



**POLITÉCNICA**



# Monitorización de vehículos

---

Proyecto de Fin de Grado

Ingeniería de Computadores

Autor: Luis Diego Castillo Espejo

Tutor: Vicente A. García Alcántara

Cotutor: Alberto Cruz Ruiz

# ÍNDICE DE CONTENIDO

Objetivo	11
Resumen	13
Abstract	15
Motivación	17
Capítulo 1: Marco del problema	19
Capítulo 2: Estado del arte	21
2.1. Smartceta	22
2.2. Diseño y desarrollo de un sistema de diagnóstico de vehículos	23
2.3. Control de dispositivos acoplados a un automóvil	24
Capítulo 3: Metodología	27
Capítulo 4: Análisis de requisitos	29
4.1 Requisitos funcionales	29
4.2 Requisitos no funcionales	30
Capítulo 5: Diseño e implementación	31
5.1 Arquitectura del sistema	31
5.2 Funcionamiento del sistema	32
5.3 Consumo de energía	33
Capítulo 6: Recursos hardware	35
6.1 ESP-32	35

---

## Índice de contenido

---

6.2 ELM327 mini	37
6.3 GPS NEO-6M	40
Capítulo 7: Recursos software	43
7.1 MicroPython	43
7.2 ThingsBoard	44
7.3 Protocolo MQTT	45
7.4 Bluetooth Low Energy	46
7.5 Telegram	50
Capítulo 8: Conectividad a Internet	51
8.1 SIM800L GSM GPRS	51
8.2 SIM7600E-H	52
8.3 LORA SX1278	53
8.4 Wisol SFM10R1	54
Capítulo 9: Configuraciones del sistema	57
9.1 Configuración de ESP32	57
9.2 Comunicación con ELM327	57
9.3 Comunicación con GPS NEO-6M	60
9.4 Configuración ThingsBoard	61
9.5 Configuración Telegram	64
Capítulo 10: Pruebas unitarias	67

---

10.1 vehículo detenido	67
10.2 Lectura de OBD-II	69
10.3 Transición de estado del vehículo	70
10.4 Distancia recorrida	72
10.5 Visualización de datos	73
Capítulo 11: Pruebas del sistema	75
11.1 Problemas encontrados	79
Capítulo 12: Planificación del proyecto	81
12.1 Planificación temporal	81
12.2 Planificación económica	82
Capítulo 13: Impacto social y medioambiental	85
Capítulo 14: Conclusiones	87
Capítulo 15: Líneas futuras	89
Apéndice A. Referencias	91
Apéndice B. Glosario	95
Apéndice C: Material entregado a Moodle y código	103
Apéndice D. Imágenes del proyecto	105

---



# ÍNDICE DE FIGURAS

Figura 1: Aplicación Mercedes Me Connect	22
Figura 2 : Arquitectura Smartceta	23
Figura 3 : Arquitectura del proyecto	31
Figura 4 : Diagrama de flujo	32
Figura 5 : ESP32 pinout	36
Figura 6 : Puerto OBD2	38
Figura 7 : Modos OBD-II	40
Figura 8 : GPS NEO-6M	41
Figura 9 : Dashboard	44
Figura 10 : Protocolo MQTT	45
Figura 11 : Dispositivo central BLE	47
Figura 12 : Capas BLE	48
Figura 13 : Capa GATT BLE	49
Figura 14 : SIM800L GSM GPRS	51
Figura 15 : SIM7600E-H	52
Figura 16 : LORA SX1278	53
Figura 17 : Wisol SFM10R1	54
Figura 18 : Wisol SFM10R1	54

---

## Índice de figuras

---

Figura 19 : Servicios BLE de ELM327	58
Figura 20 : Trama GPS	60
Figura 21 : Dispositivo en Thingsboard	61
Figura 22 : Cadena de reglas en Thingsboard	62
Figura 23 : Comportamiento temperatura alta	62
Figura 24 : Función mensaje Telegram	63
Figura 25 : Rest API Telegram	63
Figura 26 : Cadena de reglas raíz	64
Figura 27 : Bot de Telegram	64
Figura 28 : Prueba Vehículo detenido	68
Figura 29 : Medición de error GPS	68
Figura 30 : Ejemplo de RPM	69
Figura 31 : Ejemplo de tiempos RPM	70
Figura 32 : Cambio de estado del vehículo	71
Figura 33 : Distancia Google Maps	73
Figura 34 : Función para calcular la distancia	73
Figura 35 : Telemetría en ThingsBoard	74
Figura 36 : Gráfica de datos en ThingsBoard	75
Figura 37 : Gráfica distancia recorrida	76

---

Figura 38 : Gráfica posición del acelerador	76
Figura 39 : Gráfica RPM	77
Figura 40 : Trayecto recorrido	77
Figura 41 : Avisos Telegram	78
Figura 42 : Diagrama de Gantt	81



## ÍNDICE DE TABLAS

Tabla 1 : Características ESP32	36
Tabla 2 : Características ELM327	37
Tabla 3 : Características GPS-NEO6M	41
Tabla 4 : Diferencias entre BLE y Bluetooth clásico	50
Tabla 5 : Principales tecnologías de comunicación	50
Tabla 6 : Precio de los componentes del sistema	82



# OBJETIVO

El objetivo de este proyecto es desarrollar un sistema IoT (*Internet of Things*) capaz de monitorizar los distintos sensores integrados de un vehículo. El sistema también contará con módulo GPS para obtener la geolocalización en tiempo real. El dispositivo debe ser capaz de recopilar datos de los sensores del automóvil, como la velocidad, la temperatura y la presión del colector de admisión entre otros. Los datos recolectados serán enviados a la plataforma ThingsBoard para su visualización y análisis. Ello probablemente requiera de la integración de varias tecnologías, incluyendo un dispositivo ELM327 para establecer la comunicación con la Unidad de Control Electrónico del vehículo mediante el protocolo OBD-II (*On-Board Diagnostics*), un microcontrolador ESP-32 programado en MicroPython, sensores integrados del vehículo, tecnologías de comunicación como el Bluetooth Low Energy y la plataforma IoT llamada ThingsBoard. Dado que el dispositivo se alimentará mediante la toma *USB* del automóvil, se deben tener en cuenta las consideraciones de eficiencia energética, así como la de diseño, seguridad, y la facilidad de uso. Además, uno de los principales fundamentos es el desarrollo de un sistema asequible al alcance de la mayoría de la población.



# RESUMEN

En este proyecto se desarrolla, programa e implementa un sistema para la adquisición de datos de los sensores de vehículos, los datos se visualizarán utilizando una plataforma IoT (*Internet of Things*). El proyecto se estructura en múltiples etapas, que abarcan desde el análisis de los requisitos hasta el análisis de los datos recolectados.

El componente central del sistema es un microcontrolador ESP32 programado en MicroPython, el cual se combina con el módulo GPS NEO-6M mediante una comunicación UART para obtener y procesar los datos de localización GPS. El sistema también cuenta con un dispositivo ELM327, el cual es capaz de obtener datos de los sensores del vehículo, este dispositivo se comunicará con el microcontrolador ESP32 a través Bluetooth Low Energy.

Los datos se envían a ThingsBoard, una plataforma IoT de código abierto donde se procesan para su posterior visualización, además se envían notificaciones al usuario mediante la aplicación de mensajería instantánea Telegram.

El sistema entra en funcionamiento cuando se suministra energía, se inicializa y se ejecuta el *firmware* correspondiente. En caso de que el vehículo esté detenido, el ciclo de trabajo se enfocará en verificar la ubicación GPS y activar el modo de hibernación profunda (*deepsleep*) para ahorrar energía. Por otro lado, si el vehículo está en movimiento, se leerán los datos de los sensores y se enviarán a ThingsBoard para su procesamiento.

Finalmente, se llevó a cabo un análisis exhaustivo y una verificación de los datos obtenidos tanto en las pruebas unitarias como en las pruebas del sistema.



## ABSTRACT

In this project, a system is developed, programmed, and implemented for acquiring data from vehicle sensors, which will be visualized using an IoT (Internet of Things) platform. The project is structured into multiple stages, ranging from requirements analysis to analysis of the collected data.

The central component of the system is an ESP32 microcontroller programmed in MicroPython, which is combined with the NEO-6M GPS module through UART communication to obtain and process GPS location data. The system also includes an ELM327 device, capable of retrieving data from vehicle sensors, which communicates with the ESP32 microcontroller via Bluetooth Low Energy.

The data is sent to ThingsBoard, an open-source IoT platform, where it is processed for further visualization. Additionally, user notifications are sent through the Telegram instant messaging application.

The system is activated when power is supplied, initialized, and the corresponding firmware is executed. If the vehicle is stationary, the system focuses on verifying the GPS location and activating the deep sleep mode to save energy. On the other hand, if the vehicle is in motion, sensor data is read and sent to ThingsBoard for processing.

Finally, a comprehensive analysis and verification of the obtained data were conducted, both in the unit tests and system tests.



# MOTIVACIÓN

En la actualidad, la tecnología de Internet de las Cosas (IoT, por sus siglas en inglés) ha revolucionado la forma en que se interactúa con el mundo circundante. La integración de sensores, dispositivos electrónicos y plataformas en línea ha permitido el monitoreo y control de objetos cotidianos de una manera nunca vista. En el contexto de los vehículos, los dispositivos OBD-II se han convertido en una herramienta indispensable para la monitorización del rendimiento del motor y la identificación de problemas (AUTODOC, 2023).

Debido a la necesidad de mejorar la seguridad y eficiencia de los vehículos, cada vez más, la industria automotriz se está enfocando en la incorporación de Internet de las Cosas (IoT) en sus vehículos. El desarrollo de un sistema que permita obtener tanto la información en tiempo real sobre la ubicación y estado del vehículo, como del rendimiento y consumo de combustible, puede mejorar no solo la seguridad del vehículo, al detectar posibles problemas mecánicos o prevenir robos, sino también ayudar a combatir el cambio climático, ya que al tener datos del consumo en tiempo real, el conductor tendrá la posibilidad de adaptar su conducción para reducir el consumo de combustible y, por tanto, reducir las emisiones de gases contaminantes (HOMYHUB, 2023).



# CAPÍTULO 1: MARCO DEL PROBLEMA

En un sistema de monitorización de automóviles con tecnología IoT se ha de tener en cuenta varias cuestiones: La comunicación entre los distintos dispositivos que conforman el sistema ya sea mediante cables físicos o de forma inalámbrica. La obtención de datos en tiempo real de los diferentes sensores y sistemas del automóvil es esencial para realizar un monitoreo efectivo. Sin embargo, acceder y procesar estos datos de manera rápida y confiable puede ser complicado debido a la complejidad de la arquitectura electrónica de los vehículos. La transmisión de información se debe de hacer de la forma segura, debido a que el sistema de monitorización del automóvil se comunica con diferentes componentes y sistemas del vehículo, es fundamental garantizar la protección de la información. La integridad de los datos transmitidos, así como la confidencialidad de la información del vehículo y del usuario, deben ser consideraciones primordiales.

A su vez la escalabilidad en los sistemas de monitorización de vehículos IoT puede ser difícil debido a la gran cantidad de datos que se generan y la necesidad de procesamiento y almacenamiento. La implementación de un sistema escalable puede requerir la utilización de tecnologías de nube y la integración con servicios de terceros generando costos adicionales.

Abordar estos desafíos y limitaciones permitirá el desarrollo de un sistema de monitorización del automóvil eficiente y confiable, que brinde a los conductores información precisa sobre el estado del vehículo, contribuyendo así a la seguridad y el mantenimiento adecuado del automóvil.



## CAPÍTULO 2: ESTADO DEL ARTE

El Internet de las Cosas (IoT) es un concepto tecnológico que se refiere a la interconexión de dispositivos y objetos cotidianos a través de Internet, lo que les permite recopilar, transmitir y analizar datos en tiempo real para mejorar su funcionamiento y rendimiento, utiliza 3 tecnologías principalmente: las tecnologías de comunicación inalámbrica, los sistemas microelectromecánicos (*MEMS*) y los microservicios de Internet. En 1990 apareció el primer dispositivo conectado a internet, John Romkey y Simon Hacket, diseñaron una tostadora con conectividad a Internet a través del protocolo TCP/IP, con la capacidad de determinar su encendido, apagado y configuración del tiempo de “tostado” desde cualquier ordenador (Santos, 2023).

En los últimos años, el *IoT* ha evolucionado rápidamente gracias al aumento de dispositivos conectados, la mejora de la conectividad inalámbrica y la adopción de tecnologías de Inteligencia Artificial y análisis de datos. Actualmente la industria automotriz está adoptando cada vez más tecnologías del Internet de las Cosas para mejorar la seguridad y eficiencia de los vehículos, así como para optimizar la gestión de flotas, *BMW CarData* (BMW CarData , 2023) y *Mercedes Me Connect* (Mercedes Me Connect , 2023) son un ejemplo. Ambos son plataformas de acceso seguro que permiten ver los principales datos de vehículos en cualquier momento y además permite a los propietarios compartir datos de telemetría de sus vehículos con terceros de confianza, como talleres mecánicos, compañías de seguros y proveedores de servicios de movilidad. Las plataformas se basan en una arquitectura de nube y utilizan tecnología de seguridad de última generación para proteger los datos del usuario.

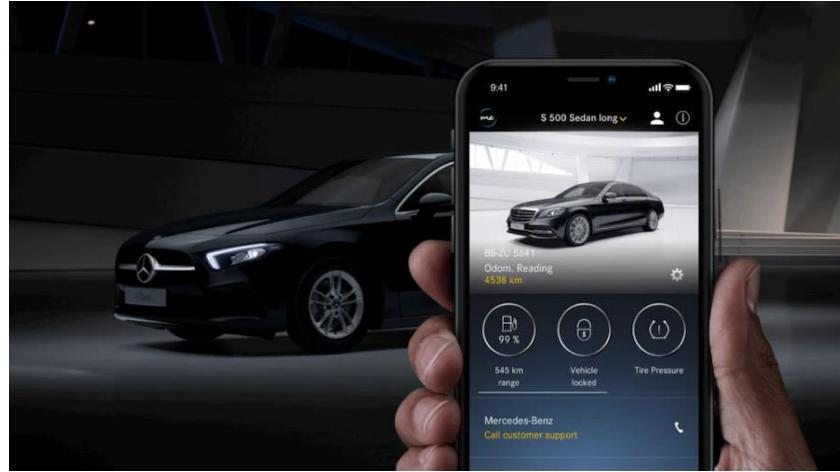


Figura 1: Aplicación Mercedes Me Connect

A continuación, se presentará una breve descripción de trabajos relacionados con IoT y el protocolo OBD-II (*On-Board Diagnostics II*) que es un sistema estandarizado obligatorio para todos los vehículos matriculados a partir de 1996, utilizado para el monitoreo y diagnóstico de diversos aspectos relacionados con su funcionamiento.

## 2.1. Smartceta, sistema de medición de valores básicos de una planta utilizando IoT

Este trabajo se centra en desarrollar un sistema de monitorización de una planta, haciendo uso de sensores y actuadores, el sistema obtiene información sobre la tierra y el ambiente vegetal, para posteriormente a través de una conexión a internet enviar estos datos en tiempo real a la aplicación ThingsBoard y Telegram (Santos-Olmo, 2022).

Herramientas utilizadas:

- ESP32: Es un microcontrolador de bajo costo y bajo consumo de energía que cuenta con conectividad WiFi y Bluetooth, es utilizado como unidad de control del sistema.
- Arduino IDE: Entorno basado en C++ utilizado para programar el comportamiento de la unidad de control.
- ThingsBoard: Plataforma utilizada para la gestión y visualización de los datos obtenidos.

En la siguiente imagen se muestra la arquitectura seguida en el proyecto.

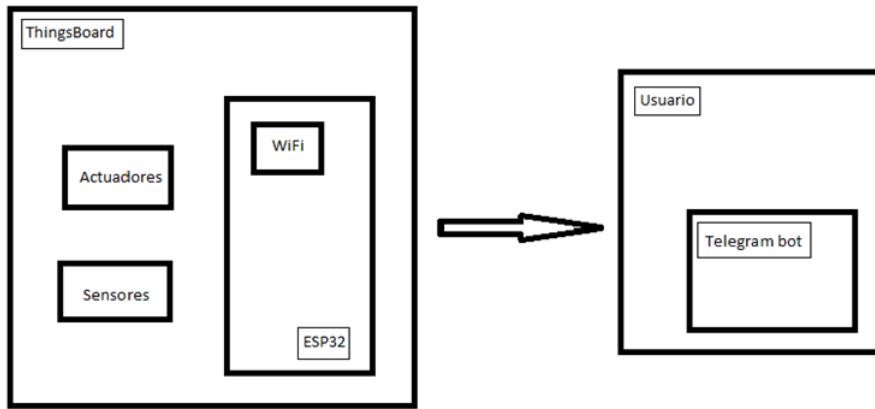


Figura 2 : Arquitectura Smartceta

En la conclusión de este trabajo, cabe remarcar la gran variedad de funcionalidades que puedes integrar con ThingsBoard para el tratamiento y visualización de datos, así como del uso de Telegram para alertar al usuario de una forma inmediata.

## 2.2. Diseño y desarrollo de un sistema de diagnóstico de vehículos basado en módulos empotrados de bajo coste y en dispositivos móviles inteligentes

En este proyecto se hace uso de hardware y software libre para el diseño y desarrollo de un sistema capaz de leer los parámetros de vehículos y, además tener la capacidad de borrar averías de la Unidad de Mando (*ECU*) (Mula, 2021).

Las herramientas utilizadas son las siguientes:

- Android: Es el entorno de desarrollo utilizado en este proyecto.
- Arduino UNO: La placa Arduino UNO está basado en el microcontrolador ATMEGA328 y es el componente principal del sistema en este proyecto.
- Bluetooth HC-05: Es un módulo de comunicación inalámbrica que utiliza tecnología Bluetooth V2, dicho dispositivo permite la comunicación inalámbrica entre el Arduino UNO y un teléfono móvil.

- *OBD-II* Uart Hookup guide: Está formado por dos componentes, ELM327 y MCP2551 que permiten al usuario acceder a los protocolos *CAN (Control Area Network)* y *OBD-II*.

