no se comunican entre sí, sino que se

[11] https://www.oasis-open.org/

comunican con el bróker, un servidor que recibe y distribuye las comunicaciones de los diferentes clientes.

El MQTT es un protocolo controlado por eventos, por lo que la transmisión de datos es la mínima esencial. Un cliente solo publicará cuando requiera enviar información, y un bróker solo comunicará con los suscriptores conectados cuando reciba datos nuevos.

7.3. Mensajes en MQTT

La arquitectura de los mensajes también contribuye a la reducción del volumen de transmisión en MQTT. Los mensajes tienen un encabezado fijo de dos *bytes* y una carga útil máxima de 256 MB. Al mismo tiempo, se dispone de tres calidades de servicio (*Quality Of Service* o *QoS*) para optimizar al máximo la transmisión en función de las necesidades:

- *QoS* 0: Nivel más básico y ligero de transmisión, donde el bróker envía el mensaje a cada suscriptor una vez sin esperar confirmación de estos.
- *QoS* 1: Conocido como “Entregado al menos una vez”. El bróker envía el mensaje a los suscriptores y espera una confirmación de recepción de estos. De no recibirla vuelve a mandar el mensaje hasta recibir la confirmación.
- *QoS* 2: Conocido como “Entregado exactamente una vez”. Se establece un enlace de cuatro pasos entre el cliente y el bróker para garantizar que el cliente reciba todos los mensajes únicamente una vez.

Los mensajes se publican como ‘Temas’ con niveles de jerarquía. Esto permite delimitar fácilmente que suscriptores recibirán el mensaje, si todos los suscritos a la primera rama de temas o sólo los suscritos a la rama específica en la que se publica.

El bróker por su parte no retiene los mensajes que recibe a no ser que estén marcados con la bandera de retención. Estos mensajes retenidos son aquellos que los editores consideran importantes y que cualquier cliente debe recibir una vez se suscriba al tema, independientemente de la fecha de su suscripción.

Para mantener el protocolo al mínimo, se establecen únicamente cuatro posibles acciones en cualquier comunicación; publicar, suscribirse, cancelar suscripción o hacer PING:

- Publicar: Enviar un bloque de datos al bróker. Pueden ser instrucciones, lecturas de sensores u órdenes de encendido/apagado.
- Suscribirse: Convertir a un cliente en suscriptor hace que este reciba los nuevos mensajes del tema al que se ha suscrito, así como a mensajes retenidos del mismo tema.
- Cancelar suscripción: Un suscriptor o editor manda un mensaje de DISCONNECT al bróker para dejar de estar en la cola de mensajes de un tema y que ya no recibirá datos de este editor. El cliente puede volver a conectarse al bróker si lo desea usando la identidad anterior.
- PING: Un cliente puede comprobar que la conexión con el bróker siga funcionando correctamente haciendo PING al bróker mediante PINGREQ, a lo que el bróker le responderá con un PINGRESP.

8. Obtención de los datos

8.1. Arquitectura del sistema

El sistema se basa en siete elementos principales:

- ECU
- TunerStudio[12]
- Raspberry Pi 3B+
- Arduino nano
- Sensores
- Script para el procesado de datos
- *Cloud*

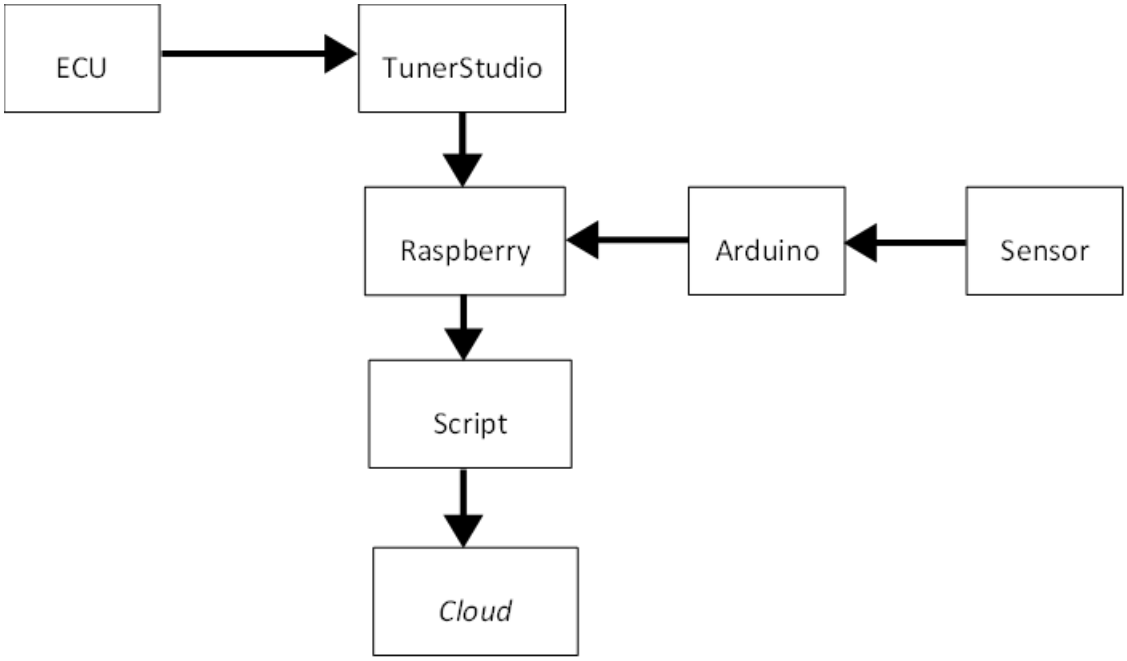


Ilustración 16 Arquitectura del sistema de adquisición de datos

[12] Programa utilizado para la recolección de datos de la ECU MegaSquirt MS2.

8.2. Formato de archivos obtenidos de la ECU

Los datos generados por la ECU son recolectados por el programa TunerStudio, quien se encarga de la generación de los archivos finales de datos en formato *.msl* que utiliza Raspberry para generar el procesado de datos antes de subirlos a la *cloud*.

