

Grado Universitario en Ingeniería Informática  
Curso Académico 2024-2025

*Trabajo Fin de Grado*

“Diseño de un salpicadero virtual de un vehículo en un PC usando microcontrolador”

Jose Crespo Cantador

Tutor

Jose Enrique Suarez Pascual

Colmenarejo, 08 de Septiembre de 2025



Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**

**RESUMEN**

El trabajo recogido en este documento tiene como objetivo el diseño e implementación de un salpicadero virtual en el que representar una colección de datos obtenidos a través de un microcontrolador.

La intención es que la comunicación entre el microcontrolador y el sistema que represente el salpicadero ocurra de manera alámbrica. Los datos que habrá que representar serán obtenidos a través de sensores conectados al microcontrolador. Dichos datos serán interpretados por el mencionado microcontrolador, y al ser recibidos por el sistema que contiene el salpicadero, serán tratados para adaptarse de forma correcta a la representación.

El proyecto pretende demostrar la posibilidad de desarrollar una solución contenida y asequible, siendo esta solución un posible punto de partida para el desarrollo de trabajos futuros.

**Palabras clave**

Comunicación Alámbrica, Microcontrolador, Salpicadero Virtual, Termistor, Sensor Fotoeléctrico

**AGRADECIMIENTOS**

Quiero empezar agradeciendo a mis padres todo el cariño y apoyo que me han dado a lo largo de los años, no solo durante mi carrera universitaria. Sin ellos esto no habría sido posible y no estaría aquí ahora mismo.

También agradecer a mis amigos, con los que he compartido el grado universitario y mis amigos de toda la vida, que siempre me han sacado una sonrisa.

A mi pareja, Ainoa que me ha aguantado, soportado, motivado y querido durante todo el tiempo que llevamos juntos. Sin ella no se si habría sido capaz de pasar por todo lo que he pasado, apoyándome en los días buenos y en los malos.

A todos los profesores que han dedicado su esfuerzo a transmitirme los conocimientos necesarios que hoy uso en mi día a día, tanto en lo profesional como en lo personal.

También a mi tutor José Enrique Suárez Pascual que, aunque no hemos tenido las conversaciones más profundas, siempre ha contestado a mis dudas con gran agilidad.

A todos ellos, muchísimas gracias.

**ÍNDICE DE CONTENIDOS**

[1 1. INTRODUCCIÓN 1](#_Toc208073918)

[1.1 Motivación del Trabajo. 1](#_Toc208073919)

[1.2 Objetivos 1](#_Toc208073920)

[1.2.1 Microcontrolador 1](#_Toc208073921)

[1.2.2 Conexión Alámbrica 1](#_Toc208073922)

[1.2.3 Salpicadero Virtual 1](#_Toc208073923)

[1.3 Marco regulador 2](#_Toc208073924)

[1.4 Estructura del documento 2](#_Toc208073925)

[2 ESTADO DEL ARTE 3](#_Toc208073926)

[2.1 Sensores 3](#_Toc208073927)

[2.1.1 Revoluciones por minuto 3](#_Toc208073928)

[2.1.2 Velocidad y distancia recorrida 4](#_Toc208073929)

[2.1.3 Temperatura 4](#_Toc208073930)

[2.1.4 Nivel de combustible 5](#_Toc208073931)

[2.2 Salpicaderos 5](#_Toc208073932)

[2.2.1 Mazda3 5](#_Toc208073933)

[2.2.2 Tesla Model 3 8](#_Toc208073934)

[2.2.3 Mercedes Clase C 9](#_Toc208073935)

[3 DESARROLLO DEL SISTEMA 11](#_Toc208073936)

[3.1 Requisitos 11](#_Toc208073937)

[3.2 Componentes 11](#_Toc208073938)

[3.2.1 Microcontrolador 12](#_Toc208073939)

[3.2.2 Sensor de ultrasonidos 13](#_Toc208073940)

[3.2.3 Sensor de temperatura 14](#_Toc208073941)

[3.2.4 Sensor fotoeléctrico 15](#_Toc208073942)

[3.2.5 Sensor de aceleración 16](#_Toc208073943)

[3.2.6 Potenciómetro 16](#_Toc208073944)

[3.3 Diseño del salpicadero 17](#_Toc208073945)

[4 IMPLEMENTACIÓN 19](#_Toc208073946)