La conclusión principal de este trabajo es el uso de OBD-II y estudio de conocimientos generales para interaccionar con la Unidad de Control del Motor (*ECU*).

### **2.3. Desarrollo Android para el control de dispositivos acoplados a un automóvil**

El propósito central de este proyecto consiste en la elaboración de una aplicación para dispositivos Android, que permita la visualización de los datos adquiridos tanto de los sensores internos del vehículo como de sensores ajenos a él. Se utiliza un sistema gestor de base de datos PostgreSQL y una Raspberry Pi incorporada en el vehículo (Hoyas, 2015).

Las herramientas utilizadas son:

- Android studio: Es un entorno de desarrollo integrado (*IDE Integrated Development Environment*) para Android.
- Python: Es un lenguaje de programación bastante utilizado, gracias a su fácil sintaxis.
- Telegram: Es un servicio de mensajería instantánea que ofrece una *API\** (*Application Programming Interfaces*) para diferentes plataformas, lo que permite a los usuarios poder crear sus propios clientes de Telegram.
- Raspberry Pi B+: Es el módulo principal del sistema, mediante el cual se procesan los datos obtenidos del ELM327, para enviarlos posteriormente a la base de datos.
- ELM327: Es un dispositivo utilizado para obtener los datos directamente desde el vehículo.
- Sensor MQ7: Es un sensor de monóxido de carbono de alta sensibilidad y respuesta rápida.

- Sensor DHT-11: Es un sensor capaz de medir de manera fiable la temperatura y humedad del ambiente.

De este proyecto se tiene que destacar el ELM327, ya que es un dispositivo de bajo coste con el que se puede establecer una comunicación con el automóvil. Además, también se debe tener presente la posibilidad de incorporar nuevos sensores que nos proporcionen datos útiles y complementarios los datos obtenidos a través de los sensores propios del coche.

En general, los trabajos previos han sentado las bases teóricas y prácticas necesarias para el desarrollo de un dispositivo que pueda leer los sensores de vehículos a través de un dispositivo OBD-II y enviar los datos a la plataforma ThingsBoard.



## CAPÍTULO 3: METODOLOGÍA

Dada la funcionalidad del proyecto, el cual consiste en la lectura de los sensores de vehículos, la metodología empleada empieza por el análisis de requisitos, donde se identificarán los requisitos funcionales y no funcionales, una vez realizado el análisis de requisitos, proseguirá el estudio y análisis de los distintos dispositivos y protocolos de comunicación ya implementados, posteriormente se describirán los fundamentos teóricos y prácticos detrás de las tecnologías utilizadas, para asegurar el uso de una documentación fiable, se ha usado la base de datos de patentes y solicitud de patentes de *Google Patents* (Google Patents, 2023). Además, del uso del Archivo Digital de la Universidad Politécnica de Madrid (Archivo Digital, 2023).

Posteriormente, se iniciará con el diseño y la implementación tanto software como hardware del proyecto, en el cual se definirá la arquitectura del sistema y se detallará el flujo de trabajo a seguir teniendo en cuenta los requisitos necesarios.

La siguiente etapa será la de pruebas, el objetivo principal es verificar que el software desarrollado cumpla con los requisitos establecidos y funcione de acuerdo con el diseño previsto. Esta fase implica la realización de diferentes tipos de pruebas, como pruebas unitarias, pruebas de integración, pruebas de sistema y pruebas de aceptación. Durante esta fase, se diseñan casos de prueba y se ejecutan para identificar y corregir posibles errores o defectos en el software. Los resultados de las pruebas se registran y se utilizan para evaluar la calidad del software antes de avanzar hacia la fase de implantación.

La etapa de implantación o despliegue es una etapa crucial en el desarrollo de software, en la cual el software desarrollado se instala y pone en funcionamiento en el entorno de producción. Esta fase implica la preparación y configuración del entorno de producción, la instalación del software en los sistemas objetivo y la realización de pruebas adicionales para asegurarse de que todo funcione correctamente en el entorno real.

---

## Capítulo 3: Metodología

---

En conclusión, se ha seguido una metodología en cascada, cuyo enfoque es secuencial y estructurado en una serie de fases bien definidas. A través del método en cascada, se puede lograr una planificación rigurosa, un diseño detallado y una implementación gradual del software. Además, este enfoque permite una mejor gestión del proyecto al dividirlo en etapas claramente definidas, lo que facilita la asignación de recursos y el seguimiento del progreso.

# CAPÍTULO 4: ANÁLISIS DE REQUISITOS

En esta sección, se abordarán los puntos clave relacionados con el análisis de requisitos para el proyecto en cuestión. Se establecerán las necesidades que el sistema debe cumplir para lograr una implementación exitosa.

## 4.1 Requisitos funcionales

Los requisitos funcionales son las acciones específicas que un sistema o software debe ser capaz de realizar. Estos requisitos son fundamentales para garantizar que cumpla con las necesidades y expectativas de los usuarios.

1. Recopilación de datos de los sensores del automóvil: El sistema debe ser capaz de recopilar datos de los diversos sensores presentes en el automóvil. Esto implica la capacidad de extraer información relevante, como la velocidad, la temperatura del refrigerante, las revoluciones por minutos, entre otros, para su posterior procesamiento y análisis.
2. Envío de datos a la plataforma ThingsBoard: Los datos recopilados deberán ser enviados a la plataforma ThingsBoard para su visualización y gestión. Esto requiere una integración adecuada entre el sistema y la plataforma, asegurando una transmisión confiable y segura de los datos a través de protocolos de comunicación compatibles.
3. Cálculo de la distancia recorrida: El sistema debe tener la capacidad de calcular y registrar la distancia recorrida por el vehículo.
4. Avisos al usuario: El sistema debe ser capaz de enviar notificaciones y avisos al teléfono móvil del usuario. Estos avisos pueden incluir información relevante sobre el trayecto realizado y alertas de mantenimiento.
5. Toma USB del automóvil: El sistema debe ser compatible con la toma USB presente en el automóvil. Esto permitirá obtener la alimentación de energía necesaria para el funcionamiento del sistema y garantizar una conexión estable y confiable.

6. Coste económico: Un aspecto importante a considerar es la viabilidad económica del sistema. Se debe tener en cuenta la disponibilidad de recursos financieros y asegurarse de que la implementación del proyecto se realice dentro de un presupuesto razonable. Se buscarán soluciones rentables sin comprometer la calidad y funcionalidad del sistema.
7. Consumo de energía eficiente: Para garantizar un funcionamiento óptimo y reducir el impacto en la batería del automóvil, el sistema deberá ser diseñado de manera que su consumo de energía sea lo más eficiente posible. Se buscarán estrategias para minimizar el consumo energético sin comprometer el rendimiento y las funcionalidades requeridas.

## 4.2 Requisitos no funcionales

Los requisitos no funcionales son criterios que definen las características y atributos que el sistema debe cumplir, más allá de sus funcionalidades específicas. Estos requisitos se centran en aspectos como el rendimiento, la seguridad, la usabilidad, la disponibilidad, la escalabilidad y otros aspectos importantes para asegurar la calidad y el cumplimiento de los objetivos del sistema. A continuación, se presentan algunos ejemplos de requisitos no funcionales:

1. Rendimiento: El sistema debe ser capaz de procesar y responder a las solicitudes de manera eficiente, manteniendo un tiempo de respuesta rápido y una baja latencia en las operaciones críticas.
2. Seguridad: El sistema debe implementar medidas de seguridad adecuadas para proteger los datos y garantizar la confidencialidad, integridad y disponibilidad de la información.
3. Disponibilidad: El sistema debe estar disponible y accesible en todo momento, asegurando un alto grado de disponibilidad para los usuarios.
4. Escalabilidad: El sistema debe ser capaz de manejar un aumento en la carga de trabajo y adaptarse a cambios en los volúmenes de datos y usuarios sin comprometer su rendimiento y funcionalidad.

# CAPÍTULO 5: DISEÑO E IMPLEMENTACIÓN

En esta sección se transformarán los requisitos recopilados en soluciones tangibles y funcionales, además se analizarán los aspectos clave del diseño y la implementación.

## 5.1 Arquitectura del sistema

El sistema estará formado por un microcontrolador ESP32 como dispositivo principal, el cual estará programado en MicroPython que es un intérprete de Python enfocado a microcontroladores. Este microcontrolador se comunicará con el dispositivo ELM327 para la obtención y procesado de los datos provenientes del automóvil. Para tener una visualización de los datos se hará uso de la plataforma ThingsBoard, y Telegram proporcionará avisos al usuario mediante mensajes de texto, para que eso ocurra es necesario estar conectado a Internet, este caso se usará un teléfono móvil como punto de acceso WiFi. También se incorporará un módulo GPS, con el cual se obtendrán los datos de localización del vehículo en tiempo real.

A continuación, se muestra una imagen en la que se describe los distintos componentes que conforman el sistema y como se comunican entre ellos, ofreciendo así una visión global del sistema.

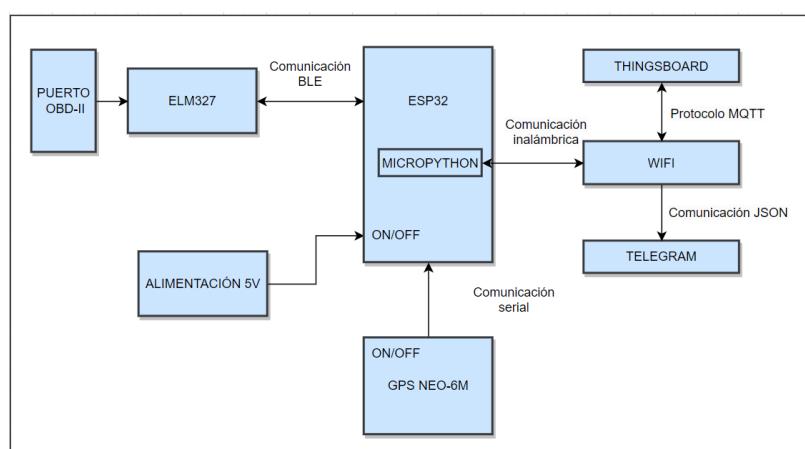


Figura 3 : Arquitectura del proyecto

## 5.2 Funcionamiento del sistema

El sistema tiene dos tipos de modos distintos, dependiendo de si el automóvil está en movimiento o apagado, el primer modo parte desde el estado donde el vehículo permanece detenido, entonces el ESP32 varía del estado activo donde obtendrá los datos GPS a modo *deepsleep* en un bucle continuo hasta que se detecte algún cambio en la posición GPS, con el fin de ahorrar energía. Una vez el sistema detecta que el vehículo está en movimiento, activa el resto de los dispositivos y a continuación, enviará un aviso a través de un chat de Telegram, mientras el vehículo este encendido, es decir, mientras se tenga constancia de la recepción de datos devueltos por los sensores del automóvil, dichos datos serán enviados de forma continua a la plataforma ThingsBoard para su posterior procesamiento y visualización, además de la geolocalización obtenida por el GPS.

Una vez el vehículo se apague, es decir, sus revoluciones por minuto dejen de emitir datos o desciendan a cero pasado un tiempo determinado, se enviará un aviso mediante el chat de Telegram y el sistema volverá al estado inicial.

La siguiente imagen muestra el diagrama de flujo que sigue el sistema, donde se visualiza de manera secuencial las etapas y procesos involucrados, además de mostrar la secuencia de acciones, decisiones y resultados.

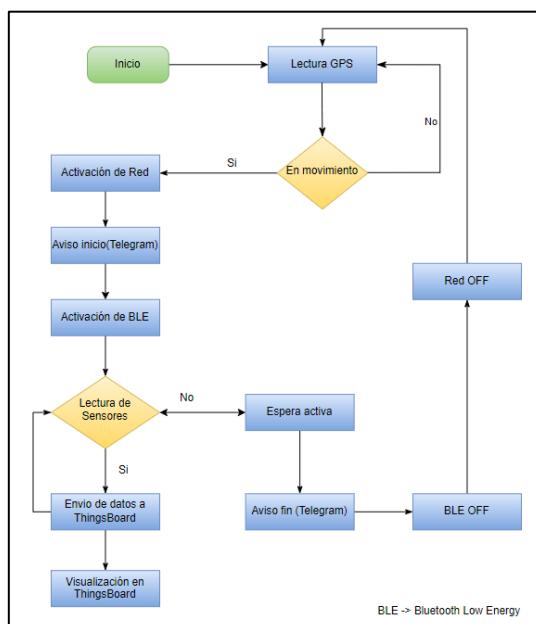


Figura 4 : Diagrama de flujo

### 5.3 Consumo de energía

Dado que es un requisito que el consumo de energía del sistema sea bajo, en este apartado se estudiará el consumo eléctrico del proyecto.

El consumo de energía varía según el comportamiento del sistema, durante el estado de reposo del vehículo, el sistema se encuentra en un modo de funcionamiento específico para minimizar el consumo de energía. Durante esta etapa, el microcontrolador alterna entre dos estados: el estado normal, en el cual se obtienen los datos del GPS, y el modo de ahorro de energía conocido como *deepsleep*. Durante el intervalo en que está en este modo el microcontrolador tiene un consumo de energía extremadamente bajo, de tan solo 5 $\mu$ A, lo cual contribuye significativamente al ahorro de energía. En comparación, en su estado normal de funcionamiento, el microcontrolador consume aproximadamente 60mA (Electrodaddy, 2023).

Por otro lado, el módulo GPS, en su modo de búsqueda, puede llegar a consumir hasta 67mA, mientras que en el modo de seguimiento su consumo se reduce a alrededor de 11mA (Sumador, 2023).

En el peor de los casos, cuando tanto el microcontrolador ESP32 como el módulo GPS estén en pleno funcionamiento, el consumo del sistema sería la suma de los consumos individuales de ambos componentes.

En este caso, el consumo total sería de:

$$60\text{ mA(ESP 32)} + 67\text{ mA(GPS)} = 127\text{ mA}$$

Además, es necesario sumar también el consumo del dispositivo ELM327 mini, que es de aproximadamente 45 mA. Este dispositivo, aunque se utiliza en el sistema, es independiente y se conecta directamente a la salida OBD2 del coche (Herramienta Automotriz, 2023).

$$127\text{ mA} + 45\text{ mA(ELM 327)} = 172\text{ mA}$$

---

En el caso en el que el coche esté en movimiento y, por lo tanto, el ESP32 tendrá conexión BLE y WiFi, el consumo podría aumentar hasta los 352 mA.

# CAPÍTULO 6: RECURSOS HARDWARE

En esta sección se presentarán de manera detallada los aspectos técnicos de cada uno de los dispositivos hardware empleados en el desarrollo del proyecto.

## 6.1 ESP-32

El ESP32 es un microcontrolador de bajo coste y bajo consumo de energía proveniente de la familia de los sistemas en chip (*SoC*), lo que significa que integra una variedad de componentes electrónicos en un solo chip, incluyendo procesadores, memoria, interfaces de comunicación y otros periféricos. Fue desarrollado por Espressif Systems y es una evolución del ESP8266, con mayores capacidades y mejores características. El ESP32 tiene dos núcleos de procesamiento Tensilica Xtensa LX6, que funcionan a una velocidad de reloj de hasta 240 MHz. También cuenta con WiFi integrado y soporte para Bluetooth 4.2 y BLE (Bluetooth de bajo consumo). Además, tiene una amplia variedad de periféricos, incluyendo puertos *SPI*, *I2C*, *UART*, *ADC*, *DAC*, *PWM* y más, lo que lo hace muy versátil para una variedad de aplicaciones. (AZ-Delivery, 2023)

El ESP32 se puede programar en varios lenguajes de programación, incluyendo C++, MicroPython, JavaScript y más, es muy popular en la comunidad de IoT debido a sus características, bajo coste y facilidad de uso. Se utiliza en una amplia variedad de proyectos de IoT, incluyendo sistemas de control de hogar inteligente, dispositivos de medición de calidad del aire, monitores de temperatura y humedad, dispositivos de seguimiento de mascotas entre otros.

En la siguiente imagen presenta la disposición y la función de los pines en el ESP32, lo cual resulta fundamental para la correcta conexión y configuración de los componentes externos.

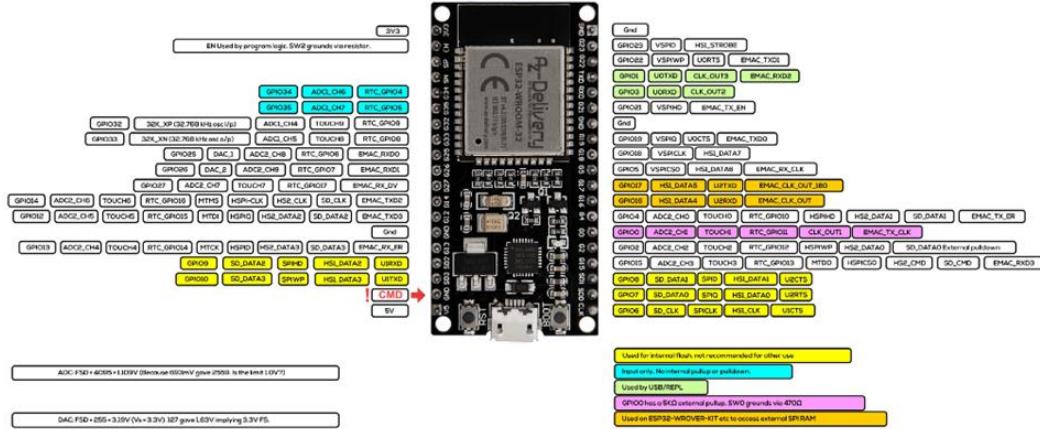


Figura 5 : ESP32 pinout

A continuación, se presenta una tabla con las características principales del ESP32, esta tabla proporciona una visión general rápida y concisa de las especificaciones más relevantes del ESP32, permitiendo una comparación fácil entre diferentes modelos o alternativas.