En estos archivos los datos están estructurados en forma de tabla, donde la primera fila está reservada para el valor de las columnas y la segunda para las unidades:

- *Time*, tiempo, en nanosegundos [=ns]
- *SecL*, contador del serial secundario, en segundos [=s]
- *RPM* en revoluciones por minuto [=rpm]
- *MAP*, sensor de presión del aire, en kilopascales [=kPa]
- *Boost psi*, porcentaje de presión del turbo, en este caso como el motor es atmosférico el valor siempre será negativo y el porcentaje 0 [=%]
- *TPS*, posición del pedal del acelerador, en ratio aire-combustible [=%]
- *AFR*, ratio aire-combustible [=afr]
- *MAT*, sensor de temperatura del aire, en grados centígrados [=ºC]
- *CLT*, sensor de temperatura de líquido refrigerante, en binario [=bit]
- *Engine*, voltaje de motor, en voltios [=v]
- *Batt V*, nivel de batería, en porcentaje [=%]

9. Desarrollo del modelo de obtención y gestión de datos

El elemento principal de este proyecto es el desarrollo de un sistema que facilite la gestión de los datos obtenidos por el equipo en las diferentes competiciones y test. Para ello se ha diseñado un sistema capaz de coger los archivos ubicados en un directorio determinado, convertirlos para que la librería *pandas*[13] de Python pueda leerlos y poder así empezar el proceso de filtrado, limpieza, envío y almacenaje.

Para ver todas las librerías utilizadas y como obtenerlas referirse al Anexo I, para ver el código referirse al Anexo II.

9.1. Obtención y lectura de los datos

El primer paso es distinguir entre dos tipos de datos, los datos provenientes de la ECU y los datos generados por los sensores.

9.1.1. Obtención y lectura de los datos generados por la ECU

Los datos generados por la ECU contienen información sobre las métricas de motor, revoluciones por minuto, porcentaje de apertura de la entrada de admisión, relación entre la mezcla de aire y combustible, o temperatura de aceite y de refrigerante entre otros.

Estos datos son recogidos en unos archivos con extensión *msl* (.msl) que solo la aplicación MegaLogViewer[14] de la empresa que fabrica las ECU, EFI Analytics[15], puede leer. Es por eso por lo que se necesita convertir estos archivos a un formato legible por el lenguaje de programación Python. En este caso se ha decido convertir estos a archivos a formato TXT (.txt),

[13] Librería de Python utilizada como herramienta de análisis y manipulación de datos de código abierto rápida, potente, flexible y fácil de usar. Más información en: https://pandas.pydata.org/

[14] https://www.efianalytics.com/MegaLogViewer/

[15] http://www.tunerstudio.com/

donde los datos siguen la estructura del archivo original, valores organizados por columnas y separados por tabulaciones.

Un ejemplo de archivo generado por la ECU sería el siguiente:

Time	SecL	RPM	MAP	Boost psi		TPS	AFR	AFR2	MAT	CLT	Engine	Batt V	EGO cor1		EGO cor2	
	s	RPM	kPa		%	AFR	AFR	°C	°C	bit	v	%	%	%	%	%
0.000	2430	2085	98.3	-0.2	57.7	10.80	10.80	-17.7	52.5	5	13.4	100.0	100.0	100.0	111	100.0
0.003	2430	2309	98.2	-0.3	57.7	9.90	9.90	-17.7	52.5	5	13.2	100.0	100.0	100.0	111	100.0
0.036	2430	2180	98.3	-0.2	56.5	9.60	9.60	-17.7	52.5	5	13.2	100.0	100.0	100.0	111	100.0
0.079	2430	1871	98.3	-0.2	52.1	8.50	8.50	-17.7	52.5	5	13.3	100.0	100.0	100.0	111	100.0
0.129	2430	2054	98.3	-0.2	51.9	8.40	8.40	-17.7	52.5	5	13.4	100.0	100.0	100.0	111	100.0
0.185	2430	2028	98.3	-0.2	51.9	8.60	8.60	-17.7	52.5	5	13.4	100.0	100.0	100.0	110	100.0
0.245	2431	2173	98.3	-0.2	51.9	8.60	8.60	1.7	52.5	5	13.4	100.0	100.0	100.0	110	100.0
0.309	2431	2212	98.5	-0.2	51.9	8.40	8.40	-8.1	52.5	5	13.5	100.0	100.0	100.0	110	100.0
0.371	2431	1978	98.5	-0.2	51.9	8.30	8.30	-8.1	52.5	5	13.5	100.0	100.0	100.0	110	100.0
0.435	2431	2077	98.5	-0.2	51.9	8.10	8.10	-12.9	52.5	5	13.5	100.0	100.0	100.0	110	100.0
0.504	2431	2301	98.2	-0.3	51.9	7.90	7.90	-15.3	52.4	5	13.5	100.0	100.0	100.0	110	100.0
0.574	2431	2263	98.3	-0.2	51.9	7.80	7.80	-16.6	52.4	5	13.4	100.0	100.0	100.0	109	100.0
0.638	2431	2128	98.3	-0.2	51.9	7.80	7.80	-17.2	52.4	5	13.5	100.0	100.0	100.0	109	100.0
0.703	2431	2341	98.3	-0.2	51.9	7.90	7.90	-17.4	52.4	5	13.5	100.0	100.0	100.0	109	100.0
0.769	2431	2227	98.3	-0.2	51.9	7.90	7.90	-17.6	52.4	5	13.4	100.0	100.0	100.0	109	100.0
0.838	2431	2414	98.1	-0.3	52.6	8.00	8.00	1.7	52.4	5	13.5	100.0	100.0	100.0	109	100.0
0.904	2431	2479	98.1	-0.3	54.7	8.00	8.00	-8.0	52.4	5	13.4	100.0	100.0	100.0	108	100.0

Ilustración 17 Captura de pantalla de un fragmento de archivo generado por la ECU tras convertirlo a TXT

Una vez convertidos estos archivos a TXT, utilizando la librería *pandas* podemos abrir, leer y almacenar en un *dataframe*[16] el contenido de cada uno de los archivos por separado.