[4.1 Microcontrolador 19](#_Toc208073947)

[4.1.1 Comunicación alámbrica 19](#_Toc208073948)

[4.1.2 Sensores de proximidad 20](#_Toc208073949)

[4.1.3 Sensores de temperatura 22](#_Toc208073950)

[4.1.4 Sensor de aceleración 23](#_Toc208073951)

[4.1.5 Sensor fotoeléctrico 24](#_Toc208073952)

[4.1.6 Potenciómetro 25](#_Toc208073953)

[4.1.7 Diagrama de conexiones 25](#_Toc208073954)

[4.2 Salpicadero digital 26](#_Toc208073955)

[4.2.1 Dial de rpm y velocidad 27](#_Toc208073956)

[4.2.2 Indicador de distancia de seguridad 30](#_Toc208073957)

[4.2.3 Cuadro de información del vehículo 32](#_Toc208073958)

[4.2.4 Comunicación alámbrica 36](#_Toc208073959)

[4.2.5 Función principal 37](#_Toc208073960)

[4.3 Resultados 39](#_Toc208073961)

[5 ANÁLISIS DEL PROYECTO 43](#_Toc208073962)

[5.1 Contexto socioeconómico 43](#_Toc208073963)

[5.2 Presupuesto 43](#_Toc208073964)

[5.2.1 Coste de elementos electrónicos 43](#_Toc208073965)

[5.2.2 Coste de licencias 44](#_Toc208073966)

[5.2.3 Coste de personal 44](#_Toc208073967)

[5.2.4 Coste total 44](#_Toc208073968)

[6 CONCLUSIONES 46](#_Toc208073969)

[6.1 Líneas futuras de trabajo 46](#_Toc208073970)

[7 REFERENCIAS BIBLIOGRÁFICAS 48](#_Toc208073971)

**ÍNDICE DE FIGURAS**

[Figura 2.1 Bosch Inductive Crankshaft Position Sensor [3] 3](#_Toc208073972)

[Figura 2.2 Bosch Wheel Speed Sensor [5] 4](#_Toc208073973)

[Figura 2.3 Sensor de Temperatura NTC M12 Bosch [7] 4](#_Toc208073974)

[Figura 2.4 Sensor de nivel de combustible Ford E150 [9] 5](#_Toc208073975)

[Figura 2.5 Tablero de instrumentos Mazda3 [11] 6](#_Toc208073976)

[Figura 2.6 Tacómetro Mazda3 [11] 6](#_Toc208073977)

[Figura 2.7 Visualizador de datos múltiples Mazda3 [11] 7](#_Toc208073978)

[Figura 2.8 Medidor de temperatura del refrigerante del motor Mazda3 [11] 7](#_Toc208073979)

[Figura 2.9 Medidor de combustible Mazda3 [11] 7](#_Toc208073980)

[Figura 2.10 Visualización de distancia para el estacionamiento [12] 8](#_Toc208073981)

[Figura 2.11 Panel de instrumentos Tesla Model 3 [13] 9](#_Toc208073982)

[Figura 2.12 Pantalla táctil Tesla Model 3 [14] 9](#_Toc208073983)

[Figura 2.13 Visualizador del conductor Mercedes Clase C [15] 10](#_Toc208073984)

[Figura 3.1 Vista de perspectiva de la placa STM32F407VE [16] 12](#_Toc208073985)

[Figura 3.2 Microcontroladora STM32F407G-DISC1 [17] 12](#_Toc208073986)

[Figura 3.3 Sensor de ultrasonidos HC-SR04 [18] 13](#_Toc208073987)

[Figura 3.4 Diagrama de tiempos sensor HC-SR04 [19] 13](#_Toc208073988)

[Figura 3.5 Sensor de temperatura PT100 [22] 14](#_Toc208073989)

[Figura 3.6 Conversor de señal RTD a digital [23] 15](#_Toc208073990)

[Figura 3.7 Sensor fotoeléctrico Daokai LM393 [26] 15](#_Toc208073991)

[Figura 3.8 Rueda perforada con 20 ranuras [27] 16](#_Toc208073992)

[Figura 3.9 Vista de los ejes de detección de aceleración [28] 16](#_Toc208073993)

[Figura 3.10 potenciómetro rotatorio [29] 17](#_Toc208073994)

[Figura 3.11 Prototipo del diseño del salpicadero 18](#_Toc208073995)