Voltaje de alimentación	5V
Voltaje de entrada / salida	3.3V
Corriente de Funcionamiento	min. 500mA
SoC	ESP32-WROOM 32
Frecuencia de Reloj	80MHz / 240MHz
RAM	512Kb
Memoria Flash externa	4MB
Pines I / O	34
Interfaces	SPI, I2C, I2S, CAN, UART
Protocolos WiFi	802.11n hasta 150 Mbps
Frecuencia WiFi	2.4 GHz - 2.5 GHz
Bluetooth	V4.2 - BLE y Bluetooth clásico
Antena inalámbrica	PCB
Dimensiones	56x28x13mm

Tabla 1 : Características ESP32

## 6.2 ELM327 mini

Es un microcontrolador fabricado por Elm Electronics, diseñado específicamente para traducir la interfaz *OBD*. Se trata de un pequeño adaptador electrónico que se conecta al puerto de diagnóstico a bordo de un vehículo, permitiendo la comunicación entre la *ECU* del vehículo y un dispositivo externo, proporcionando información en tiempo real sobre los sensores del motor y otros sistemas.

La siguiente tabla proporciona una visión general concisa de las especificaciones clave del ELM327, facilitando la comparación y selección del adaptador más adecuado para las necesidades de diagnóstico del vehículo. (Herramienta Automotriz, 2023)

Voltaje de alimentación	12V
Corriente de Funcionamiento	45mA
Frecuencia de Reloj	38400Hz
Bluetooth	BLE y Bluetooth clásico
Dimensiones	47x23x31mm
Peso	34g

Tabla 2 : Características ELM327

El dispositivo se conecta a través del puerto OBD-II (*On-Board Diagnostics*), que es un puerto de diagnóstico estándar encontrado en la mayoría de los vehículos fabricados a partir de 1996. El puerto OBD-II a través del protocolo CAN (*Control Area Network*) proporciona información de diagnóstico sobre el motor y otros sistemas del vehículo mediante de un conjunto estandarizado de códigos y protocolos de comunicación.

Los protocolos OBD-II soportados son los siguientes: (CSS Electronics, 2023)

- J1850 PWM (*Pulse Width Modulation*) utilizado por Ford Motor Company y Mazda.
- J1850 VPW (*Variable Width Modulation*) usado por General Motors y en algunos camiones o camionetas de carga ligera.
- ISO 9141-2 protocolo anterior de Chrysler, así como de vehículos europeos y asiáticos entre 2000-2004.

- ISO 14230-4 KWP2000 (*Keyword Protocol 2000*) común entre diversos fabricantes después del 2003, existen 2 variantes de este protocolo.
  - ISO 14230-4 KWP SLOW (5 baud init, 10.4 kbaud).
  - ISO 14230-4 KWP FAST (fast init, 10.4 kbaud).
- ISO 15765-4 CAN-BUS introducido en el 2003, siendo de uso general para todos los vehículos después del 2008, para este protocolo existen 4 variantes con diferentes longitudes de palabra y velocidad.
  - ISO 15765-4 CAN (11 bit ID, 500 Kbaud).
  - ISO 15765-4 CAN (29 bit ID, 500 Kbaud).
  - ISO 15765-4 CAN (11 bit ID, 250 Kbaud).
  - ISO 15765-4 CAN (29 bit ID, 250 Kbaud).

El conector OBD-II, le permite acceder fácilmente a los datos de su automóvil, la siguiente ilustración, muestra un ejemplo de un conector pin OBD-II tipo A (también conocido como conector de enlace de datos). Además, es importante mencionar que existen conectores OBD-II de tipo B que se utilizan específicamente en vehículos de servicio mediano y pesado. El pin 16 suministra energía de la batería, el *pinout* OBD-II depende del protocolo de comunicación El protocolo más común es CAN (a través de ISO 15765), lo que significa que los pines 6 (CAN-H) y 14 (CAN-L) normalmente se conectarán

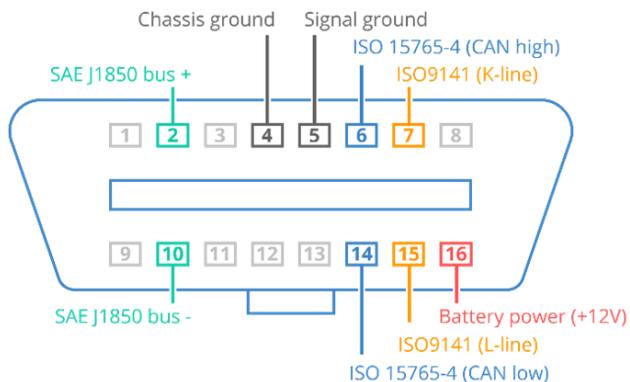


Figura 6 : Puerto OBD2

El estándar SAE J1979 OBD-II define 10 modos de diagnóstico OBD-II. El modo 1 muestra los datos actuales y se usa para ver parámetros en tiempo real como la velocidad del vehiculó, las RPM, la posición del acelerador, etc.

El modo 2 muestra los valores de los sensores en el momento en que se registra un código de falla (DTC). Permite revisar las condiciones específicas que provocaron el almacenamiento del código de falla.

El modo 3 permite leer de la memoria de la ECU todos los DTC almacenados.

EL modo 4 permite borrar los códigos de falla almacenados en la memoria del vehículo una vez que se hayan solucionado los problemas.

El modo 5 proporciona información sobre los resultados de los sistemas de monitoreo a bordo, como la prueba de evaporación de combustible, las pruebas de catalizador y otros sistemas de control del vehículo.

El modo 6 permite acceder a pruebas de diagnóstico adicionales específicas del fabricante que van más allá de los requisitos básicos del OBD-II. Estas pruebas son útiles para diagnosticar problemas más detallados o específicos del vehículo.

El modo 7 permite leer de la memoria de la ECU todos los DTC pendientes.

El modo 8 permite al dispositivo de diagnóstico enviar comandos de control y recibir respuestas del vehículo. Se utiliza para realizar pruebas y ajustes en los sistemas del vehículo, como activar componentes o realizar pruebas de actuadores.

El modo 9 proporciona información general sobre el vehículo, como el número de identificación del vehículo (VIN), el número de calibración del software y otros datos de identificación y configuración.

El modo 10 es usado para ver el histórico de códigos inclusive cuando estos ya han sido borrados (CSS Electronics, 2023).

---

#### OBD2 diagnostic services/modes (SAE J1979)

- 01** Show **current data** (e.g. real-time data)
- 02** Show **freeze frame data** (as above, but at time of freeze frame)
- 03** Show **stored Diagnostic Trouble Codes (DTCs)**
- 04** **Clear DTCs** and stored values
- 05** **Test results** for oxygen sensors (non CAN only)
- 06** **Test results** for system monitoring (and oxygen sensors for CAN)
- 07** Show **Pending DTCs**
- 08** **Control operation** of on-board system
- 09** Request **vehicle information** (e.g. VIN)
- 0A** **Permanent DTCs** (aka cleared DTCs)

Figura 7 : Modos OBD-II

### 6.3 GPS NEO-6M

El módulo GPS NEO-6M es un dispositivo compacto y económico que utiliza tecnología de posicionamiento global (*GPS*) para proporcionar información de posición de manera precisa. Utiliza una antena GPS de 25 x 25 mm, con conector UFL, para recibir señales de los satélites GPS y proporciona datos de posición, velocidad y tiempo en formato NMEA (*National Marine Electronics Association*). Posee una pequeña batería y una memoria EEPROM que permite guardar los últimos datos de posicionamiento y así lograr una detección más rápida, también dispone de una salida *TTL* serial y posee cuatro pines, TX, RX, VCC y GND de los cuales, los pines TX y RX se destinan a la comunicación serial, el pin VCC corresponde al pin de suministro de energía y el pin GND se encarga de la conexión a tierra, estos pines se conectarán directamente al ESP32. El voltaje de operación del módulo es de 3.3V - 5V, pero la lógica del puerto serial (TX, RX) es de 3.3V. Si se va a utilizar el pin RX con lógica de 5V, se debe emplear un divisor de tensión para reducirla a 3.3V. Por lo general solo se usa el pin TX ya que para conocer el posicionamiento solo se requiere leer las sentencias NMEA que genera el chip (Sumador, 2023).

*Figura 8 : GPS NEO-6M*

La siguiente tabla presenta las características principales del GPS NEO-6M, proporcionando una visión general concisa de las especificaciones clave del módulo.

Voltaje de alimentación	3.3 - 5V
Consumo de búsqueda	Hasta 67mA
Consumo en tracking	11mA aprox
Tipo de comunicación	UART
Velocidad de puerto serial por defecto	9600 baudios
Frecuencia de actualización de posición	Hasta 5Hz
Tipo de memoria	EEPROM
Tiempo de inicio	38 s aprox
Tecnología de seguridad	Anti-jamming
Tecnología para interiores	GPS SuperSense (-162 dBm sensibilidad de seguimiento)
SBAS soportados	WAAS, EGNOS, MSAS, GAGAN
Tamaño de la antena	25mm x 25 mm
Tamaño del módulo	25mm x 35mm

*Tabla 3 : Características GPS-NEO6M*



# CAPÍTULO 7: RECURSOS SOFTWARE

Este capítulo se centra en la descripción y análisis de los recursos software empleados, incluyendo lenguajes de programación, entornos de desarrollo integrados (*IDE*) y cualquier otro software relevante utilizado durante el proceso. Los recursos software desempeñan un papel fundamental en el logro de los objetivos propuestos, ya que proporcionan las herramientas y plataformas necesarias para el diseño, desarrollo, implementación.

## 7.1 MicroPython

MicroPython es un intérprete de Python que se ejecuta en dispositivos con recursos limitados, como microcontroladores y sistemas embebidos. Fue creado en 2013 por el científico de computación australiano Damien George, quien desarrolló un intérprete de Python que se ejecuta en microcontroladores ARM Cortex-M3.

Una de las características más importantes de MicroPython es que se puede programar directamente en el dispositivo, lo que lo hace muy útil en el desarrollo de proyectos de *IoT* y sistemas embebidos. Para programar el dispositivo, se utiliza un terminal serial que se conecta al dispositivo mediante un cable *USB*, además incluye una serie de módulos específicos para trabajar con periféricos y sensores a través de los puertos *GPIO*, *I2C*, *SPI*, *UART*, entre otros. Debido a su sintaxis limpia y legible, la gran cantidad de bibliotecas disponibles y la facilidad de aprendizaje será el lenguaje utilizado en este proyecto y el entorno de desarrollo que se utilizará será Thonny, un *IDE* (*Integrated Development Environment*) de código abierto para Python, diseñado para hacer que la programación en Python sea fácil y accesible utilizando una interfaz gráfica de usuario intuitiva y sencilla, que permite a los usuarios escribir y depurar código Python de manera eficiente además cuenta con la capacidad de interactuar con distintas placas de desarrollo, como el microcontrolador ESP32 mediante la instalación de módulos específicos (Circuitdigest, 2023).

## 7.2 ThingsBoard

ThingsBoard es una plataforma IoT de código abierto utilizada para la recopilación, el procesamiento, la visualización y la gestión de datos en tiempo real. Se trata de una solución altamente escalable y personalizable, que se adapta a las necesidades de cada proyecto.

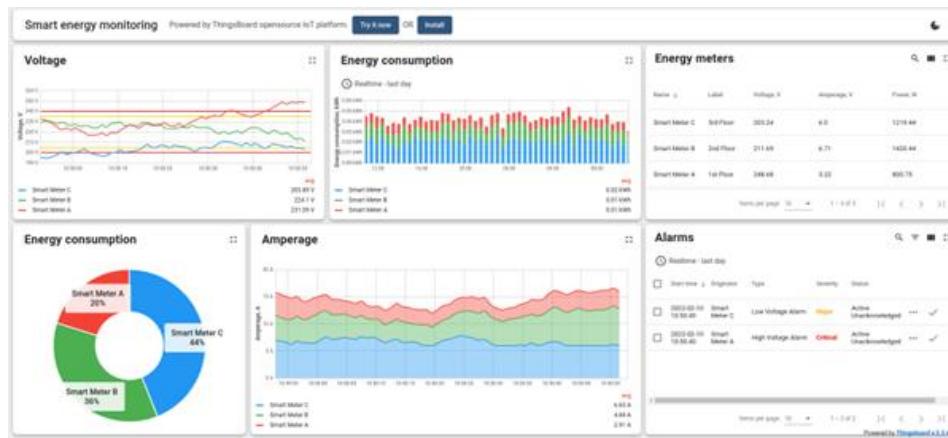


Figura 9 : Dashboard

Entre las principales ventajas de ThingsBoard, destacan su facilidad de uso, gracias a su interfaz de usuario intuitiva y amigable, así como la posibilidad de integrar dispositivos de diferentes fabricantes y protocolos, lo que permite una mayor flexibilidad y adaptabilidad a las necesidades específicas de cada proyecto, además de ser tolerante a fallos y escalable. Soporta protocolos de comunicación HTTPS, MQTT y CoAP, la conexión con los dispositivos hardware se hace mediante token de acceso y los datos obtenidos a través de un flujo definido por un motor de reglas. Otras ventajas significativas de ThingsBoard incluyen su capacidad para procesar grandes volúmenes de datos de manera eficiente y su amplia gama de herramientas de análisis y visualización de datos, que permiten a los usuarios tomar decisiones informadas basadas en información precisa y actualizada (ThingsBoard, 2023).

### 7.3 Protocolo MQTT

El protocolo MQTT (*Message Queuing Telemetry Transport*) es un protocolo de mensajería ligero y de código abierto diseñado para dispositivos y redes de baja potencia, y es ampliamente utilizado en el ámbito del Internet de las Cosas (IoT). MQTT se basa en un modelo de publicación-suscripción, donde los dispositivos (clientes) se conectan a un servidor central bróker, el bróker MQTT es responsable de recibir y enrutar los mensajes entre los dispositivos clientes que publican (publicadores) y los dispositivos clientes que suscriben (suscriptores) a determinados temas. Además de facilitar la comunicación entre los dispositivos, el bróker MQTT también puede gestionar la autenticación y autorización de los clientes, controlar la calidad del servicio de los mensajes y garantizar la entrega confiable de los mensajes.



Figura 10 : Protocolo MQTT

Una de las principales ventajas de MQTT es su eficiencia en el uso de la red, ya que utiliza un modelo de conexión persistente que minimiza la cantidad de datos de protocolo necesarios para establecer y mantener la conexión. Además, MQTT es altamente escalable con la capacidad de conectarse con millones de dispositivos IoT. Otra característica importante de MQTT es su soporte para calidades de servicio (QoS) configurables, que permiten a los dispositivos y aplicaciones establecer diferentes niveles de garantía de entrega de mensajes (MQTT, 2023).

## 7.4 Bluetooth Low Energy

El Bluetooth de baja energía (*Bluetooth Low Energy o BLE*), es un subconjunto del estándar Bluetooth v4.0. Dispone de una pila de protocolos en referencia a la capa OSI completamente nueva y orientada a conexiones sencillas en aplicaciones de muy baja potencia (dispositivos dependientes de batería o pila), fue desarrollada en 2010 por Bluetooth SIG (*Special Interest Group*) como una versión mejorada del Bluetooth clásico (Bluetooth, 2023).

Debido a su bajo consumo de energía, los dispositivos son capaces de funcionar durante meses o incluso durante años con una sola batería, debido a esto, es especialmente importante para dispositivos IoT que pueden estar ubicados en lugares remotos o de difícil acceso y no pueden ser alimentados por una fuente de energía externa. Un dispositivo Bluetooth Low Energy puede utilizar el mecanismo *Advertising* (publicidad), en el cual un dispositivo emite mensajes periódicos conocidos como "paquetes de publicidad". Estos paquetes contienen información sobre el dispositivo, como su identificador único y los servicios que ofrece, en este modo los dispositivos transmiten estos paquetes en intervalos regulares para anunciar su presencia y permitir que otros dispositivos cercanos los detecten, o *Connections* (conexiones), en cuyo modo dos dispositivos *BLE* se emparejan y establecen una conexión de comunicación bidireccional. Después de que un dispositivo en modo *Advertising* es detectado por otro dispositivo, se inicia el proceso de conexión. Esto implica el intercambio de información de emparejamiento y la negociación de parámetros de comunicación, como el intervalo de conexión y el nivel de seguridad.

En este escenario, el ESP32 asume el papel de dispositivo central o maestro, estableciendo una conexión con un periférico o esclavo específico, el módulo ELM327. El dispositivo central inicia una comunicación con el dispositivo periférico, una vez establecida la conexión, el dispositivo central gestionará la sincronización y el intercambio de datos. Por otro lado, el dispositivo periférico enviará periódicamente paquetes publicitarios que permiten la conectividad y también aceptará conexiones entrantes.

---

Cuando la comunicación queda establecida, el periférico deja de enviar paquetes publicitarios iniciándose la sincronización y control del dispositivo central. (Gotoiot, 2023)

La siguiente imagen ilustra la topología de conexión, donde el dispositivo central está conectado a varios periféricos.