9.1.2. Obtención y lectura de los datos generados por los sensores

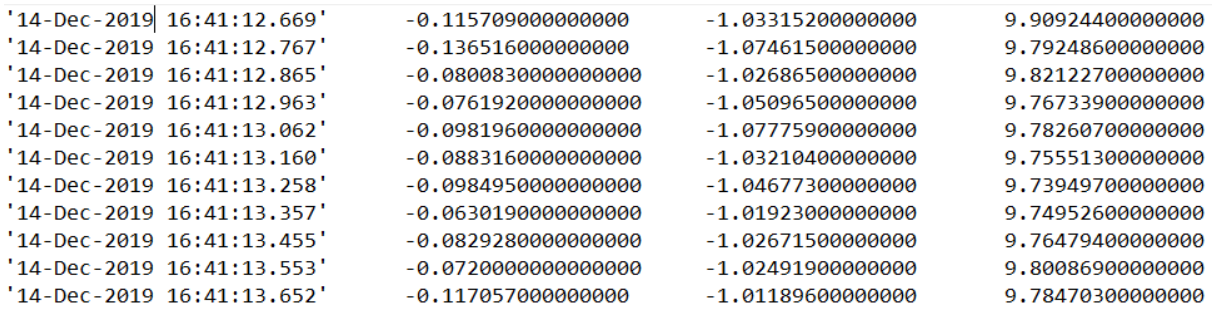
En el caso de los sensores, el proceso de obtención se realiza de forma independiente en cada uno de los Arduino NANO instalados a lo largo del monoplaza, y la recogida de datos se hace directamente en archivos TXT, por lo que no es necesario hacer la conversión de archivos y la lectura resulta más directa.

Para abrir, leer y almacenar los datos se sigue utilizando la misma librería que en el caso anterior, *pandas*, y el sistema de almacenado en local también se lleva a cabo mediante un *dataframe*.

[16] Paneles bidimensionales compuestos por filas y columnas, que permiten destacar las relaciones entre las distintas variables de la serie de datos [21].

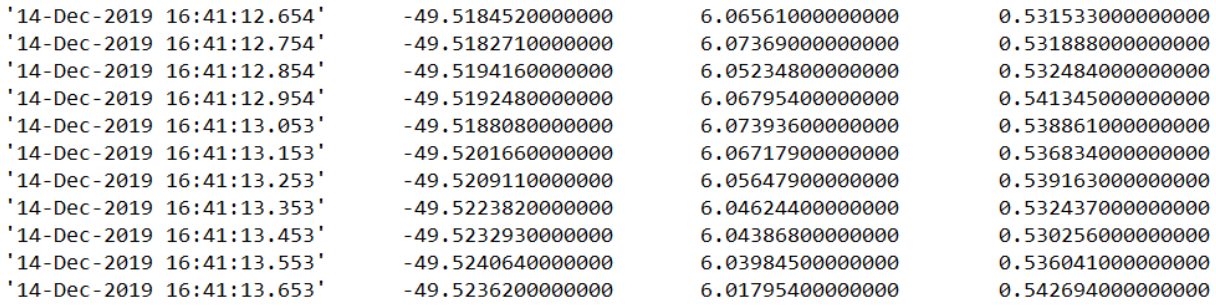
Hay que tener en cuenta que para este tipo de datos se requiere de un trabajo previo donde se estudia la salida del sensor y se programa la recogida de datos procedentes de la lectura de para convertirlos en datos funcionales, tabulados, y relacionados en función del tiempo.

Un ejemplo de archivo generado por un sensor, en este caso el sensor IMU, sería el siguiente:



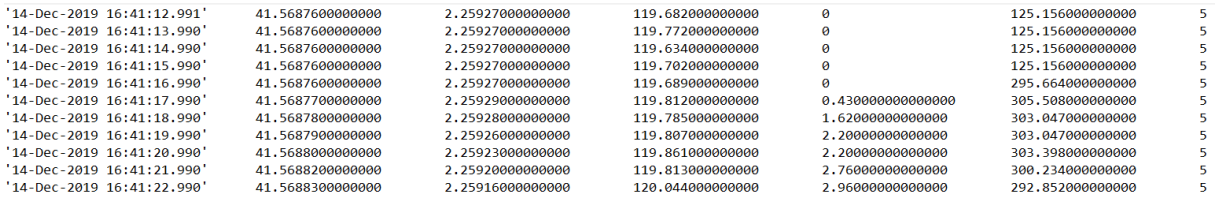
'14-Dec-2019 16:41:12.669'	-0.115709000000000	-1.03315200000000	9.90924400000000
'14-Dec-2019 16:41:12.767'	-0.136516000000000	-1.07461500000000	9.79248600000000
'14-Dec-2019 16:41:12.865'	-0.0800830000000000	-1.02686500000000	9.82122700000000
'14-Dec-2019 16:41:12.963'	-0.0761920000000000	-1.05096500000000	9.76733900000000
'14-Dec-2019 16:41:13.062'	-0.0981960000000000	-1.07775900000000	9.78260700000000
'14-Dec-2019 16:41:13.160'	-0.0883160000000000	-1.03210400000000	9.75551300000000
'14-Dec-2019 16:41:13.258'	-0.0984950000000000	-1.04677300000000	9.73949700000000
'14-Dec-2019 16:41:13.357'	-0.0630190000000000	-1.01923000000000	9.74952600000000
'14-Dec-2019 16:41:13.455'	-0.0829280000000000	-1.02671500000000	9.76479400000000
'14-Dec-2019 16:41:13.553'	-0.0720000000000000	-1.02491900000000	9.80086900000000
'14-Dec-2019 16:41:13.652'	-0.117057000000000	-1.01189600000000	9.78470300000000

Ilustración 18 Captura de pantalla de un fragmento de archivo generado por el sensor IMU (aceleración en x, y, z)