[Figura 4.1 Conversor USB a TTL UART [32] 20](#_Toc208073996)

[Figura 4.2 Declaración del buffer para el envío de datos 20](#_Toc208073997)

[Figura 4.3 Código encargado de la comunicación serie 20](#_Toc208073998)

[Figura 4.4 Código encargado de la funcionalidad del sensor HC-SR04 22](#_Toc208073999)

[Figura 4.5 Código encargado de la lectura del sensor pt100 23](#_Toc208074000)

[Figura 4.6 Implementación del filtro 24](#_Toc208074001)

[Figura 4.7 Código encargado de calcular el valor de las rpm 24](#_Toc208074002)

[Figura 4.8 Código encargado de la obtención del valor del potenciómetro 25](#_Toc208074003)

[Figura 4.9 Esquema de pines STM32F407 25](#_Toc208074004)

[Figura 4.10 Diagrama de conexiones 26](#_Toc208074005)

[Figura 4.11 Constructor de la clase GaugeCanvas 27](#_Toc208074006)

[Figura 4.12 Método draw\_gauge de la clase GaugeCanvas 29](#_Toc208074007)

[Figura 4.13 Método draw\_needle de la clase GaugeCanvas 29](#_Toc208074008)

[Figura 4.14 Dial revoluciones por minuto del salpicadero digital 29](#_Toc208074009)

[Figura 4.15 Dial de velocidad del salpicadero digital 30](#_Toc208074010)

[Figura 4.16 Constructor de la clase DistanceCanvas 30](#_Toc208074011)

[Figura 4.17 Método draw\_scene de la clase DistanceCanvas 31](#_Toc208074012)

[Figura 4.18 Método update\_distances de la clase DistanceCanvas 31](#_Toc208074013)

[Figura 4.19 Método draw\_arc\_distnaces de la clase DistanceCanvas 32](#_Toc208074014)

[Figura 4.20 Indicador de distancia de seguridad del salpicadero digital 32](#_Toc208074015)

[Figura 4.21 Constructor de la clase RectangleDashboardCanvas 33](#_Toc208074016)

[Figura 4.22 Métodos de la clase RectangleDashboardCanvas 33](#_Toc208074017)

[Figura 4.23 Indicador de distancia recorrida del Cuadro de información 34](#_Toc208074018)

[Figura 4.24 Indicador de temperatura del habitáculo del Cuadro de información 34](#_Toc208074019)

[Figura 4.25 Constructor de la clase Gauge 34](#_Toc208074020)

[Figura 4.26 Método draw\_gauge de la clase Gauge 35](#_Toc208074021)

[Figura 4.27 Método draw\_tick de la clase Gauge 35](#_Toc208074022)

[Figura 4.28 Método update de la clase Gauge 36](#_Toc208074023)

[Figura 4.29 Dial de nivel de combustible del Cuadro de información 36](#_Toc208074024)

[Figura 4.30 Dial de temperatura del motor del Cuadro de información 36](#_Toc208074025)

[Figura 4.31 Declaración del puerto serie 36](#_Toc208074026)

[Figura 4.32 Función process\_data 37](#_Toc208074027)

[Figura 4.33 Declaración de los elementos de la interfaz 38](#_Toc208074028)

[Figura 4.34 Salpicadero digital completo 39](#_Toc208074029)

[Figura 4.35 Montaje de sensor fotoeléctrico y motor DC 39](#_Toc208074030)

[Figura 4.36 Posición inicial del vehículo radiocontrol conectado al PC 40](#_Toc208074031)

[Figura 4.37 Montaje del microcontrolador sobre el vehículo radiocontrol 40](#_Toc208074032)

[Figura 4.38 Punto de referencia donde se alcanza un metro de distancia 41](#_Toc208074033)

**ÍNDICE DE TABLAS**

[Tabla 5.1 Coste de elementos electrónicos 44](#_Toc208074034)

[Tabla 5.2 Coste de licencias 44](#_Toc208074035)

[Tabla 5.3 Coste de personal 44](#_Toc208074036)

[Tabla 5.4 Coste total 45](#_Toc208074037)

# 1. INTRODUCCIÓN

## Motivación del Trabajo.

Desde hace aproximadamente dos décadas los vehículos han ido introduciendo el uso de pantallas y otros elementos digitales para proporcionar más y mejor información al conductor y sus ocupantes. Este proceso de digitalización del vehículo se ha visto acelerado por la introducción de tecnologías que permiten la comunicación ente nuestros dispositivos móviles y las consolas de los vehículos, hasta el punto de sustituir elementos analógicos como el cuentarrevoluciones o el velocímetro.