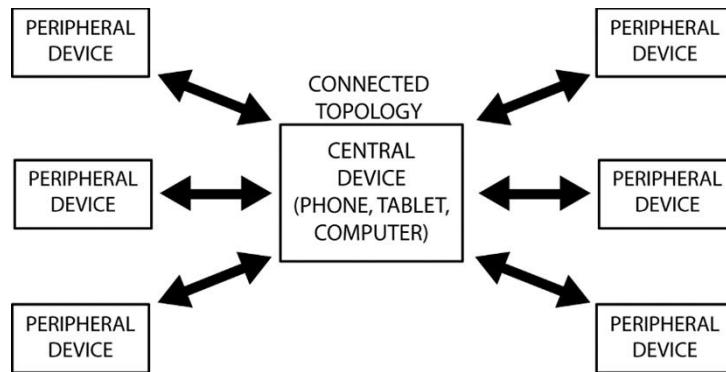


Figura 11 : Dispositivo central BLE

A pesar de que el dispositivo central gestiona la conexión, los datos pueden ser enviados por cualquiera de los dispositivos, del mismo modo un dispositivo puede actuar como central y periférico al mismo tiempo. *BLE* está organizado en 3 capas, la más alta se denomina Aplicación y es responsable de contener la lógica, la interfaz de usuario y el manejo de datos relacionados al caso de uso de la aplicación. La capa intermedia es el *Host*, conformado por los protocolos *Generic Access Profile* (GAP), *Generic Attribute Profile* (GATT), *Logical Link Control and Adaptation Protocol* (L2CAP), *Attribute Protocol* (ATT), *Security Manager* (SM) y la capa *Host Controller Interface* (HCI) del lado del host. Por último, la capa más inferior de la arquitectura es el Controlador, formado por los protocolos *Host Controller Interface* (HCI), la capa *Link Layer* (LL) y la capa *Physical Layer* (PHY). (Gotoiot, 2023)

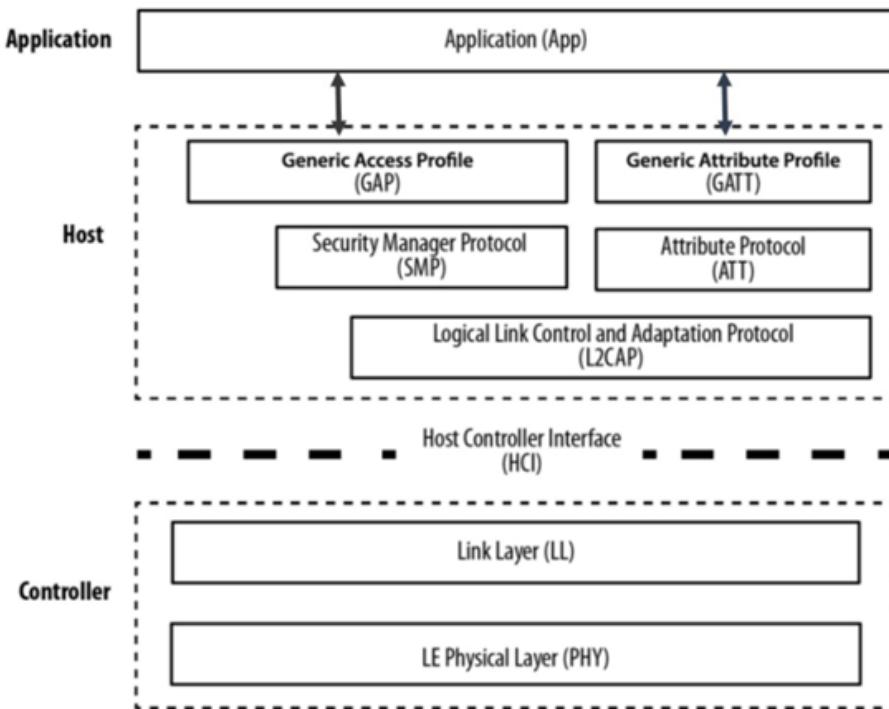


Figura 12 : Capas BLE

Dentro de la capa *Host* los protocolos que interactúan directamente con la Aplicación son los protocolos de las capas *GAP* Y *GATT*. La capa *GAP* define cómo los dispositivos BLE descubren, se conectan y se autentican entre sí. Además, el perfil *GAP* permite que los dispositivos definan sus roles, como dispositivo central o periférico, y manejen los requisitos de seguridad y privacidad.

La capa *GATT* se basa en el perfil *GAP* y se encarga de la transferencia de datos y el acceso a los servicios y características de los dispositivos BLE. Define la estructura y el formato de los datos, así como los procedimientos para leer, escribir y notificar cambios en los valores de los atributos.

La siguiente imagen presenta una descripción visual de los elementos clave de la capa *GATT*, como los perfiles, los servicios, las características y los descriptores (Gotoiot, 2023).

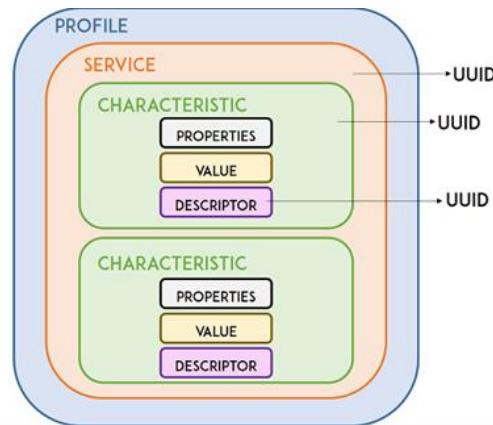


Figura 13 : Capa GATT BLE

El Perfil es el nivel más alto de la jerarquía y es una colección predefinida de servicios compilada por Bluetooth *SIG* y por los diseñadores de periféricos. Puede contener uno o más servicios. El Servicio es un conjunto de características relacionadas que realizan una función específica, que puede, o no, centrarse en la información proveniente de un solo sensor. Las Características son una propiedad de un servicio y es quien contiene los datos reales, por un lado, el Descriptor que proporciona información adicional sobre una característica y por otro su valor. Las Propiedades describen cómo se puede interactuar con el valor de una Característica, ya sea leyendo o escribiendo sobre esta. El Identificador Único Universal (*UUID*) es único para cada perfil, cada servicio y cada característica, cuya funcionalidad es la identificación únicamente a los componentes (Gotoiot, 2023).

Una vez explicado el funcionamiento de *Bluetooth Low Energy* cabe destacar las principales diferencias frente al Bluetooth clásico, dichas diferencias son las mostradas en la siguiente tabla:

Especificaciones	Classic Bluetooth	Low Energy
Rango	100m	Mayor que 100m
Velocidad de transmisión	1-3 Mbps	1 Mbps
Frecuencia	2.4 Ghz	2.4 Ghz
Seguridad	56/128-bit	128-bit AES
Robustez	Adaptative fast frequency	24-bit CRC
Latencia	100 ms	6 ms

Especificaciones	Classic Bluetooth	Low Energy
Capacidad de voz	Si	No
Topología de red	Estrella	Estrella
Potencia consumida	1 W	0.01 a 0.5 W
Pico de consumo	Menos de 30mA	Menos de 15 mA

Tabla 4 : Diferencias entre BLE y Bluetooth clásico

Asimismo, al explicar las ventajas de BLE en comparación con otras tecnologías, se puede brindar una visión completa de las características distintivas que ofrece.

Variable	Wi-Fi	Z-Wave	Zigbee	Thread	BLE (V4.2)
Year first launched in market	1997	2003	2003	2015	2015
PHY/MAC Standard	IEEE 802.11.1	ITU-T G.9959	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.1
Frequency Band	2.4GHz	900MHz	2.4GHz	2.4GHz	2.4GHz
Nominal Range @ 0dBm	100m	30-100m	10-100m	10-100m	30m
Maximum Data Rate	54 Mbps	40-100kbps	250kbps	250kbps	1Mbps
Topology	Star	Mesh	Mesh	Mesh	Scatternet
Power Consumption	High	Low	Low	Low	Low
Alliance	Wi-Fi Alliance	Z-Wave Alliance	Zigbee Alliance	Thread Group	Bluetooth Sig

Tabla 5 : Principales tecnologías de comunicación

## 7.5 Telegram

Telegram es una aplicación de mensajería instantánea que permite a los usuarios enviar y recibir mensajes, fotos, videos y otros archivos a través de Internet. Fue desarrollada en 2013 por los hermanos Nikolái y Pável Dúrov, y se destaca por su enfoque en la privacidad y seguridad de los usuarios.

Ofrece dos tipos de API's para diferentes plataformas: La *Bot API* permite crear fácilmente programas que usan mensajes de Telegram en una interfaz. La API de Telegram y TDLib le permiten crear sus propios clientes de Telegram personalizados. Ambas API's son totalmente gratuitas. En este proyecto, se empleará exclusivamente la API de Bot para generar notificaciones de manera inmediata al usuario (Telegram, 2023).

# CAPÍTULO 8: CONECTIVIDAD A INTERNET

Para abaratar los costes y simplificar el sistema, se hará uso de un teléfono móvil como punto de acceso WiFi, de esta forma el ESP32 tendrá acceso a Internet de una forma rápida y sencilla. No obstante, en el mercado existen numerosos módulos que también brindan acceso a Internet. A continuación, se describirán de manera concisa los más relevantes, junto con sus ventajas y desventajas.

## 8.1 SIM800L GSM GPRS

El módulo SIM800L es un módem *GSM/GPRS* que permite la comunicación móvil de voz y datos en una amplia gama de aplicaciones. Este módulo funciona con una tensión de 3.7-4.2V y utiliza un protocolo *UART* para la comunicación con otros dispositivos (AZ-Delivery, 2023).



Figura 14 : SIM800L GSM GPRS

Las distintas ventajas son:

- Compatibilidad con una amplia gama de redes móviles en todo el mundo.
- Es pequeño y compacto, lo que lo hace fácil de integrar en dispositivos móviles y otros sistemas.
- Permite la transmisión de datos a través de GPRS/EDGE y la conexión a Internet.

- Tiene un bajo consumo de energía, lo que lo hace adecuado para aplicaciones de baja potencia.

Por otro lado, las desventajas del módulo son:

- La velocidad de transmisión de datos es relativamente lenta en comparación con otros módems.
- La conexión a internet puede ser limitada en áreas de baja cobertura o mala calidad de señal.
- Puede haber problemas de compatibilidad con algunos sistemas debido a las limitaciones del protocolo UART.

## 8.2 SIM7600E-H

El módulo SIM7600E-H es un módulo de comunicaciones inalámbricas de alta velocidad, diseñado para proporcionar conectividad de red de datos a dispositivos electrónicos (Waveshare, 2023).



Figura 15 : SIM7600E-H

Entre sus principales ventajas se encuentran:

- Soporte para redes 4G LTE, 3G y 2G, con velocidades de descarga y carga de hasta 150Mbps y 50Mbps, respectivamente.
- Dispone de interfaces de comunicación como USB, UART y GPIO, lo que lo hace fácilmente adaptable a diferentes sistemas.

- Soporte para múltiples bandas de frecuencia de operación, lo que permite su uso en diferentes partes del mundo.
- Incluye funciones avanzadas de seguridad y autenticación, lo que lo hace adecuado para aplicaciones en entornos sensibles a la seguridad.
- Dispone de una memoria interna de 4GB, que permite el almacenamiento de datos en el propio módulo.

Entre las desventajas del módulo se encuentran:

- Su precio puede ser relativamente alto comparado con otros módulos de comunicación de datos.
- Requiere de una fuente de alimentación estable y suficiente para su correcto funcionamiento.
- Puede ser más complejo de utilizar que otros módulos de comunicación inalámbrica, por lo que puede requerir de conocimientos técnicos especializados para su configuración y uso adecuado.

### 8.3 LORA SX1278

El módulo LORA SX1278 es un dispositivo de comunicación inalámbrico de largo alcance que utiliza la tecnología LoRa (*Long Range*) para establecer conexiones de comunicación de bajo consumo y larga distancia. Este módulo opera en la banda de frecuencia de 433MHz y se puede utilizar en conjunto con microcontroladores como Arduino o ESP32 (Amazon, 2023).



Figura 16 : LORA SX1278

Entre las principales ventajas del módulo se encuentran:

- Permite la comunicación a distancias de varios kilómetros.
- Tiene un bajo consumo de energía lo que permite que el módulo funcione con baterías durante largos períodos de tiempo.
- Interferencia mínima debido a la utilización de técnicas de modulación de espectro ensanchado que permiten una mayor resistencia a las interferencias.
- Facilidad de integración, el módulo es capaz de integrarse fácilmente con otros dispositivos de microcontrolador, como Arduino o ESP32.

Sin embargo, también existen algunas desventajas asociadas con este módulo, entre ellas:

- Velocidad de transmisión limitada, la velocidad de transmisión de datos es más lenta en comparación con otros módulos de comunicación inalámbrica.
- Limitaciones de frecuencia, ya que este módulo solo puede operar en la banda de frecuencia de 433MHz, lo que puede limitar su uso en algunos países donde esta banda no está disponible para uso público.
- Configuración compleja, lo que puede requerir un mayor conocimiento técnico.

## 8.4 Wisol SFM10R1

El módulo Wisol SFM10R1 es un módulo de comunicación inalámbrica de bajo consumo diseñado para la red Sigfox. Este módulo ofrece un enfoque de bajo consumo de energía para aplicaciones *IoT*, lo que lo hace ideal para dispositivos alimentados por batería (Rs-online, 2023).

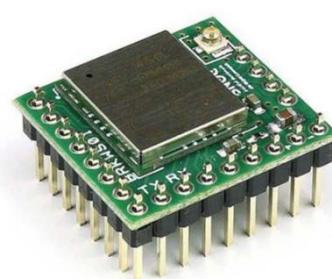


Figura 17 : Wisol SFM10R1

Algunas de las ventajas incluyen:

- Bajo consumo de energía, este módulo consume muy poca energía en comparación con otras tecnologías de comunicación inalámbrica.
- Cobertura global, la red Sigfox está expandiéndose rápidamente en todo el mundo, lo que significa que el módulo se puede utilizar en prácticamente cualquier lugar.
- Fácil de usar, se puede integrar fácilmente en una amplia variedad de aplicaciones IoT.

Algunas de las desventajas son:

- Velocidad de transmisión limitada, la velocidad de transmisión de datos es limitada, por lo que no es adecuado para aplicaciones que requieren grandes cantidades de datos.
- El coste puede ser más alto en comparación con otras tecnologías de comunicación inalámbrica.



# CAPÍTULO 9: CONFIGURACIONES DEL SISTEMA

En este apartado se abordarán las configuraciones necesarias para utilizar los distintos elementos que diferentes elementos que componen el proyecto. Se explicará detalladamente cómo configurar cada componente de manera adecuada para su correcto funcionamiento e integración dentro del sistema.

## 9.1 Configuración de ESP32

MicroPython es una implementación del lenguaje de programación Python optimizada para microcontroladores, como el ESP32. Lo primero será descargar el firmware de MicroPython desde el sitio oficial de MicroPython. Se puede programar la placa utilizando el programa [esptool.py](#), la primera vez, se debe borrar toda la memoria flash utilizando el siguiente comando:

```
esptool.py --chip esp32 --port /dev/ttyUSB0 erase_flash
```

A continuación, se escribirá el archivo binario en la memoria flash de un ESP32 a través del programa esptool.py, estableciendo la configuración adecuada como el chip, el puerto serie, la velocidad de comunicación y la dirección de inicio de escritura.

```
esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 460800  
write_flash -z 0x1000 esp32-20190125-v1.10.bin
```

## 9.2 Comunicación con ELM327

Una vez establecida la conexión Bluetooth Low Energy entre el dispositivo ELM327 y el microcontrolador ESP32 utilizando la biblioteca bluetooth (Bluetooth, 2023) de MicroPython, para poder recibir los datos provenientes de los sensores del automóvil será necesario conocer los servicios de transmisión de datos del dispositivo ELM327, junto con sus respectivos UUID. Para la visualización de los distintos servicios BLE de dicho

---

dispositivo se hará uso de la aplicación móvil *LightBlue*. Con *LightBlue*, los usuarios pueden descubrir dispositivos Bluetooth cercanos, ver información detallada de cada dispositivo, establecer conexiones Bluetooth y acceder a Servicios y Características específicas proporcionadas por los dispositivos. La aplicación también permite enviar y recibir datos a través de Bluetooth, lo que facilita la interacción y el control de dispositivos compatibles.

En la imagen siguiente se pueden apreciar los distintos servicios y características ofrecidos por el dispositivo ELM327.

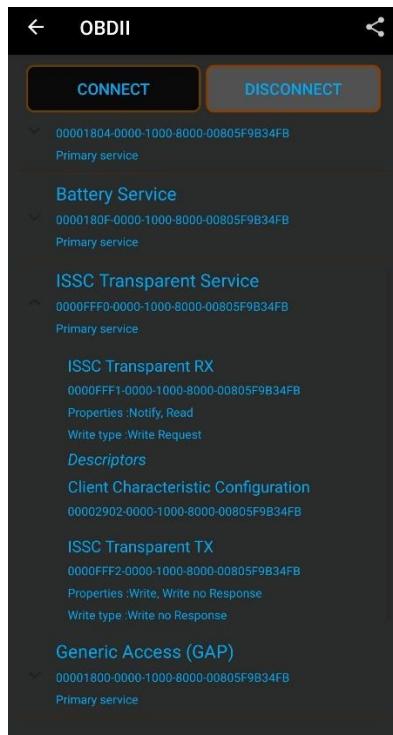


Figura 19 : Servicios BLE de ELM327

Una vez determinado que el UUID del servicio de transmisión de datos empieza por la dirección 0xFFFF0, se procederá a examinar las características de dicho servicio. Se encontraron dos características cuya finalidad es la comunicación en serie, una con la dirección 0xFFFF1 para la transmisión RX y otra con la dirección 0xFFFF2 para la transmisión TX. Sin embargo, en este caso en particular para poder recibir datos provenientes del ELM327, es necesario habilitar la función de notificación que se encuentra en el Descriptor con dirección 0x2902 de la Característica 0xFFFF1.

---

Para interactuar con los Descriptores y Características utilizando la biblioteca Bluetooth de MicroPython, es necesario obtener previamente los valores hexadecimales correspondientes a dichas capas.

```
#Valor en hexadecimal RX de la Característica 0xFFFF1
read
value_handle_read=0x000C
#Valor en hexadecimal del descriptor notify 0x2902
value_dschandle=0x000D
#Valor en hexadecimal TX de la Característica 0xFFFF2
write
value_handle_write=0x000F
```

La inicialización del dispositivo se hará con los siguientes comandos:

```
#Activar notificaciones una vez se escriba en el módulo
ELM327
self.write_data(value_dschandle,b'\x01\x00')
#Reiniciar el dispositivo
self.write_data(value_handle_write,b'ATZ\r\n')
#Comunicación OBD-II en modo automático
self.write_data(value_handle_write,b'ATSP0\r\n')
#Modo eco desactivado
self.write_data(value_handle_write,b'ATE0\r\n')
```

La comunicación con el dispositivo ELM327 se hace mediante comandos AT como los mostrados anteriormente. El comando **ATZ** generalmente se utiliza para realizar un reinicio o restablecimiento del dispositivo ELM327. El comando **ATSP0** establece el protocolo de comunicación OBD-II en modo automático. **ATE0** representa un comando para desactivar el eco de los comandos enviados al dispositivo ELM327.