'14-Dec-2019 16:41:12.654'	-49.5184520000000	6.06561000000000	0.531533000000000
'14-Dec-2019 16:41:12.754'	-49.5182710000000	6.07369000000000	0.531888000000000
'14-Dec-2019 16:41:12.854'	-49.5194160000000	6.05234800000000	0.532484000000000
'14-Dec-2019 16:41:12.954'	-49.5192480000000	6.06795400000000	0.541345000000000
'14-Dec-2019 16:41:13.053'	-49.5188080000000	6.07393600000000	0.538861000000000
'14-Dec-2019 16:41:13.153'	-49.5201660000000	6.06717900000000	0.536834000000000
'14-Dec-2019 16:41:13.253'	-49.5209110000000	6.05647900000000	0.539163000000000
'14-Dec-2019 16:41:13.353'	-49.5223820000000	6.04624400000000	0.532437000000000
'14-Dec-2019 16:41:13.453'	-49.5232930000000	6.04386800000000	0.530256000000000
'14-Dec-2019 16:41:13.553'	-49.5240640000000	6.03984500000000	0.536041000000000
'14-Dec-2019 16:41:13.653'	-49.5236200000000	6.01795400000000	0.542694000000000

Ilustración 19 Captura de pantalla de un fragmento de archivo generado por el sensor IMU (orientación en x, y, z)



'14-Dec-2019 16:41:12.991'	41.5687600000000	2.25927000000000	119.682000000000	0	125.156000000000	5
'14-Dec-2019 16:41:13.990'	41.5687600000000	2.25927000000000	119.772000000000	0	125.156000000000	5
'14-Dec-2019 16:41:14.990'	41.5687600000000	2.25927000000000	119.634000000000	0	125.156000000000	5
'14-Dec-2019 16:41:15.990'	41.5687600000000	2.25927000000000	119.702000000000	0	125.156000000000	5
'14-Dec-2019 16:41:16.990'	41.5687600000000	2.25927000000000	119.689000000000	0	295.664000000000	5
'14-Dec-2019 16:41:17.990'	41.5687700000000	2.25929000000000	119.812000000000	0.430000000000000	305.508000000000	5
'14-Dec-2019 16:41:18.990'	41.5687800000000	2.25928000000000	119.785000000000	1.62000000000000	303.047000000000	5
'14-Dec-2019 16:41:19.990'	41.5687900000000	2.25926000000000	119.807000000000	2.20000000000000	303.047000000000	5
'14-Dec-2019 16:41:20.990'	41.5688000000000	2.25923000000000	119.861000000000	2.20000000000000	303.398000000000	5
'14-Dec-2019 16:41:21.990'	41.5688200000000	2.25920000000000	119.813000000000	2.76000000000000	300.234000000000	5
'14-Dec-2019 16:41:22.990'	41.5688300000000	2.25916000000000	120.044000000000	2.96000000000000	292.852000000000	5

Ilustración 20 Captura de pantalla de un fragmento de archivo generado por el sensor IMU (posición en x, y, z, valor del giroscopio en x, y, z)

9.2. Limpieza y filtrado

El primer paso después de obtener los datos es limpiarlos. En el caso del *dataframe* generado con los valores de la ECU este proceso se resume en eliminar las filas no necesarias, las dos primeras filas del *dataframe* que contienen información sobre la generación del archivo en

.msl y la cuarta fila, la cual contiene información referente al valor con el que se miden las columnas (porcentajes, vatios, RPM…). Como limpieza final, se sustituyen todos aquellos valores no numéricos, *NaN*, por ceros.

En el caso de los datos recogidos por los sensores, los datos no contienen información no relevante, por lo que solo hace falta substituir los valores no numéricos por ceros para asegurar un procesado de datos funcional.

9.3. Envío y almacenamiento de datos

Los datos almacenados en el *dataframe* son enviados utilizando el protocolo MQTT. Para ello se necesita generar un bróker encargado de enviar todos los datos valor por valor a un servidor mediante un sistema de tópicos para el envío y recibimiento. En este caso el servidor utilizado se trata del servidor MQTT *Eclipse Projects de la empresa Eclipse foundation*[17]. El motivo de esta elección se basa en que se trata de un servicio MQTT gratuito ofrecido por una empresa con la que ya se ha trabajado durante la carrera dentro de otros entornos.

De forma paralela, los datos son enviados a una base de datos privada creada en MongoBD y hospedada dentro del entorno en el *cloud* de Azure[18]. Para que el envío sea posible es necesario crear un sistema usuario – contraseña para poder acceder a la base de datos. Cabe destacar que la conexión solo es posible si la IP desde la que el usuario realiza la conexión es una IP fiable, es decir, si la IP ha sido previamente aprobada desde un dispositivo con acceso a la base de datos.

Dentro de MongoBD, los datos se almacenan por temas, creando una tabla de datos para cada columna del *dataframe*. Estos datos se muestran en formato JSON[19] y cuentan con una cadena alfanumérica o ID para hacer posible su identificación.

---

[17] https://www.eclipse.org/projects/

[18] https://azure.microsoft.com/es-es/

[19] Formato ligero de intercambio de datos. *JavaScript Object Notation*: https://json.org/json-es.html

Estos datos son accesibles desde *MongoBD Compass*[20] y desde otros *scripts*. En la siguiente ilustración se puede ver un ejemplo de datos almacenados en la base de datos:

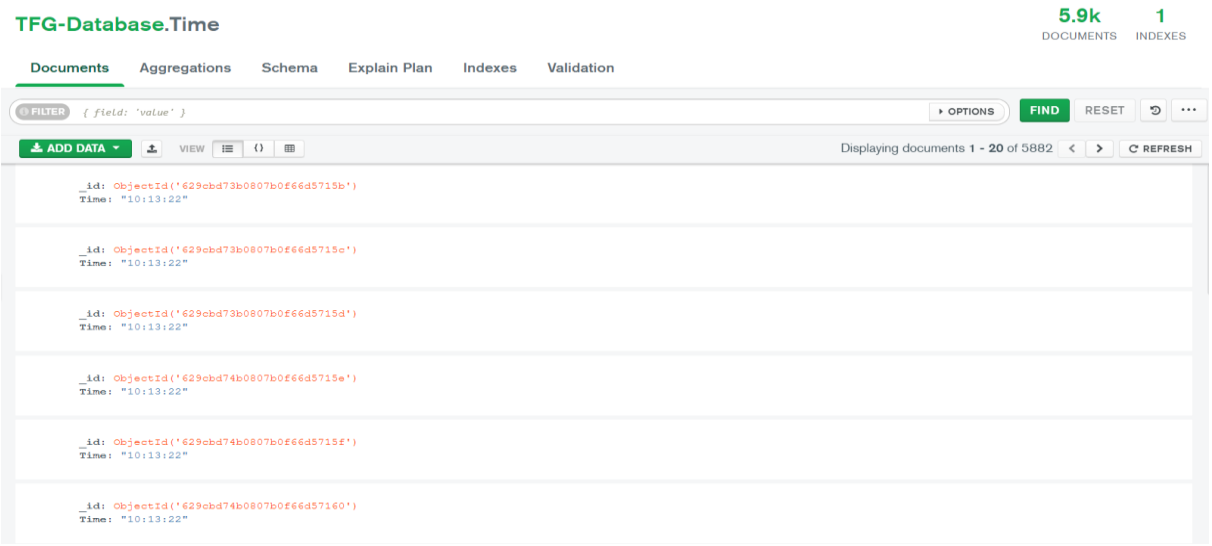


Ilustración 21 Captura de pantalla de fragmento de la base de datos MongoBD con información sobre el tema 'Tiempo'

En la siguiente ilustración se puede ver una vista generalizada de la base de datos:

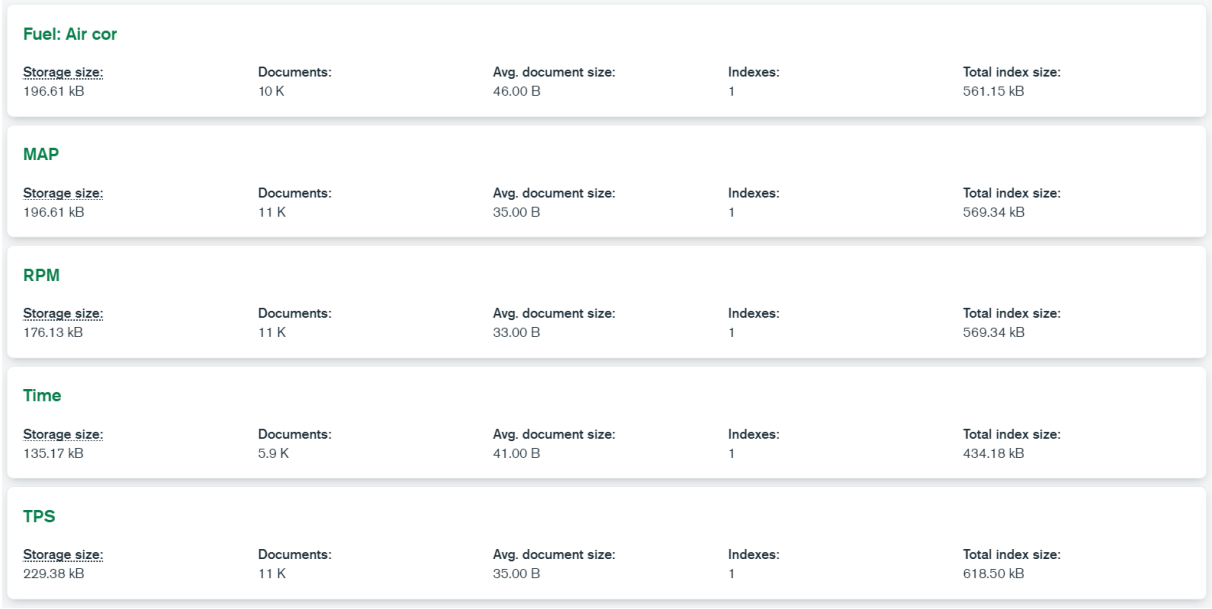


Ilustración 22 Captura de pantalla de un fragmento de las tablas almacenadas en la base de datos

[20] Interfaz gráfica de usuario para la conexión a bases de datos MongoDB. Más información en https://www.mongodb.com/products/compass

9.4. Recogida de datos de la base de datos

Una vez procesados y almacenados, los datos pueden ser recogidos de la base de datos para su futuro análisis. Para ello, es necesario tener acceso a la base de datos mediante IP y credenciales usuario – contraseña. En el *script* de recogida solo hace falta determinar la dirección a la base de datos donde están los datos junto con los credenciales.

Los datos se recogen recorriendo las tablas de datos una a una, fila por fila, y se almacenan en un nuevo *dataframe*. Una vez recogidos, este *dataframe* se convierte a un archivo csv.

9.5. Análisis y representación de datos

Con el archivo csv generado en el apartado anterior podemos crear un nuevo *dataframe*, el último, con el que se graficarán los datos a analizar.

Para hacer una demostración y comprobar la funcionalidad del sistema se han creado dos escenarios diferentes:

9.5.1. Primer escenario

Graficar los datos referentes a las RPM en función del tiempo para el test con el monoplaza EM-02 del día 28 de diciembre del 2020. Para ello se ha utilizado un modelo de gráfica de línea continua y dos ejes de datos.