Aunque los elementos del salpicadero hayan sido digitalizados, los sensores analógicos siguen formando parte de todos los vehículos y ahora más que nunca es importante conocer cómo funciona el proceso de recolección y tratamiento de estos datos, esencial para el desarrollo de tecnologías de conducción autónoma y seguridad vial.

Para que el vehículo sea seguro y el conductor tenga toda la información relevante, esta debe ser mostrada de una forma clara y actualizada. Y para esto tanto el sistema encargado de mostrar dicha información y el encargado de recogerla deben trabajar de forma sincronizada y robusta.

Este proyecto tiene como objetivo recoger información de sensores a través de un microcontrolador programable y presentarlo en un salpicadero digital en otro dispositivo con una conexión alámbrica aplicando conocimientos adquiridos en el grado de Ingeniería Informática

## Objetivos

El proyecto se puede dividir en tres objetivos

### Microcontrolador

Este objetivo tiene como fin la configuración adecuada del microcontrolador, así como del desarrollo del código necesario para la obtención de la información requerida para el salpicadero virtual.

### Conexión Alámbrica

En este objetivo buscamos desarrollar la capacidad de comunicar de forma alámbrica los dispositivos de los que disponemos, es decir, el microcontrolador y el PC donde se dibujara el salpicadero virtual.

### Salpicadero Virtual

Por último, este objetivo desarrollara una interfaz simple que muestre la información obtenida por el microcontrolador de forma clara y rápida.

## Marco regulador

Para el desarrollo del proyecto es importante entender y tener en cuenta el marco regulador que nos afecta en el diseño y funcionamiento del sistema.

El Reglamento 2019/2144 de la Unión Europea establece los requisitos de homologación para los distintos sistemas de los vehículos. Centrándose en los sistemas avanzados de asistencia a la conducción y la protección de las personas.

También debemos considerar la ISO 26262 que marca un estándar para la seguridad funcional de vehículos y los sistemas Eléctricos y Electrónicos.

Dado que la mayor parte del proyecto se centra en el desarrollo de código debemos tener en cuenta estándares establecidos en la industria. El estándar MISRA C establece una serie de mejores prácticas para el desarrollo de código en el lenguaje C dirigido a la industria de la automoción. Por otro lado, se debe tener en cuenta el estándar PEP 8 dirigido al desarrollo de código en el lenguaje Python. El cual trata de establecer una serie de buenas prácticas para la creación de código más mantenible y seguro.

## Estructura del documento

En este apartado se expone la estructura que sigue el documento, detallando la información pertinente sobre el contenido que forma cada capítulo de la memoria.

* **Introducción**: Explicar la motivación del proyecto, también contiene los objetivos del trabajo a realizar, el marco regulador que afecta al trabajo y este apartado.
* **Estado del arte**: Describir la situación actual del ecosistema en el cual se va a situar el proyecto que se va a realizar, con un énfasis en los componentes relacionados con el trabajo.
* **Desarrollo del sistema**: Presentar las distintas partes que conformaran el sistema a desarrollar, centrado en el aspecto técnico de cada componente del sistema.
* **Implementación**: Mostrar el desarrollo del sistema planteado, de sus partes, su funcionamiento y la lógica tras ello. Además de presentar evidencias del funcionamiento de las distintas partes del sistema.
* **Análisis del proyecto**: Examinar el contexto social y económico en que se realiza el proyecto, así como el coste total de este.
* **Conclusiones**: Detallar conclusiones extraídas del desarrollo de los capítulos anteriores, y discutir puntos a mejorar y trabajos futuros realizables para el proyecto
* **Referencias bibliográficas**: Enumerar las fuentes documentales y otros medios de los que se ha extraído algún valor para la realización del proyecto.

# ESTADO DEL ARTE

En 1968 Volkswagen introduce un sistema de inyección electrónica en su modelo *Tipo III E*, que usaba unos 25 transistores para realizar un control preciso de la inyección de la gasolina al motor [1]. No es hasta los años 70 que se empieza a usar microcontroladores de forma más general, para cumplir con el aumento de regulaciones a las emisiones de los vehículos y la eficiencia de los motores.