Los principales comandos que se usarán en este proyecto para la recopilación de datos serán los siguientes (CSS Electronics, 2023):

```
#Lectura de la temperatura del refrigerante del motor
self.write_data(value_handle_write,b'0105\r\n')
#Lectura de la presión absoluta del colector de
admisión
self.write_data(value_handle_write,b'010B\r\n')
#Lectura de las rpm del motor
self.write_data(value_handle_write,b'010C\r\n')
#Lectura de la velocidad del vehículo
self.write_data(value_handle_write,b'010D\r\n')
#Posición del pedal del acelerador
self.write_data(value_handle_write,b'0149\r\n')
#Nivel del combustible
self.write_data(value_handle_write,b'012F\r\n')
```

---

### 9.3 Comunicación con GPS NEO-6M

El módulo GPS NEO-6M se conecta al ESP32 mediante una conexión serial (UART). El ESP32 actúa como maestro, mientras que el módulo GPS NEO-6M se comporta como esclavo. Para establecer la comunicación, se configuran los parámetros de comunicación del puerto serial del ESP32, la velocidad de transmisión será de 9600 (*baud rate*), los *bits* de datos indican la cantidad de *bits* utilizados para representar cada carácter, en este caso, 8 *bits*. No se utiliza ningún *bit* de paridad para verificar la integridad de los datos, y se utilizará 1 bit de parada para determinar cuántos bits se envían al final de cada carácter indicando el final de la transmisión.

Una vez configurado el puerto serial, el GPS NEO-6M envía al ESP32 a información de posición y otros datos de navegación. Estos comandos están definidos en el estándar NMEA (*National Marine Electronics Association*) y se envían en forma de cadenas de texto.

Como se muestra en la siguiente imagen, estas cadenas de texto contienen datos como la, como la latitud, longitud, altitud, velocidad, dirección, entre otros (Mathworks, 2023).

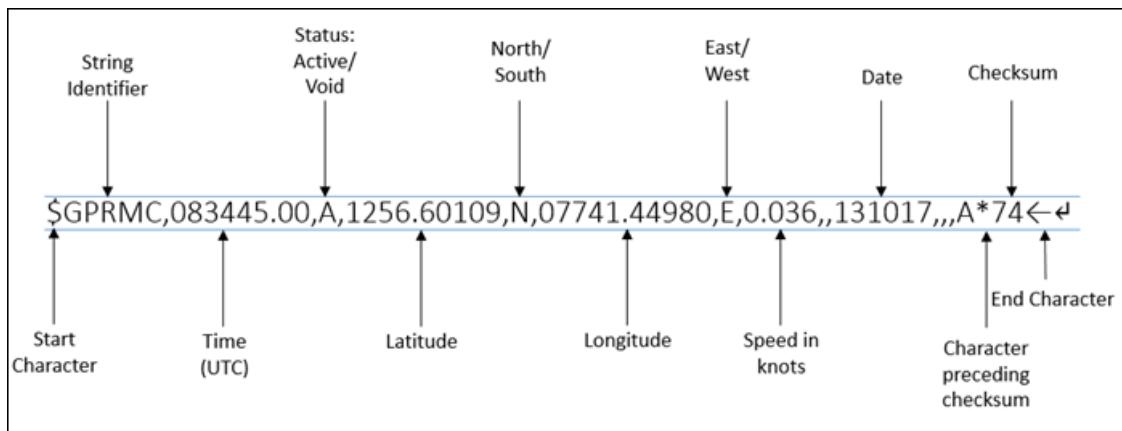


Figura 20 : Trama GPS

## 9.4 Configuración ThingsBoard

ThingsBoard ofrece la posibilidad de utilizar *Widgets*, que son componentes visuales diseñados para mostrar información de manera atractiva. Estos *Widgets* permiten crear paneles visualmente atractivos que facilitan la visualización y el análisis de datos.

En primer lugar, es necesario agregar un nuevo dispositivo en ThingsBoard. A continuación, se muestra el dispositivo creado previamente en la plataforma.

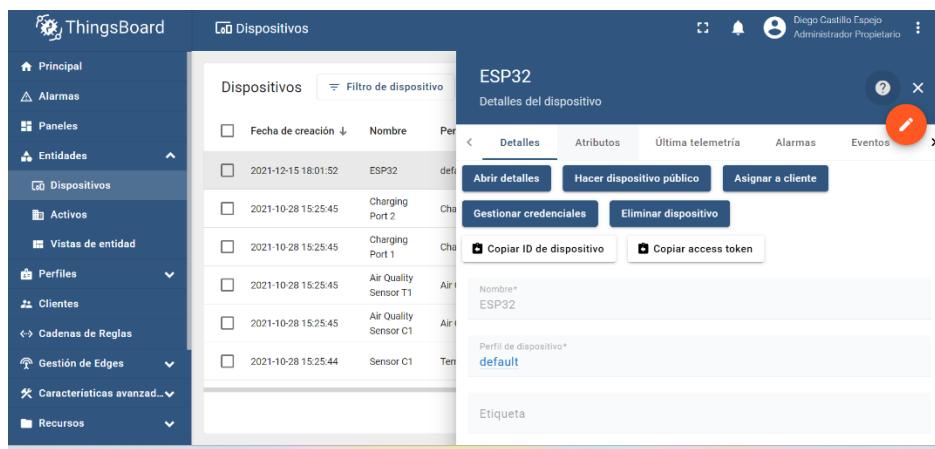


Figura 21 : Dispositivo en Thingsboard

La comunicación entre el microcontrolador y la plataforma se hace mediante el protocolo MQTT, utilizando el token de acceso proporcionado por ThingsBoard, para ello se hará uso de la librería *TBDeviceMqttClient* (GitHub, 2023).

Para facilitar la detección de valores anormales en los datos de telemetría, se realizará la configuración de alarmas en el sistema. Estas alarmas se basan en cadenas de reglas que permiten establecer el flujo de trabajo y las condiciones específicas para cada una de ellas. Mediante esta configuración, se podrá visualizar de manera eficiente cualquier dato que se considere fuera de los parámetros normales, facilitando así la identificación de posibles problemas o situaciones críticas (ThingsBoard, 2023).

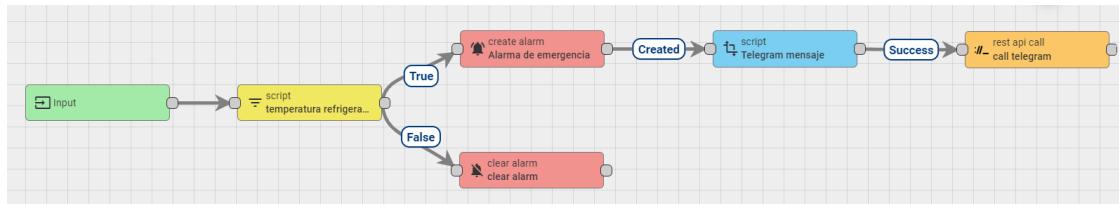


Figura 22 : Cadena de reglas en Thingsboard

En este caso, se configuró una alarma que se activará únicamente cuando la temperatura del aceite del motor supere un límite elevado predefinido. Esta alarma se estableció utilizando una cadena de reglas específicas que se encargará de monitorear continuamente la temperatura del refrigerante. Si la temperatura medida excede el límite establecido, la alarma se activará, lo que permitirá una respuesta rápida ante una situación potencialmente peligrosa o dañina para el motor.

A continuación, se presenta una imagen que ilustra la programación del límite a partir del cual se activará la alarma.

```

function Filter(msg, metadata, msgType) {
  1 | return msg.temp_refrigerante > 100;
}
  
```

Figura 23 : Comportamiento temperatura alta

Una vez que se han definido los valores y se ha creado la alarma, es necesario declarar el mensaje que se enviará Telegram y utilizar una API que permita la comunicación con la plataforma. Para poder enviar el mensaje a Telegram, es necesario conocer el token de acceso y *chat ID* del *Bot* creado, el cual será proporcionado por Telegram.

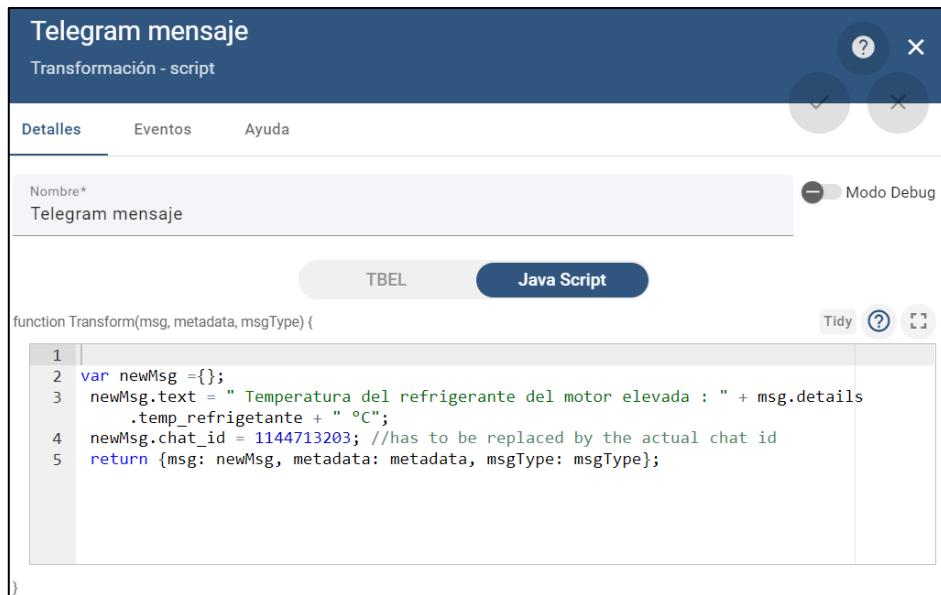


Figura 24 : Función mensaje Telegram

Por último, se define el eslabón de la cadena *Rest API call*, el cual se encarga de la llamada a la *Application Programming Interface* de Telegram para enviar el mensaje previamente definido.

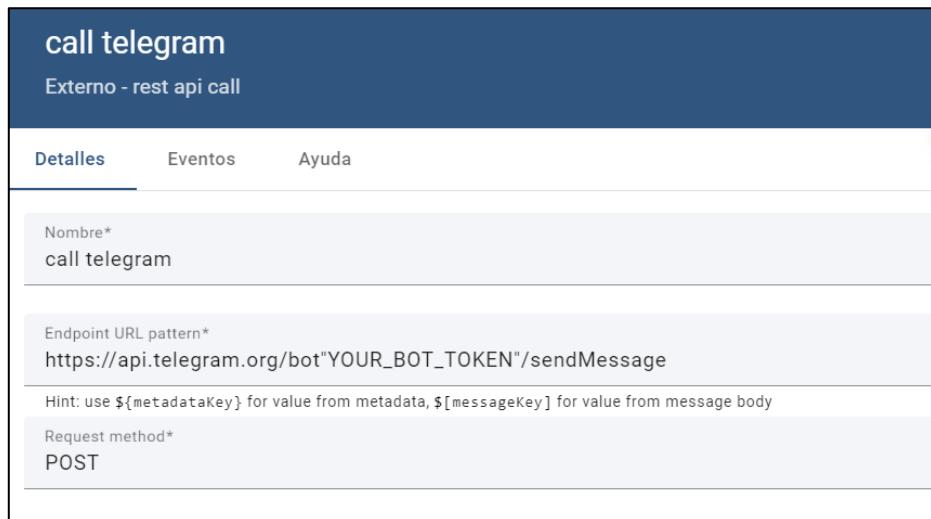


Figura 25 : Rest API Telegram

Todas las cadenas de reglas, a excepción de la raíz, se configuran de manera similar, diferenciándose únicamente en las variables que evalúan y los mensajes que se envían al *Bot* de Telegram. Después de configurar cada cadena de reglas para cada alarma específica, es necesario integrarlas en la cadena de reglas raíz.

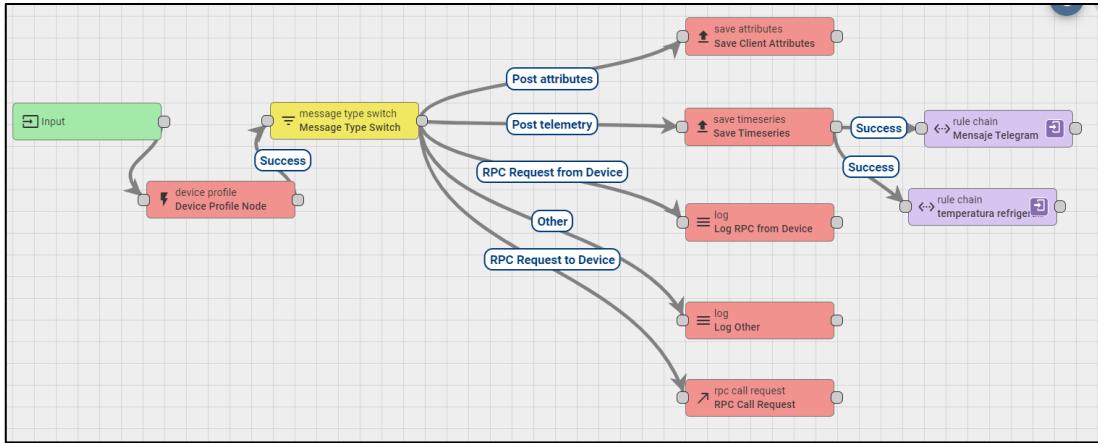


Figura 26 : Cadena de reglas raíz

## 9.5 Configuración Telegram

Para configurar un *Bot* de Telegram, es necesario buscar *BotFather* en la barra de búsqueda de Telegram. *BotFather* es una herramienta especializada que permite a los usuarios crear y personalizar su propio *Bot* de manera sencilla y rápida, adaptándolo a sus necesidades y objetivos particulares.

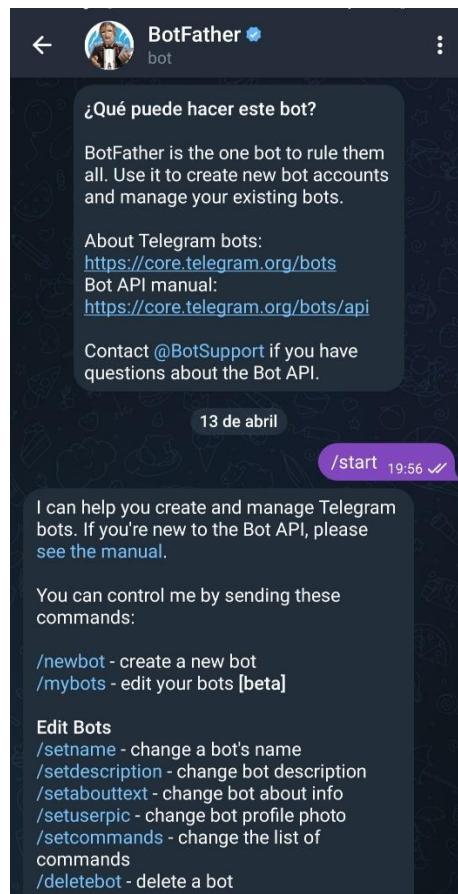


Figura 27 : Bot de Telegram

Posteriormente se tendrá que insertar el comando `/newbot` y un nombre de usuario que se usará dicho *Bot*. A continuación, Telegram proporcionará un token de acceso y un identificador de chat con los que se podrá enviar mensajes a dicho *Bot*. Los mensajes serán enviados a Telegram desde la plataforma ThingsBoard mediante el uso de reglas, como se ha mostrado en el apartado anterior. Para esta comunicación, se utilizará el formato JSON (*JavaScript Object Notation*), el cual es un formato de intercambio de datos ligero, fácil de leer y escribir. Este formato se basa en la sintaxis de objetos de JavaScript y es ampliamente utilizado en muchos lenguajes de programación debido a su simplicidad y compatibilidad.



# CAPÍTULO 10: PRUEBAS UNITARIAS

Las pruebas unitarias son una práctica fundamental en el desarrollo de software que permite asegurar la calidad y el correcto funcionamiento de los programas. Este capítulo se centrará en verificar el comportamiento individual de las distintas funcionalidades que tiene el sistema.

## 10.1 vehículo detenido

Inicialmente, se obtienen los datos del GPS y se espera un periodo corto de tiempo para volver a obtener datos. Una vez que se ha realizado esta obtención de datos, se procede a realizar una comprobación. En caso de que los datos de latitud y longitud obtenidos del módulo GPS no varíen en el tiempo definido, se activará el modo de *deepsleep* del microcontrolador durante un tiempo determinado, en este caso, el tiempo de espera será configurable antes de reiniciar el proceso. Este enfoque permite ahorrar energía durante los períodos en los que el coche permanece detenido, reduciendo el consumo de energía del sistema y prolongando la duración de la batería.

En la siguiente ilustración se presenta un desglose de los datos obtenidos junto con la hora actual en la que se lanza cada proceso:

```

Hora actual : 00:13:26 Inicio del sistema
Latitud = 40.39042 Longitud = -3.627067
Hora actual : 00:13:28 Obtención de nuevos datos
Latitud = 40.39042 Longitud = -3.627067
Hora actual : 00:13:28 Activando modo deepsleep
ets Jul 29 2019 12:21:46

rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4540
ho 0 tail 12 room 4
load:0x40078000,len:12448
load:0x40080400,len:4124
entry 0x40080680
Hora actual : 00:13:58 Inicio del sistema
Latitud = 40.39042 Longitud = -3.627067
Hora actual : 00:14:00 Obtención de nuevos datos
Latitud = 40.39042 Longitud = -3.627067
Hora actual : 00:14:00 Activando modo deepsleep

```

*Figura 28 : Prueba Vehículo detenido*

No obstante, es importante considerar la existencia de una tasa de error inherente al módulo GPS, dado que su omisión podría acarrear imprecisiones tanto en el proceso de determinar si el vehículo está en movimiento como en la medición de la distancia recorrida, proceso que se detallará más adelante.

```

Error de longitud: 0.00000953674316406250
Error de latitud : 0.00001144409179687500
Error de longitud: 0.00000500679016113281
Error de latitud : 0.00001144409179687500
Error de longitud: 0.00000810623168945313
Error de latitud : 0.00000762939453125000
Error de longitud: 0.00001287460327148438
Error de latitud : 0.000000000000000000000000
Error de longitud: 0.00002217292785644531
Error de latitud : 0.00001525878906250000
Error de longitud: 0.00001573562622070313
Error de latitud : 0.000000000000000000000000
Error de longitud: 0.00001215934753417969
Mayor error de latitud : 0.00001907348632812500
Mayor error de longitud: 0.00002217292785644531

```

*Figura 29 : Medición de error GPS*

En la imagen previa se presentan los diversos errores generados al obtener los datos del GPS en una misma posición. A través de la repetición de dicho procedimiento en múltiples ocasiones, se ha llegado a la conclusión de que el error máximo para la latitud es de 0.00003507348632812500 grados decimales, mientras que el error máximo para la

longitud es de 0.00003517292785644531 grados decimales, lo que significa que el módulo tiene un error de unos 5 metros aproximadamente.