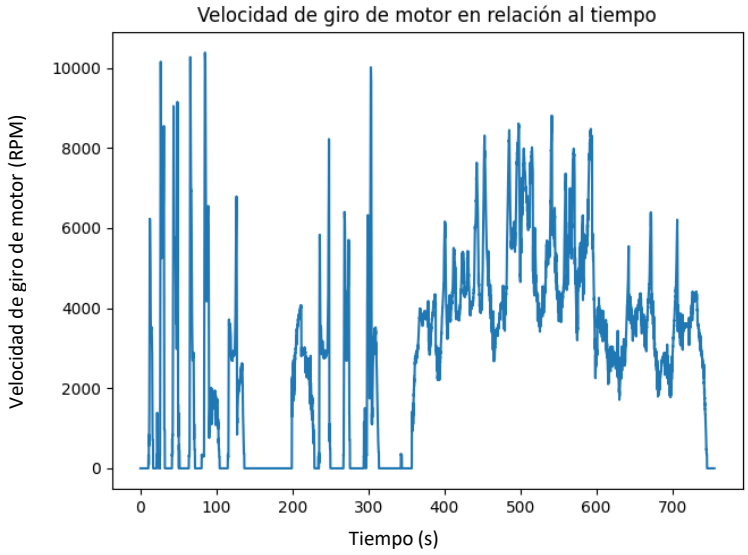


Ilustración 23 Gráfica del escenario uno, RPM - Tiempo

En la primera mitad de la gráfica se puede observar un comportamiento inusual que se debe a que el piloto está alternando entre estado de ralentí y acelerones que llevan el motor al corte. La finalidad de este comportamiento es forzar al alternador a cargar la batería sin sobrecalentar el motor y al mismo tiempo permitir a los ingenieros de motor ajustar el mapeado de este en todas las regiones de funcionamiento. En esta primera mitad también se aprecian claramente dos valles en cero RPM que se corresponden a dos momentos donde el piloto mantuvo el motor apagado por ordenes del ingeniero de pista para permitir al motor enfriarse.

En la segunda mitad se puede apreciar el comportamiento del motor en una tanda real, donde el piloto hace trabajar el motor en el rango de revoluciones óptimo (entre 5000 y 8000 RPM).

9.5.2. Segundo escenario

En este segundo escenario, el ingeniero de pista quiere analizar el tipo de pilotaje del piloto. Para ello compara la entrada de acelerador o TPS que el piloto requiere al motor en cada instante de tiempo con el régimen de revoluciones en el que está trabajando el motor. Si porcentaje de apertura del TPS es muy elevado (>80%) y el rango de revoluciones se encuentra por debajo de las 5000 RPM significará que el piloto está usando una relación de transmisión incorrecta y que podría ganar tiempo reduciendo una o varias marchas en esa sección del circuito.

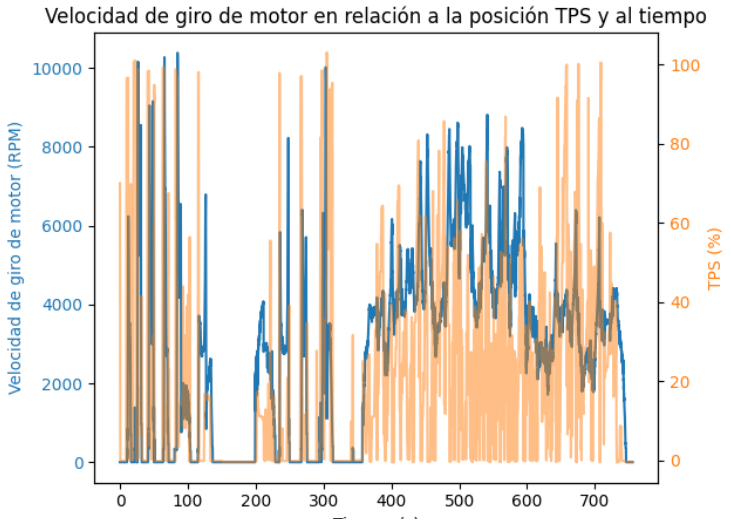


Ilustración 24 Gráfica del escenario dos, comparativa entre RPM y TPS en función del tiempo

Para ello se ha utilizado un modelo gráfica de líneas continuas y tres ejes de datos. Para que una representación no tape a la otra se ha decidido suavizar la opacidad de la línea del TPS un 20%.

9.6. Interfaz de usuario de la aplicación

Inicialmente todas las interacciones con el usuario se realizan mediante la consola del IDE donde se ejecuta el programa. No obstante, se ha estudiado la posibilidad de integrar el funcionamiento de este a una interfaz de usuario mediante la programación de una aplicación.

De esta manera tendríamos dos partes del código bien diferenciadas, la parte de inteligencia de la aplicación o *backend* que reúne todo lo desarrollado en este proyecto y la parte visual o *frontend* con la que interactuará el usuario final.

Mediante la herramienta de diseño de interfaz colaborativa Figma[21] se han diseñado tres propuestas de pantalla para la aplicación.

9.6.1. Pantalla inicial

En esta primera pantalla se puede observar una plantilla con los colores y logo corporativos del equipo EUSS Motorsport junto con un formulario inicial donde el usuario puede:

- Escoger el archivo a procesar desde su ordenador.
- Introducir el nombre del piloto que estaba en el monoplaza en el momento que se obtuvieron los datos.
- Introducir la fecha en la que se recogieron los datos.
- Introducir el número de *run* o vuelta en la que se recogieron los datos.
- Introducir la hora de inicio del *run*.

[21] https://www.figma.com/

Ilustración 25 Pantalla de inicio de la interfaz de usuario

Se ha optado por pedir estos datos al usuario en vez de tomarlos directamente del nombre del archivo porque el ingeniero de pista los recoge en el *run plan*[22] cada vez que el coche sale a pista. El motivo por el que se ha decidido no depender de los datos que la ECU genera automáticamente en el nombre del archivo es porque los datos de fecha y hora tienden a desconfigurarse cuando el coche se queda sin batería o se cambia el ordenador con el que se activa la obtención de datos.

9.6.2. Pantalla de selección de datos a procesar

En esta segunda pantalla se siguen utilizando los colores y logos corporativos del equipo EUSS Motorsport. Su funcionalidad se basa en una lista de valores recogidos por la ECU donde el usuario puede seleccionar solo aquellos valores que quiera subir a la base de datos para su posterior análisis. De esta manera, lo que se busca es tener una base de datos no sobrecargada, es decir, no almacenar datos que no se vayan a utilizar.

[22] Plan de test donde se organiza la planificación del test con los pilotos del día, los datos a obtener, y qué se va a mirar una vez el coche entre en box. También se apuntan datos referentes a la meteorología para cada *run* y el tiempo utilizado en la puesta a punto del monoplaza.

Ilustración 26 Pantalla de selección de los datos a procesar

9.6.3. Pantalla de análisis

En esta última pantalla el usuario puede escoger qué tipo de gráfica desea visualizar: lineal, por puntos o por barras, y entrar por teclado qué valores quiere ver. En este ejemplo se puede ver como se representan tres valores, no obstante con dos ya se podría generar la gráfica.

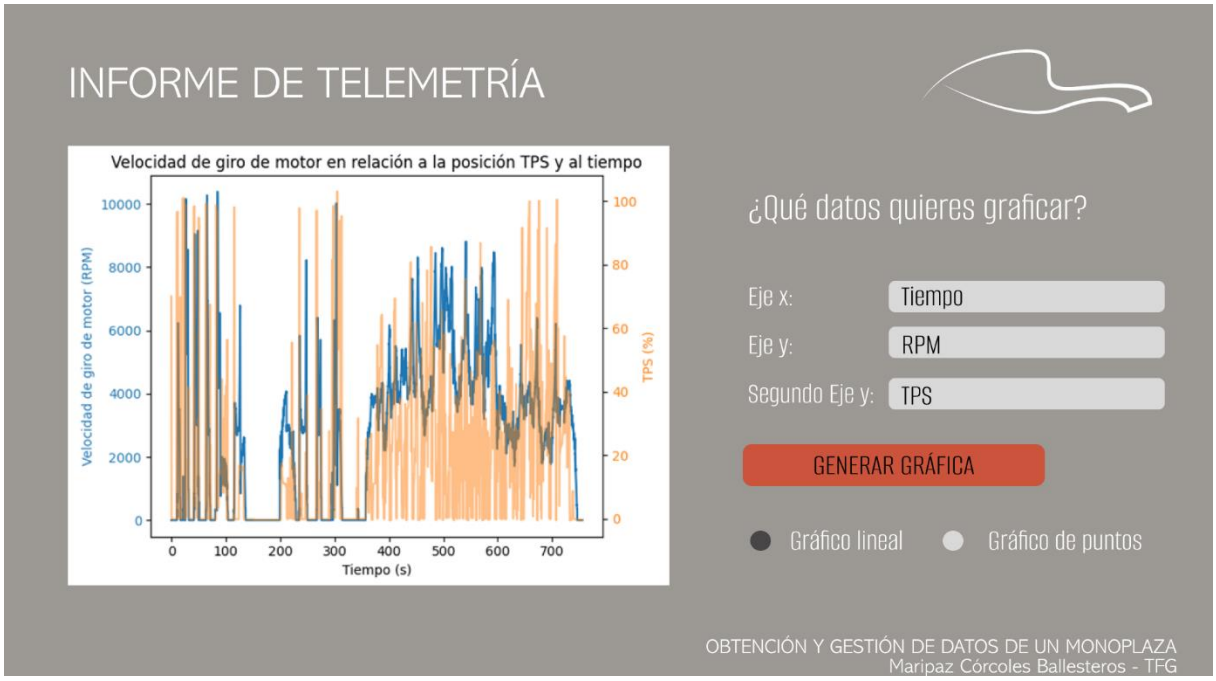


Ilustración 27 Pantalla de representación gráfica de los datos escogidos por el usuario

9.7. Implementación del sistema en el monoplaza

Como ya se ha explicado anteriormente, no ha sido posible hacer la instalación del sistema en el monoplaza, todo ha sido probado de forma externa y queda pendiente la implementación en el nuevo modelo por parte de los integrantes del departamento de electrónica de la temporada 2022 – 2023.

Por este motivo se ha querido representar cómo sería esta implementación utilizando el *software* de diseño 3D SolidWorks.

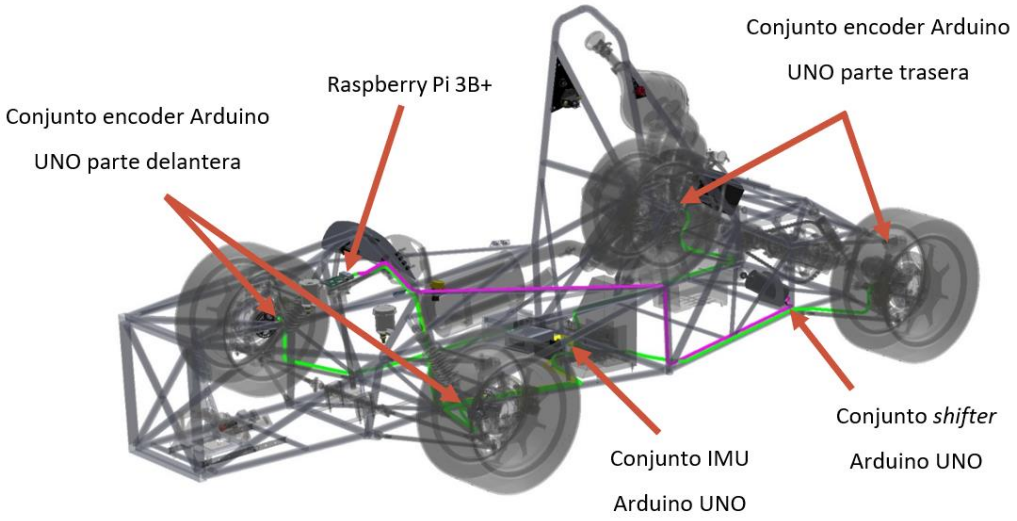


Ilustración 28 Implementación del sistema en el monoplaza EM-03

En la ilustración anterior se pueden ver los principales elementos que conforman el sistema electrónico del coche ya mencionados en el capítulo 3. A estos elementos se les ha sumado el sistema de adquisición de datos resultante de este proyecto, que consiste en una Raspberry Pi situada en la columna de transmisión, un sensor IMU, cuatro sensores encoder situados en las manguetas de suspensión, un sensor en el *shifter* para controlar los cambios en la caja de cambios y tres Arduino UNO, uno por conjunto de sensores.

Se ha intentado ubicar los Arduino lo más cercano al sensor para minimizar la pérdida de datos. En el caso de los encoder, al haber cuatro, se ha decidido ubicar en el centro geométrico del monoplaza.

10. Conclusiones

Tras la finalización de este proyecto, se ha conseguido desarrollar un sistema de obtención y gestión de datos automatizado donde los datos generados por la ECU y los sensores son leídos y enviados a una base de datos con solo correr un *script*. Esto facilita tanto la forma de proceder con el análisis de los datos en pista como el proceso de segurización de los mismos al tenerlos almacenados en el *cloud*.