## 10.2 Lectura de OBD-II

En esta sección se mostrará el análisis de la obtención de datos de los sensores del automóvil durante su funcionamiento en movimiento, en particular, se presentará la adquisición de las revoluciones por minuto (RPM), habiéndose llevado a cabo dicha operación para cada uno de los sensores incorporados en el código. Dado que los datos se obtienen a través de respuestas a comandos AT enviados al dispositivo ELM327, siendo los comandos AT una serie de comandos y respuestas estandarizadas utilizadas para comunicarse con dispositivos de comunicación, es importante estudiar el tiempo de respuesta y procesamiento de los datos.

Además, como se muestra en la siguiente imagen hay que tener en cuenta que las respuestas del dispositivo ELM327 se presentan en bytes hexadecimales y siguiendo un formato específico.

```
notificacion ELM327 bytearray(b'41 0C 0B F8 \r')
Notificacion pasado String 410C0BF8
Tiempo transcurrido en la obtencion de RPM: 148 ms
RPM = 766.0
```

Figura 30 : Ejemplo de RPM

Como se muestra en la imagen anterior, se puede observar el formato de los datos recibidos, cuya trama de bytes es: "41 0C 0B F8", el primer byte, "41", indica que se trata de una respuesta a una solicitud, en este caso, una respuesta a una solicitud del modo 1 ( $01 + 40 = 41$ ).

El segundo byte, "0C", repite el número PID (*Parameter IDentification*) solicitado, los PID son identificadores de parámetros utilizados para solicitar información específica de un vehículo a través del puerto de diagnóstico OBD-II. Los bytes restantes contienen la información solicitada. (Elm Electronic , 2023)

Una vez realizada la verificación de los datos obtenidos se procede a un estudio de los tiempos de respuesta. Esto implica medir el tiempo transcurrido desde el envío del comando AT hasta la conversión de los datos en formato decimal utilizando la escala adecuada.

```
value handle descriptor: 1
Notificaciones activas
Reinicio de dispositivo con ATZ
Comunicación OBD-II en modo automático ATSP0
Modo eco desactivado con ATE0
Tiempo transcurrido en conexión BLE: 2457 ms
-----
notificacion ELM327 bytearray(b'SEARCHING...41 0C 0B FC \r')
Notificación pasado String SEARCHING...410C0BFC
Tiempo transcurrido en la obtención de RPM: 276 ms
RPM = 767.0
-----
notificacion ELM327 bytearray(b'41 0C 0B F8 \r')
Notificación pasado String 410C0BF8
Tiempo transcurrido en la obtención de RPM: 148 ms
RPM = 766.0
-----
notificacion ELM327 bytearray(b'41 0C 0B F8 \r')
Notificación pasado String 410C0BF8
Tiempo transcurrido en la obtención de RPM: 148 ms
RPM = 766.0
-----
notificacion ELM327 bytearray(b'41 0C 0B F4 \r')
Notificación pasado String 410C0BF4
Tiempo transcurrido en la obtención de RPM: 148 ms
RPM = 765.0
-----
notificacion ELM327 bytearray(b'41 0C 0C 04 \r')
Notificación pasado String 410C0C04
Tiempo transcurrido en la obtención de RPM: 148 ms
RPM = 769.0
-----
notificacion ELM327 bytearray(b'41 0C 0B EC \r')
```

Figura 31 : Ejemplo de tiempos RPM

### 10.3 Transición de estado del vehículo

Cuando el vehículo se encuentra en movimiento y transita hacia el modo de parada, las revoluciones por minuto (RPM) del automóvil se reducen a cero. Por consiguiente, esta variable será utilizada para determinar los cambios de estado del vehículo.

La siguiente ilustración captura el descenso gradual de las RPM de un automóvil desde el momento en que la llave de encendido se gira al modo apagado.

```

notificacion ELM327 bytearray(b'41 0C 0B 8C \r')
Notificacion pasado a String 410C0B8C
Tiempo transcurrido en la obtencion DE RPM: 269 ms
RPM= 739.0
-----
notificacion ELM327 bytearray(b'41 0C 0A 68 \r')
Notificacion pasado a String 410C0A68
Tiempo transcurrido en la obtencion DE RPM: 149 ms
RPM= 666.0
-----
notificacion ELM327 bytearray(b'41 0C 07 5C \r')
Notificacion pasado a String 410C075C
Tiempo transcurrido en la obtencion DE RPM: 149 ms
RPM= 471.0
-----
notificacion ELM327 bytearray(b'41 0C 06 20 \r')
Notificacion pasado a String 410C0620
Tiempo transcurrido en la obtencion DE RPM: 149 ms
RPM= 392.0
-----
notificacion ELM327 bytearray(b'41 0C 00 00 \r')
Notificacion pasado a String 410C0000
Tiempo transcurrido en la obtencion DE RPM: 149 ms
RPM= 0.0
-----
notificacion ELM327 bytearray(b'41 0C 00 00 \r')
Notificacion pasado a String 410C0000
Tiempo transcurrido en la obtencion DE RPM: 149 ms
RPM= 0.0
-----+
notificacion ELM327 bytearray(b'41 0C 00 00 \r')
Notificacion pasado a String 410C0000
Tiempo transcurrido en la obtencion DE RPM: 149 ms

```

*Figura 32 : Cambio de estado del vehículo*

No obstante, existen diversas situaciones en las cuales las revoluciones por minuto (RPM) pueden ser registradas como cero. Dichos escenarios pueden incluir errores puntuales en la lectura del sistema de diagnóstico a bordo (OBD-II) o una parada intencional realizada por el conductor, así como también las paradas inducidas por el propio vehículo, como es el caso de los automóviles equipados con sistemas de arranque y parada automática (*Start-Stop*). Considerando esta situación, se ha implementado una pausa activa en el código del programa. Si, transcurrido un periodo de tiempo específico, las revoluciones por minuto continúan registrándose como cero, se procede a finalizar el trayecto y retornar al estado inicial. Por otro lado, si se detectan cambios en las RPM, se obtienen nuevamente los datos provenientes del sistema de diagnóstico a bordo y se continúa con la telemetría del vehículo.

## 10.4 Distancia recorrida

En este caso se llevará a cabo la verificación de la función encargada de calcular la distancia recorrida entre dos posiciones GPS, utilizando para ello la Fórmula del Haversine. Esta fórmula se basa en las coordenadas de latitud y longitud de dos puntos para determinar la distancia que los separa en una esfera, tal como la Tierra.

$$a = \sin^2\left(\frac{\Delta\text{latitud}}{2}\right) + \cos(\text{latitud}1) \cdot \cos(\text{latitud}2) \cdot \sin^2\left(\frac{\Delta\text{longitud}}{2}\right)$$

$$c = 2 \cdot \text{atan}2(\sqrt{a}, \sqrt{1 - a})$$

$$\text{distancia} = R \cdot c$$

Donde:

- latitud1 y longitud1 son las coordenadas del punto de partida.
- latitud2 y longitud2 son las coordenadas del punto de llegada.
- $\Delta\text{latitud}$  y  $\Delta\text{longitud}$  son las diferencias en latitud y longitud, respectivamente.
- R es el radio de la Tierra (6,371 kilómetros).

Es importante tener en cuenta que esta fórmula asume una esfera perfecta y puede no ser completamente precisa en distancias cortas o en áreas donde la curvatura de la Tierra es significativa.

En las siguientes imágenes se presenta una comparativa de la distancia entre dos ubicaciones diferentes, empleando tanto la aplicación *Google Maps* como el procedimiento inherente al cálculo de la distancia recorrida implementado en el sistema.

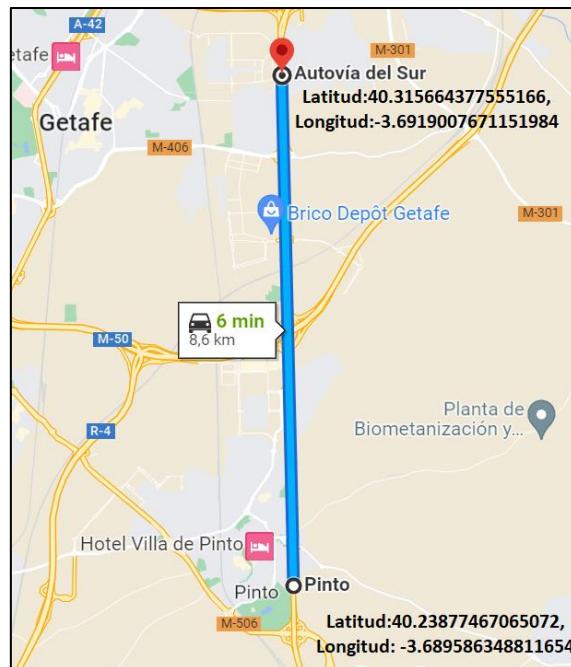


Figura 33 : Distancia Google Maps

En el procedimiento utilizado para calcular la distancia, se empleó un número de decimales equivalente al proporcionado por el módulo GPS NEO-6M.

La distancia entre los puntos:  
 $(40.238774, -3.689586)$  y  $(40.315664, -3.691900)$  es de = 8.551746 Km  
 >>

Figura 34 : Función para calcular la distancia

Dado que el vehículo se encuentra en movimiento y las vías de tránsito generalmente no son rectas, el cálculo de la distancia se realiza periódicamente en intervalos cortos de tiempo, con el objetivo de maximizar la precisión en la medición de la distancia recorrida.

## 10.5 Visualización de datos

Una vez adquiridos los datos provenientes del sistema de diagnóstico a bordo (OBD-II), se llevará a cabo el proceso de envío de la telemetría a la plataforma IoT ThingsBoard con el fin de visualizarla.

Para la visualización de los datos obtenidos, se utilizará un *dashboard* propio, los *dashboard* también conocido como tableros de instrumentos o paneles de control,

proporcionan una visión general y rápida de la información que permite a los usuarios tomar decisiones informadas y monitorear el rendimiento.

En la siguiente imagen se muestra el *dashboard*, diseñado específicamente para brindar una visión clara y concisa de los datos clave.

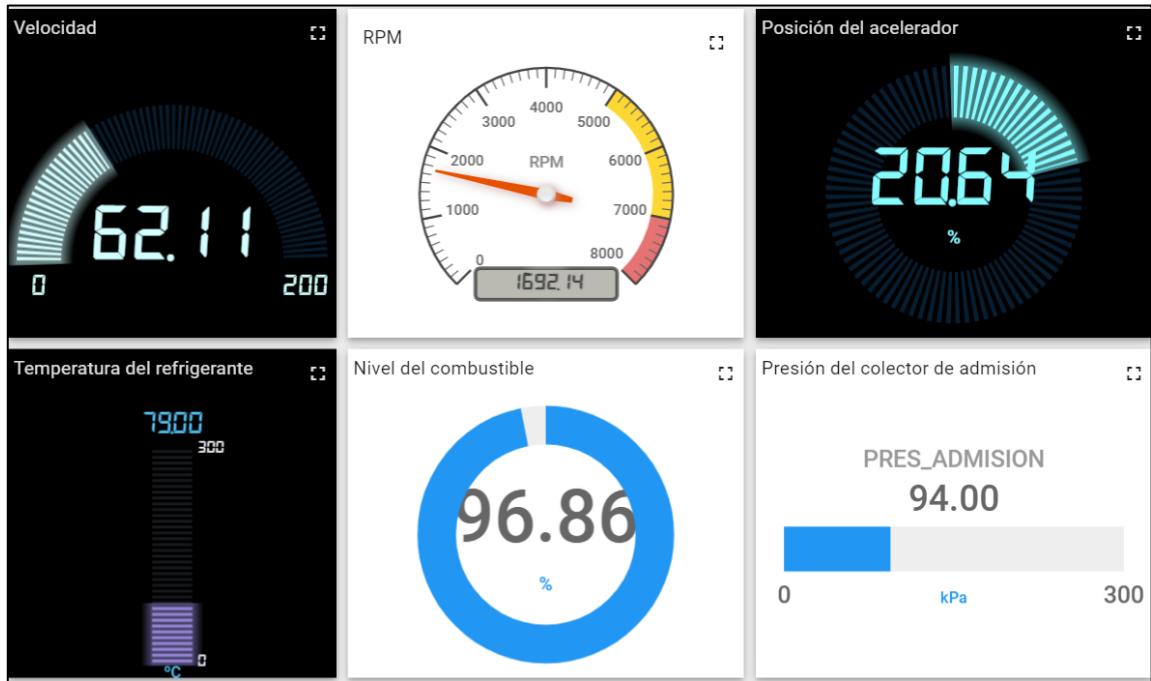


Figura 35 : Telemetría en ThingsBoard

# CAPÍTULO 11: PRUEBAS DEL SISTEMA

Las pruebas del sistema desempeñan un papel fundamental en el proceso de desarrollo de software al garantizar que el sistema en su conjunto funcione de manera correcta y cumpla con los requisitos establecidos. En esta sección, se explorará en detalle las pruebas del sistema y su importancia para asegurar la calidad y la fiabilidad del software.

Para llevar a cabo las pruebas del sistema, se utilizó un vehículo personal, al cual se le conectó el módulo ELM327 al puerto OBD-II y la parte del sistema conformada por el ESP32 y el GPS NEO-6M fue alimentada utilizando un puerto USB.

En la siguiente imagen, se puede apreciar una gráfica que muestra los datos clave agrupados, como la velocidad, la presión del colector de admisión, la temperatura del refrigerante y el nivel de combustible.

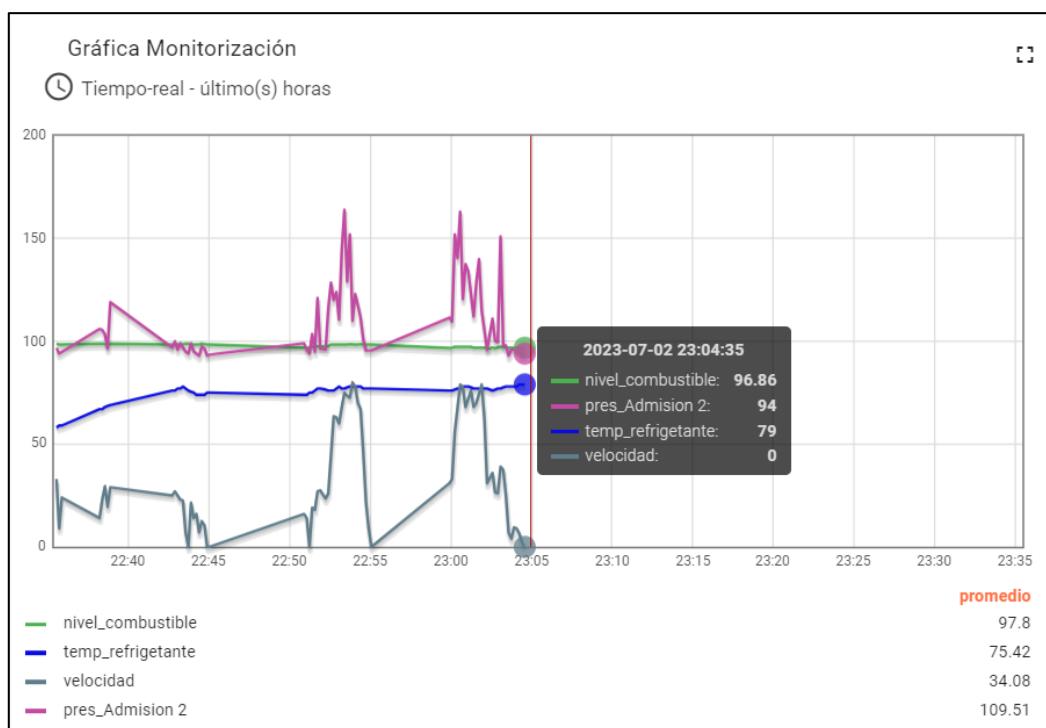
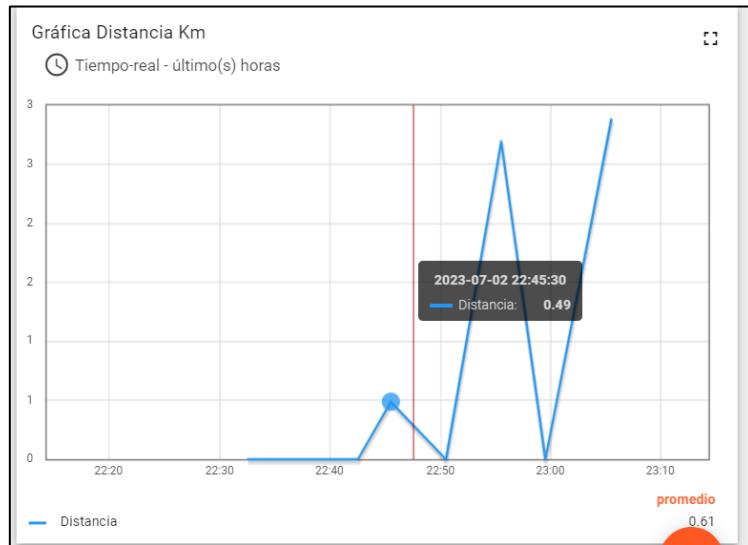


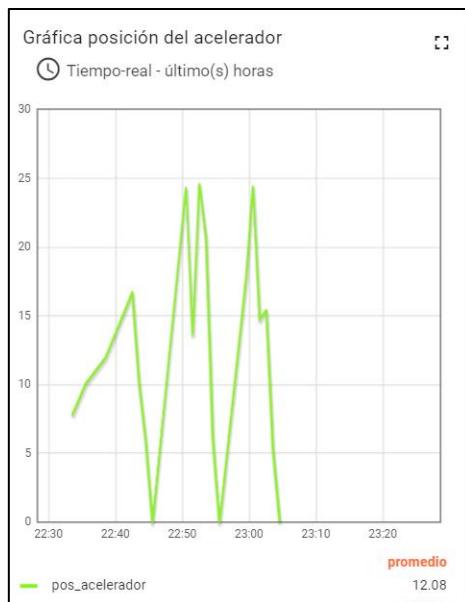
Figura 36 : Gráfica de datos en ThingsBoard

Es crucial destacar la importancia de monitorear la distancia recorrida, la cual se reinicia cada vez que se finaliza un trayecto. Este enfoque evita la mezcla de kilómetros entre distintos recorridos, proporcionando una medición precisa y clara de la distancia recorrida en cada viaje.