El hecho de poder ver las representaciones gráficas en pista ayuda a cumplir el objetivo de reducción de incidencias debido a que los problemas del monoplaza pueden verse durante el transcurso del test o de la competición sin tener que esperar a acabar el día.

Al tener los datos centralizados en la Raspberry, la comunicación entre los diferentes elementos es más fluida que en temporadas anteriores, donde todo se almacenaba por separado. Actualmente, se podrían analizar de forma conjunta los datos de la ECU y los diferentes sensores. Gracias a esta centralización, se han asentado las bases para una transmisión de telemetría en tiempo real de los datos mediante el protocolo MQTT.

Con este sistema se garantiza la integración del tren de potencia eléctrico y el procesado de datos provenientes de la nueva ECU. Esto se debe a que la forma de exportar los datos será similar, estarán estructurados por columnas, y la extensión del archivo no supondrá un problema porque el primer paso es la conversión del archivo nativo a un formato que Python puede leer.

Al no poder implementar el sistema de comunicaciones *CAN bus* por lo mencionado anteriormente, no ha sido posible crear un sistema donde se puedan incorporar nuevos sensores sin que ello implique una gran complejidad, ni preparar el monoplaza para el salto al coche autónomo integrando el sensor LiDAR, por lo que estos dos puntos quedan pendientes para un futuro.

En resumen, teniendo en cuenta las condiciones del proyecto, se han alcanzado todos los objetivos que se podían cumplir. Quedará pendiente para el futuro del equipo hacer las implementaciones necesarias para poder cumplir los dos objetivos restantes relacionados con el sistema *CAN bus* y la instalación del sensor LiDAR para la conducción autónoma.

11. Bibliografía

[1] «Formula SAE,» [En línea]. Available: https://www.fsaeonline.com/. [Último acceso: 14 Noviembre 2021].

[2] S. International, «Baja SAE,» [En línea]. Available: https://www.bajasae.net/. [Último acceso: 7 Enero 2022].

[3] I. o. M. Engineers, «Institution of Mechanical Engineers,» [En línea]. Available: https://www.imeche.org/. [Último acceso: 7 Enero 2022].

[4] Applus IDIADA, «Applus IDIADA,» [En línea]. Available: https://www.applusidiada.com/global/es/. [Último acceso: 13 Enero 2022].

[5] E. b. d. -. E. Motorsport, «EUSS Motorsport,» [En línea]. Available: https://www.eussmotorsport.com/es/home. [Último acceso: 7 Enero 2022].

[6] E. U. S. d. Sarrià, «EUSS,» [En línea]. Available: https://www.euss.cat/. [Último acceso: 7 Enero 2022].

[7] F. S. Germany. [En línea]. Available: https://www.formulastudent.de/fileadmin/user_upload/all/2022/rules/FS-Rules_2022_v1.0.pdf. [Último acceso: 24 Noviembre 2021].

[8] «Car-tec,» [En línea]. Available: https://www.car-tec.es/blog/como-funciona-una-ecu/. [Último acceso: 7 Enero 2022].

[9] C. Valladares, «Petrolhead Garage,» [En línea]. Available: https://petrolheadgarage.com/cursos-automocion/sensor-de-temperatura-de-refrigerante-ect/. [Último acceso: 7 Enero 2022].

[10] C. Valladares, «Petrolhead Garage,» [En línea]. Available: https://petrolheadgarage.com/cursos-automocion/que-es-el-sensor-de-oxigeno-o-sonda-lambda/. [Último acceso: 7 Enero 2022].

[11] R. Fernández, «Petrolhead Garage,» [En línea]. Available: https://petrolheadgarage.com/cursos-automocion/como-funciona-ckp/. [Último acceso: 7 Enero 2022].

[12] N. A. Parts, «NAPA Know How Blog,» 15 Octubre 2020. [En línea]. Available: https://knowhow.napaonline.com/what-does-a-camshaft-synchronizer-assembly-do-for-your-engine/. [Último acceso: 7 Enero 2022].

[13] B. Lampe, «NAPA Know How Blog,» 30 Marzo 2020. [En línea]. Available: https://knowhow.napaonline.com/what-is-a-camshaft-position-sensor-and-what-does-it-do/. [Último acceso: 7 Enero 2022].

[14] M. C. Keegan, «NAPA Know How Blog,» 08 Enero 2020. [En línea]. Available: https://knowhow.napaonline.com/what-is-a-throttle-position-sensor/. [Último acceso: 7 Enero 2022].

[15] canalMOTOR, «Mapfre Motor,» 25 Mayo 2020. [En línea]. Available: https://www.motor.mapfre.es/consejos-practicos/consejos-de-mantenimiento/sensor-presion-aceite/. [Último acceso: 7 Enero 2022].

[16] FSAE, «FSAE Engineeering Columbia,» [En línea]. Available: https://fsae.engineering.columbia.edu/FSAE_Rules_2021_V1_files/part290.htm. [Último acceso: 7 Enero 2022].

[17] Ingeniería Mecafenix, «Ingeniería Mecafenix,» 23 Julio 2018. [En línea]. Available: https://www.ingmecafenix.com/automatizacion/sensor-inercial/. [Último acceso: 7 Enero 2022].

[18] D. d. e. d. E. Motorsport, *OTH_IN_Comparación de sistemas de comunicación,* 2019.

[19] Paessler, «Paessler the monitoring experts,» 2022. [En línea]. Available: https://www.paessler.com/es/it-explained/mqtt. [Último acceso: 17 Abril 2022].

[20] Germany, Formula Student, «Formula Student Rules,» 2022. [En línea]. Available: https://formulastudent.de/fileadmin/user_upload/all/2022/rules/FS-Rules_2022_v0.9.pdf. [Último acceso: 9 Febrero 2022].

[21] T. redac, «DataScientest,» 27 Mayo 2022. [En línea]. Available: https://datascientest.com/es/que-es-un-dataframe. [Último acceso: 11 Junio 2022].