*Figura 37 : Gráfica distancia recorrida*

La posición del acelerador en cada momento es otro dato crucial a tener en cuenta, ya que proporciona información valiosa para el monitoreo del rendimiento del vehículo, el análisis de la eficiencia de conducción, el diagnóstico de problemas y la optimización del rendimiento del motor.



*Figura 38 : Gráfica posición del acelerador*

---

Otro dato a tener en cuenta son las revoluciones por minuto. Las RPM representan la velocidad de rotación del motor y brindan información importante sobre su rendimiento y eficiencia. En esta imagen, se muestra una visualización detallada de las RPM en tiempo real, permitiendo un monitoreo preciso y una comprensión más profunda de la dinámica del motor.

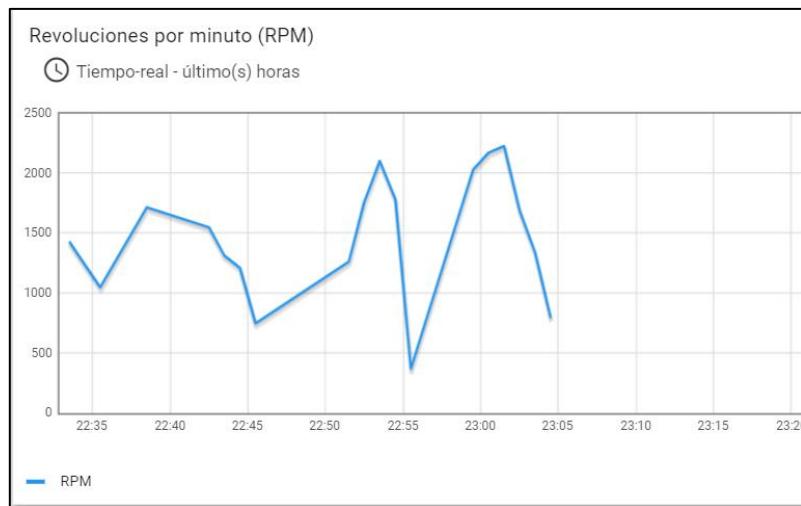


Figura 39 : Gráfica RPM

Por último, en ThingsBoard también se presenta la visualización del trayecto recorrido en un mapa. Esta funcionalidad permite ver de manera interactiva y en tiempo real la ruta seguida por el vehículo, brindando una perspectiva geográfica del recorrido. La siguiente imagen proporciona información adicional sobre la ubicación, la duración del viaje y las posibles paradas realizadas. Esta característica enriquece aún más la experiencia de monitoreo y análisis de datos en ThingsBoard.

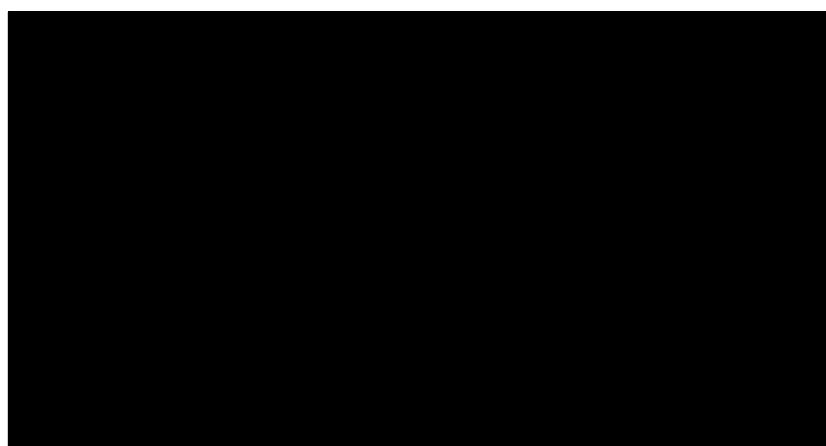


Figura 40 : Trayecto recorrido

---

Los avisos son un elemento clave en el sistema, y para garantizar una comunicación eficiente, se han integrado con la plataforma de mensajería Telegram. Esta integración permite enviar notificaciones y alertas importantes directamente a través de la aplicación Telegram, manteniendo a los usuarios informados en tiempo real sobre eventos relevantes, actualizaciones de estado y cualquier otra información crítica. La capacidad de recibir avisos rápidamente y desde cualquier ubicación mejora la capacidad de respuesta y la toma de decisiones ágil en el sistema.

La siguiente imagen proporciona una información relevante sobre el inicio de un trayecto, la distancia recorrida y la finalización de este. Además, en caso de detectar un nivel alto de refrigerante del motor, se generará una notificación de aviso. Esta medida de seguridad garantiza que los usuarios sean alertados rápidamente ante cualquier anomalía en el sistema de refrigeración. Mantener un nivel adecuado de refrigerante es fundamental para prevenir daños en el motor y garantizar un funcionamiento óptimo.



Figura 41 : Avisos Telegram

---

Es importante destacar que, debido a que las pruebas fueron realizadas con un vehículo personal, se redujo el umbral de la alerta de alta temperatura del refrigerante del motor con el fin de poder visualizar y evaluar dicha alarma en un entorno de prueba.

## 11.1 Problemas encontrados

Durante el desarrollo de este proyecto, surgieron diversos desafíos y problemas que afectaron a la capacidad para avanzar de manera eficiente. Uno de los problemas principales estuvo relacionado con la falta de información adecuada sobre el dispositivo ELM327 para establecer una conexión Bluetooth de baja energía (BLE) con el microcontrolador ESP32. La escasez de documentación y recursos disponibles dificultó la comprensión completa de cómo lograr esta conexión de manera efectiva. Fue necesario realizar una investigación exhaustiva y consultar diversas fuentes para adquirir el conocimiento necesario sobre los protocolos de comunicación, los pasos de configuración y las mejores prácticas para establecer una conexión BLE estable y confiable. También se encontraron dificultades debido al desconocimiento inicial de los estándares OBD2 (*On-Board Diagnostics 2*) y los PIDs (*Parameter IDs*).

Esto llevó a tener limitaciones en la interpretación y uso adecuado de los datos obtenidos del vehículo. Para superar este problema, fue necesario investigar y adquirir un mayor conocimiento sobre los estándares OBD2 y los PIDs específicos necesarios en el proyecto.

Otro problema significativo fue la dificultad para establecer una conexión a Internet utilizando el módulo SIM800L. Este módulo, diseñado para la comunicación celular, presentó complicaciones para conectarse a la red de telefonía móvil y acceder a Internet de manera confiable. Las dificultades encontradas incluyeron problemas de configuración, compatibilidad con los operadores de red locales y la recepción de la señal de la red móvil. Para abordar este desafío, se realizó una exhaustiva investigación de la documentación técnica del módulo y se realizaron pruebas, además de ajustar la configuración del módulo, verificar el uso de una tarjeta SIM adecuada y buscar áreas con una mejor cobertura de la señal móvil. Sin embargo, se detectaron limitaciones en la capacidad del

---

módulo SIM800L para establecer una conexión a Internet consistente y confiable. Este problema llevó a la exploración de otras soluciones, como el aprovechamiento de redes WiFi disponibles en lugar de depender exclusivamente del módulo SIM800L.

Además, durante el transcurso de las pruebas se llegó a la conclusión de que el dispositivo ESP32 no es capaz de utilizar la biblioteca Bluetooth en conjunto con la biblioteca *urequest* para la comunicación con la API de Telegram. Por lo tanto, se optó por enviar los mensajes de Telegram a través de ThingsBoard como solución alternativa.

# CAPÍTULO 12: PLANIFICACIÓN DEL PROYECTO

Este capítulo se enfoca en la planificación del proyecto desde las perspectivas temporal y económica, con el objetivo de lograr una ejecución exitosa. Se explorará en detalle cómo gestionar eficientemente los recursos temporales y financieros para optimizar los resultados.

## 12.1 Planificación temporal

En la planificación temporal de un proyecto, una herramienta invaluable es el Diagrama de Gantt. La siguiente figura captura de manera visual la secuencia de tareas, su duración y las dependencias entre ellas, lo que permite tener una visión clara del cronograma del proyecto y optimizar la asignación de recursos.

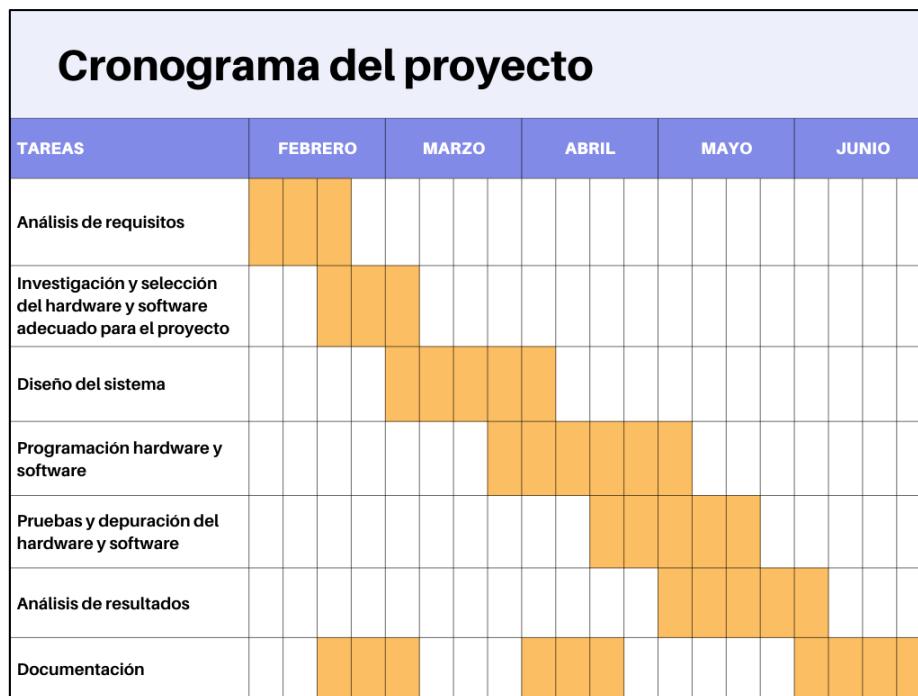


Figura 42 : Diagrama de Gantt

La imagen muestra el progreso de cada tarea a lo largo del tiempo, representado mediante barras verticales, cada barra refleja una duración aproximada de una semana.

El Diagrama de Gantt ayuda a identificar posibles cuellos de botella y retrasos en el proyecto, así como a gestionar eficientemente los recursos asignados. Al visualizar las fechas de inicio y finalización de cada tarea, se pueden tomar decisiones informadas sobre la asignación de recursos, prioridades y ajustes necesarios para cumplir con los plazos establecidos.

## 12.2 Planificación económica

En este proyecto se ha valorado la utilización de dispositivos de bajo coste para la implementación del sistema. Esto ha permitido reducir significativamente los costes de producción, lo cual es beneficioso tanto desde el punto de vista económico como en términos de accesibilidad. Los dispositivos utilizados, como el ESP32, el módulo GPS Neo-6M y el dispositivo ELM327 mini, se pueden encontrar fácilmente en el mercado y se caracterizan por su asequibilidad. Además, han demostrado ser eficientes y confiables para el propósito del sistema de monitorización de sensores del coche, dichos dispositivos han sido capaces de proporcionar los datos necesarios de manera precisa y consistente, lo que ha permitido maximizar el valor del sistema logrando un equilibrio entre funcionalidad y costo. Por otra parte, al utilizar dispositivos ampliamente disponibles, se facilita la escalabilidad del sistema.

A continuación, se presenta una tabla detallada que desglosa los precios de los productos que forman parte del proyecto.

Concepto	Unidades	Precio (€)
ESP32	1	11,49
GPS Neo-6M	1	9,99
ELM327 mini	1	11,99
Kit Cables de Puente Jumper	40	5,49
Placas Circuito Impreso PCB	40	16,49

Tabla 6 : Precio de los componentes del sistema

El sistema completo tiene un coste de 55,55 €, lo cual lo convierte en una opción altamente accesible para la mayoría de las personas. Además de esto hay que añadir el coste de desarrollo, el salario de un ingeniero informático promedio en España es de 26.500 € al año o 13,59 € por hora (Talent, 2023), considerando que el proyecto se realizó durante 5 meses y el tiempo diario aproximado fue de 5 horas, el precio total de desarrollo ha sido de 6.795 €



# CAPÍTULO 13: IMPACTO SOCIAL Y MEDIOAMBIENTAL

La implementación de un proyecto de monitorización de sensores de un coche puede tener un impacto social y medioambiental significativo.

En primer lugar, la seguridad vial puede mejorar significativamente mediante la monitorización de sensores en un coche. Al recopilar datos en tiempo real sobre parámetros como la velocidad, la temperatura del motor y los niveles de combustible, es posible identificar posibles problemas y advertir al conductor sobre situaciones de riesgo. Esto no solo contribuye a reducir el riesgo de accidentes, sino que también promueve una conducción más segura y responsable.

Además, la monitorización de sensores ofrece la posibilidad de implementar un mantenimiento preventivo en el coche. Al detectar y diagnosticar problemas mecánicos o de funcionamiento en etapas tempranas, es posible tomar medidas correctivas antes de que se conviertan en problemas mayores. Esto no solo evita averías costosas, sino que también prolonga la vida útil del vehículo, reduciendo el desperdicio de recursos al minimizar la necesidad de reparaciones importantes o reemplazos prematuros.

La eficiencia energética es otro aspecto clave que se puede abordar a través de la monitorización de sensores en un coche. Al analizar y optimizar los datos de consumo de combustible, por ejemplo, se pueden identificar patrones de conducción ineficientes y brindar retroalimentación al conductor para fomentar hábitos de conducción más económicos. Esto contribuye a reducir el consumo de combustible y, por lo tanto, las emisiones de gases de efecto invernadero, lo que tiene un impacto positivo en la mitigación del cambio climático y la mejora de la calidad del aire.

Por último, un proyecto de monitorización de sensores de coches puede generar conciencia y educación en los conductores sobre la importancia de una conducción segura,

---

responsable y sostenible. Al proporcionar retroalimentación y datos relevantes, se empodera a los conductores para tomar decisiones más informadas y adoptar comportamientos que reduzcan el impacto ambiental. Esta concienciación y educación no solo beneficia a nivel individual, sino que también tiene un impacto positivo en la sociedad en general.

## CAPÍTULO 14: CONCLUSIONES

Tras la finalización del trabajo, se pueden extraer varias conclusiones en función de los objetivos propuestos. A lo largo del proyecto, se logró diseñar y configurar el sistema de monitorización, permitiendo la recopilación de datos en tiempo real de diversos parámetros del coche, como la velocidad, la temperatura del motor y los niveles de combustible. Estos datos fueron transmitidos de manera inalámbrica al ESP32 y enviados a la plataforma ThingsBoard para su visualización y análisis. La utilización del dispositivo ELM327 simplificó la conexión y comunicación con el sistema de diagnóstico a bordo del vehículo, lo que facilitó la obtención de información relevante sobre su estado y rendimiento. Además, esta comunicación se realizó de manera eficiente, con un consumo reducido gracias a la tecnología Bluetooth Low Energy. La integración del GPS NEO6M en el sistema proporcionó datos de geolocalización precisos, lo que permitió el seguimiento y registro de la ruta recorrida por el coche. La plataforma ThingsBoard demostró ser una herramienta poderosa para la visualización y análisis de los datos recopilados. Proporcionó una interfaz intuitiva y personalizable, lo que permitió a los usuarios acceder de manera fácil y eficiente a la información del coche.

Este proyecto de monitorización de vehículos pretende mejorar la seguridad vial al detectar posibles problemas y advertir al conductor sobre situaciones de riesgo, además de fomentar el mantenimiento preventivo al detectar y diagnosticar problemas mecánicos o de funcionamiento antes de que se convirtieran en fallas importantes. La combinación de dispositivos de bajo coste, como el ELM327 y el ESP32, junto con la plataforma ThingsBoard, ha permitido obtener una solución económica y funcional.



## CAPÍTULO 15: LÍNEAS FUTURAS

En el futuro sería importante considerar las posibles líneas futuras de investigación y desarrollo para mejorar y ampliar el sistema de monitorización de sensores de un coche. Una línea interesante de investigación puede consistir en explorar técnicas y algoritmos que permitan mejorar el tiempo de respuesta del sistema. Esto podría lograrse mediante la implementación de un esquema de programación paralela eficiente.

Para aumentar la seguridad de los conductores, se podría investigar la integración de funcionalidades de llamadas de emergencia. Esto implicaría el desarrollo de un mecanismo que, en situaciones críticas detectadas por los sensores, activaría automáticamente una llamada de emergencia hacia servicios de asistencia o contactos predefinidos.

Además de los sensores internos del coche, se podría investigar la incorporación de sensores externos adicionales. Estos podrían ser sensores de temperatura ambiental, humedad, calidad del aire exterior o acelerómetros entre otros. La integración de estos sensores ampliaría la cantidad de información disponible y proporcionaría una visión más completa del entorno del vehículo. Estas líneas futuras de investigación y desarrollo suponen oportunidades para mejorar y expandir el sistema de monitorización de sensores de un coche, brindando mayores funcionalidades y beneficios tanto en términos de seguridad vial como de aprovechamiento de los datos recopilados.



## APÉNDICE A. REFERENCIAS

Amazon. (2023). *TECNOIOT LORA SX1278*. Obtenido de <https://www.amazon.es>

Archivo Digital. (2023). Obtenido de <https://oa.upm.es/>

AUTODOC. (4 de Marzo de 2023). *OBD2 - Diagnóstico de errores del coche*. Obtenido de <https://club.autodoc.es/magazin/obd2-diagnostico-de-errores-del-coche>

AZ-Delivery. (20 de Abril de 2023). *ESP32 NodeMCU Módulo WLAN WiFi Development Board con CP2102*. Obtenido de <https://www.az-delivery.de/es/products/esp32-developmentboard>

AZ-Delivery. (9 de Abril de 2023). *SIM800L GSM GPRS*. Obtenido de <https://www.az-delivery.de>

Bluetooth. (2 de Abril de 2023). *Bluetooth Low Energy*. Obtenido de <https://mediac.industry.panasonic.eu>

Bluetooth. (2023). *low-level Bluetooth*. Obtenido de <https://docs.micropython.org/en/latest/library/bluetooth.html>

BMW CarData . (2023). *BMW CarData* . Obtenido de <https://www.bmw.es/es/topics/mundo-bmw/connected-drive/bmw-cardata.html>

Circuitdigest. (22 de Marzo de 2023). *How to Program ESP32 in MicroPython using Thonny IDE*. Obtenido de <https://circuitdigest.com/microcontroller-projects/how-to-program-esp32-in-micropython-using-thonny-ide>

CSS Electronics. (25 de Marzo de 2023). *OBD2 Explained*. Obtenido de <https://www.csselectronics.com/pages/obd2-explained-simple-intro>

Electrodaddy. (2023). *Módulos Esp32 con Wi-Fi / Bluetooth: Guía de introducción*. Obtenido de <https://www.electrodaddy.com/esp32/>

---

## Apéndice A. Referencias

---

Elm Electronic . (2023). *ELM327 OBD to RS232 Interpreter*. Obtenido de [www.elmelectronics.com](http://www.elmelectronics.com)

GitHub. (2023). *ThingsBoard-python-client-sdk*. Obtenido de <https://github.com/thingsboard/thingsboard-python-client-sdk>

Google Patents. (2023). *Google Patents*. Obtenido de <https://patents.google.com/>

Gotoiot. (29 de Marzo de 2023). *Intro a Bluetooth Low Energy*. Obtenido de <https://www.gotoiot.com>

Herramienta Automotriz. (25 de Marzo de 2023). *Herramienta automotriz - Mejores adaptadores y aplicaciones ELM327*. Obtenido de <https://herramientaautomotriz.com/elm327/>

HOMYHUB. (15 de Marzo de 2023). *5 ventajas del coche conectado*. Obtenido de <https://homyhub.com/blog/5-ventajas-del-coche-conectado/>

Hoyas, V. L. (2015). *Desarrollo Android para el control de dispositivos acoplados a un automóvil*. UEX.

Mathworks. (2023). *Read Serial Data from GPS Shield Using Arduino Hardware*. Obtenido de <https://es.mathworks.com/help/supportpkg/arduino/ref/read-serial-data-from-a-gps-shield-using-arduino-hardware.html>

Mercedes Me Connect . (2023). *Mercedes Me Connect* . Obtenido de <https://www.mercedes-benz.es/passengercars/services/mercedes-me.html>

MQTT. (27 de Marzo de 2023). *MQTT The Standard for IoT Messaging*. Obtenido de <https://mqtt.org/>

Mula, J. A. (2021). *Diseño y desarrollo de un sistema de diagnóstico de vehículos*. UPTC.

Rs-online. (2023). *Wisol SFM10R1*. Obtenido de <https://es.rs-online.com/web/p/kits-desarrollo-inalambricos-y-de-comunicacion/1365801>

---

Santos, P. R. (17 de Mayo de 2023). *Breve historia de Internet de las cosas (IoT)*.

Obtenido de <https://empresas.blogthinkbig.com/breve-historia-de-internet-de-las-cosas-iot/>

Santos-Olmo, A. B. (2022). *Smartceta, sistema de medición de valores básicos de una planta utilizando IoT*. ETSISI.

Sumador. (19 de Marzo de 2023). *Módulo GPS UBLOX NEO-6M con antena*. Obtenido de <https://sumador.com/collections/otros-modulos/products/modulo-gps-neo-6m>

Talent. (2023). *Salario medio para Ingeniero Informatico en España, 2023*. Obtenido de <https://es.talent.com/salary?job=ingeniero+informatico>

Telegram. (8 de Abril de 2023). *Telegram APIs*. Obtenido de <https://core.telegram.org/>

ThingsBoard. (2023). *Notifications and Alarms on your smartphone using Telegram*. Obtenido de <https://thingsboard.io/docs/user-guide/rule-engine-2-0/tutorials/integration-with-telegram-bot/>

ThingsBoard. (26 de Marzo de 2023). *ThingsBoard Open source IoT Platform*. Obtenido de <https://thingsboard.io/>

Waveshare. (2023). *SIM7600E-H 4G HAT*. Obtenido de [https://www.waveshare.com/wiki/SIM7600E-H\\_4G\\_HAT](https://www.waveshare.com/wiki/SIM7600E-H_4G_HAT)



## APÉNDICE B. GLOSARIO

### A

ADC

*Son puertos que permiten convertir señales analógicas en señales digitales,* 33

Android

Sistema operativo móvil basado en Linux, 21

Android studio

Es un entorno de desarrollo integrado oficial para el desarrollo de aplicaciones móviles en el sistema operativo

Android, 22

API

*Cconjunto de reglas y protocolos que permite a diferentes aplicaciones y sistemas intercambiar información y comunicarse entre sí,* 22

Archivo Digital

Repositorio digital de la Universidad Politécnica de Madrid, 25

Arduino IDE

Es un entorno de programación utilizado para programar y desarrollar proyectos basado en C++, 20

Arduino UNO

Es una placa de desarrollo de hardware de código abierto basada en microcontrolador, 21

ARM Cortex-M3

Procesador de la arquitectura ARM diseñado específicamente para aplicaciones embebidas, 41

ATT

*Acrónimo de Attribute Protocol,* 45

### B

biblioteca urequest

*Biblioteca implementada en MicroPython para realizar solicitudes HTTP,* 78

Bluetooth

Tecnología inalámbrica de corto alcance que permite la comunicación y transferencia de datos entre dispositivos electrónicos, 20

Bluetooth HC-05

Módulo inalámbrico que permite la comunicación por Bluetooth entre dispositivos, 21

BotFather, 62

### C

C++

Lenguaje de programación de alto nivel y generalmente orientado a objetos, 20

CAN

## Apéndice B. Glosario

---

Protocolo de comunicación serial utilizado principalmente en aplicaciones automotrices e industriales, 22

CoAP

Acrónimo de Constrained Application Protocol, es un protocolo de mensajería ligero y eficiente diseñado para dispositivos con recursos limitados y conexiones de red no confiables, 42

### D

DAC

Son puertos que permiten convertir señales digitales en señales analógicas, 33

deepsleep

Modo ahorro de energía del ESP32, 30

DTC

Es un código de diagnóstico que se utiliza en los vehículos para identificar y registrar problemas o fallas en los sistemas electrónicos del vehículo, 37

### E

ECU

Unidad de control del motor que supervisa y controla varios componentes y sistemas del motor de un vehículo, 21

EDGE

Acrónimo de Enhanced Data rates for GSM Evolution, es una tecnología de comunicación de datos móviles que se utiliza en redes móviles de segunda generación (2G), 49

EEPROM

Memoria de sólo lectura programable y borrable eléctricamente, 38

Elm Electronics

Es una empresa especializada en el desarrollo y producción de componentes y dispositivos electrónicos, 35

ELM327

Es un dispositivo que se utiliza para leer datos y códigos de diagnóstico a través del puerto OBD-II (On-Board Diagnostics) de un automóvil , 9

ESP-32

Microcontrolador de bajo consumo de energía y alto rendimiento, 9

ESP8266

Microcontrolador y módulo de conectividad Wi-Fi desarrollado por Espressif Systems, 33

Espressif Systems

Empresa de tecnología con sede en China, especializada en el diseño y desarrollo de chips, 33

### F

Fórmula del Haversine

Fórmula que determina la distancia entre dos puntos en una esfera, 70

G

GAP

Acrónimo de Generic Access Profile, 45

GATT

Acrónimo de Generic Attribute Profile, 45

GND

Puerto que se conecta a tierra, 38

GPIO

Son pines de propósito general presentes en muchos microcontroladores y sistemas embebidos, 41

GPRS

Acrónimo de General Packet Radio Service, es una tecnología de comunicación de datos móviles que se utiliza en redes móviles de segunda generación (2G), 49

GSM

Acrónimo de Global System for Mobile Communications es un estándar de comunicación inalámbrica utilizado para la transmisión de voz y datos en redes móviles, 49

H

HCI

Acrónimo de Host Controller Interface, 45

HTTPS

Acrónimo de Hypertext Transfer Protocol Secure, es utilizado para la transferencia de información en la web, 42

I

I2C

Es un bus de comunicación de dos hilos utilizado para conectar varios dispositivos electrónicos, 33

IoT

Siglas de Internet of Things, Internet de las Cosas traducido al español, 9

ISO 14230-4 KWP2000

Estándar de comunicación utilizado en la industria automotriz para la comunicación electrónica, 36

ISO 15765-4 CAN-BUS

Estándar de comunicación utilizado en la industria automotriz para la comunicación electrónica, 36

ISO 9141-2

Estándar de comunicación utilizado en la industria automotriz para la comunicación electrónica, 35

J

J1850 PWM

Protocolo de comunicación utilizada en el ámbito automotriz con Modulación por ancho de Pulso, 35

J1850 VPW

Protocolo de comunicación utilizada en el ámbito automotriz con Modulación Variable, 35

## Apéndice B. Glosario

---

### JavaScript

Lenguaje de programación de alto nivel, interpretado y orientado a objetos, 33

### JSON

Acrónimo de JavaScript Object Notation, es un formato de intercambio de datos ligero y legible, 63

L

### L2CAP

Acrónimo de Logical Link Control and Adaptation Protocol, 45

Li

### LL

Acrónimo de Link Layer, 45

L

### LoRa

es una tecnología de comunicación inalámbrica de largo alcance, 51

M

### MCP2551

Es un controlador de transceptor CAN, 22

### MEMS

Son dispositivos miniaturizados que combinan componentes electrónicos y mecánicos en una sola estructura, 19

### MicroPython

Interprete de Python enfocado en microcontroladores, 9

### MQTT

Acrónimo de Message Queuing Telemetry Transport, es un protocolo de mensajería ligero y eficiente diseñado para dispositivos con recursos limitados y conexiones de red no confiables., 42

N

### NMEA, 58

Acrónimo de National Marine Electronics Association, que es una asociación estadounidense que establece estándares para la comunicación entre dispositivos electrónicos utilizados en la navegación marítima, 38

O

### OBD-II

Es un estándar de diagnóstico de vehículos que permite acceder y monitorear datos del rendimiento y estado del automóvil a través de un puerto de conexión., 9

OSI

Open Systems Interconnection es un modelo de referencia utilizado para describir y entender cómo funciona la comunicación de red entre diferentes sistemas informáticos, 44

P

PHY

Acrónimo de *Physical Layer*, 45

PID

Acrónimo de *Parameter IDentification*), 67

PostgreSQL

Sistema de gestión de bases de datos relacional y de código abierto, 22

PWM

Son puertos que generan señales de salida con una anchura de pulso variable, 33

Python

Lenguaje de programación interpretado, de alto nivel, 22

Q

QoS

Acrónimo de Quality of Service, Calidad de Servicio en español y se refiere a la capacidad de una red o sistema para ofrecer un nivel de rendimiento, 43

R

Raspberry Pi

Serie de ordenadores de placa única, 22

RPM

Revoluciones por minuto, 67

RX

Puerto de recepción, 38

S

SAE J1979

Es un estándar de comunicación utilizado en la industria automotriz para la comunicación electrónica entre los sistemas de control de un vehículo, 37

Sensor DHT-11

Sensor de temperatura y humedad que proporciona mediciones precisas de ambos parámetros., 23

Sensor MQ7

Sensor de gas que detecta la presencia de monóxido de carbono (CO) en el aire., 22

SIG

## Apéndice B. Glosario

---

**Bluetooth SIG**  
Abreviatura de Bluetooth Special Interest Group, es una organización de estándares que supervisa el desarrollo de los estándares de Bluetooth, 44

**Sigfox**

*es un operador de red global fundado en 2010, 52*

**SM**

*Acrónimo de Security Manager, 45*

**SoC**

**Soc**  
Siglas en inglés System-on-a-Chip, es un tipo de chip integrado que combina múltiples componentes de hardware, 33

**SPI**

*Es un protocolo de comunicación síncrono utilizado para la transferencia de datos entre dispositivos electrónicos, 33*

**Start-Stop**

*Sistema de parada y arranque automático del motor, 69*

## T

---

**TDLib**

Biblioteca de bases de datos de Telegram que permite a los desarrolladores de terceros crear sus propios clientes de Telegram, 48

**Telegram**

Aplicación de mensajería instantánea, 20

**Tensilica Xtensa LX6**

Arquitectura de procesador diseñada por Tensilica, 33

**ThingsBoard**

Plataforma IoT de código abierto, 9

**TTL**

Es un estándar de señal eléctrica utilizado en electrónica digital, 38

**TX**

Puerto de transmisión, 38

## U

---

**UART**

*Es un protocolo de comunicación asíncrono utilizado para la transmisión y recepción de datos serie, 33*

**UFL**

Son conectores coaxiales de alta frecuencia que permiten la conexión entre la antena GPS y el módulo receptor GPS dentro del dispositivo, 38

**UUID**

*Identificador Único Universal o en inglés Universally Unique Identifier, 47*

V

VCC

Puerto de alimentación, 38

VIN

Número de identificación único que se asigna a un vehículo, 37

W

*Widgets*, 59

WiFi

Tecnología de comunicación inalámbrica que permite la conexión a internet y la transmisión de datos mediante ondas de radio, 20



## APÉNDICE C: MATERIAL ENTREGADO A MOODLE Y CÓDIGO

El material entregado a Moodle consta de un archivo comprimido en formato *.zip* cuyo nombre es Proyecto, dicho fichero está compuesto por la memoria en formato *.pdf*, por el código principal con el nombre *Proyecto.py* y por las librerías diseñadas para la realización del proyecto.

La librería desarrollada para la comunicación Bluetooth Low Energy entre el ESP32 y el ELM327 es la nombrada como *BLE\_ELM327\_LIB.py* y la librería capaz de obtener los datos GPS utilizando el módulo GPS NEO-6M es la llamada *GPS\_NEO6M\_LIB.py*.

En código también puede verse en la plataforma GitHub utilizando el siguiente enlace <https://github.com/diieego97/Monitorizacion-de-vehiculos-OBD-II>.

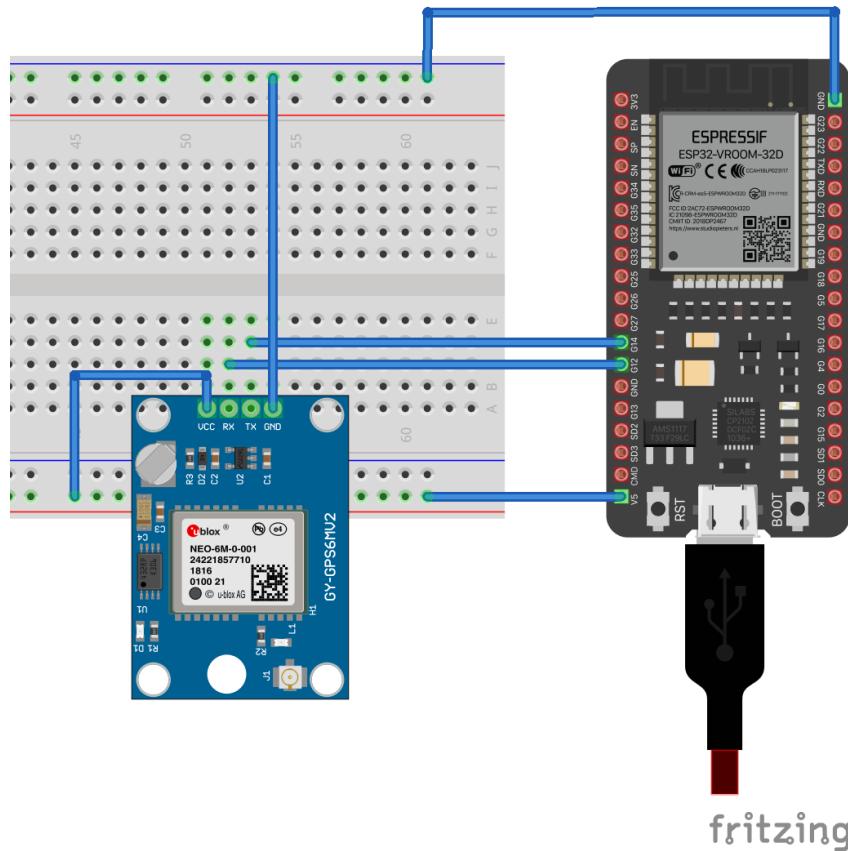


## APÉNDICE D. IMÁGENES DEL PROYECTO

En este apéndice, se presentan imágenes relevantes del diseño y la implementación del circuito desarrollado en este proyecto.

### Diseño del circuito

En la siguiente figura se muestra el diseño esquemático del circuito. Esta representación gráfica proporciona una visión clara de las conexiones eléctricas y los componentes utilizados en el circuito.



## Circuito real

En la siguiente imagen se puede observar cómo se han unido los diferentes componentes en una placa de circuito impreso (PCB). La soldadura se ha realizado de manera precisa y se ha verificado la correcta conexión de cada componente.