Actualmente los vehículos constan de varios ordenadores distintos que controlan características diferentes del vehículo. Estos ordenadores están interconectados y controlan desde las medidas de seguridad como el ABS hasta los sistemas de conducción autónoma.

La hibridación y electrificación de los vehículos modernos que intentan buscar un futuro sin emisiones han creado nuevos retos. Como la gestión de un motor térmico junto con uno eléctrico en el caso de los híbridos o la administración de las baterías en el caso de ambas tecnologías.

## Sensores

Actualmente los vehículos usan sensores estandarizados que siguen los mismos principios para obtener las mediciones. Esta afirmación es cierta para los sensores más comunes como la medición de velocidad o el nivel de combustible del vehículo. Seguidamente se muestran ejemplos de estos sensores.

### Revoluciones por minuto

Actualmente esta medida se obtiene a través de un sensor de efecto Hall. Este tipo de sensores funciona creando un voltaje cada vez que es afectado por un campo magnético. Un ejemplo de este sensor es el *Bosch Inductive Crankshaft Position Sensor* o sensor inductivo de posición del cigüeñal, el cual usa un sensor de efecto Hall para detectar el campo generado por los dientes en el volante del cigüeñal. En este caso el sensor no se usa únicamente para medir revoluciones por minuto, también asiste con tareas de sincronización del motor [2].

Forma, Círculo

El contenido generado por IA puede ser incorrecto.

Figura 2.1 Bosch Inductive Crankshaft Position Sensor [3]

### Velocidad y distancia recorrida

En los vehículos modernos la velocidad se obtiene del módulo del ABS, que a su vez lo obtiene de un sensor de efecto Hall que está colocado en las ruedas del vehículo. Se aprovecha la existencia de estos sensores para el sistema ABS y se usa la medición para representar la velocidad en el velocímetro. Al igual que la velocidad se usa este sensor para contar las rotaciones del neumático, junto con las medidas de estos se puede calcular la distancia recorrida [4].

Imagen que contiene aire

El contenido generado por IA puede ser incorrecto.

Figura 2.2 Bosch Wheel Speed Sensor [5]

### Temperatura

Para medir la temperatura del motor o del exterior se usa el mismo concepto, los termistores. Los termistores básicamente son resistencias que varían su valor de resistencia eléctrica en función de la temperatura a la que se encuentren. Para tomar medidas sobre la temperatura del motor el sensor suele estar incluido en un componente comúnmente llamado termostato, el cual se encarga de regular la temperatura del motor para que no se sobrecaliente, pero para que tampoco este demasiado frio [6].

Mientras que el sensor que toma las medidas de temperatura del exterior puede estar situado en distintas partes del vehículo, normalmente en zonas donde no se ve afectado por componentes emisores de temperatura, como el motor o los neumáticos.

Imagen que contiene cuchillo

El contenido generado por IA puede ser incorrecto.

Figura 2.3 Sensor de Temperatura NTC M12 Bosch [7]

### Nivel de combustible

El nivel de combustible se suele obtener a través de un valor de resistencia variable, es decir, un potenciómetro. El potenciómetro es accionado por un mecanismo que incluye una varilla y un flotador, según la altura a la que flote este último el potenciómetro dará un valor de resistencia concreto. Estos componentes suelen ser solidarios a la bomba de combustible, haciendo más sencillo el proceso de extracción de los elementos del tanque de combustible [8].

Imagen que contiene hombre

El contenido generado por IA puede ser incorrecto.

Figura 2.4 Sensor de nivel de combustible Ford E150 [9]

## Salpicaderos

En la actualidad existen una gran variedad de marcas de vehículos, con una gran gama de modelos distintos. Todos con prestaciones diferentes y orientados a distintos usuarios. A continuación, se presentan una serie de ejemplos de vehículos con distintos sensores y la forma en la que se muestra la información recogida al usuario del vehículo.

### Mazda3

El Mazda3 es un hatchback compacto que destaca por su diseño elegante y una conducción dinámica gracias a la tecnología Skyactiv, así como por sus avanzados sistemas de seguridad activa y asistencias a la conducción [10].

Este modelo cuenta con un tablero de instrumentos que integra elementos analógicos y una pantalla.

Diagrama

El contenido generado por IA puede ser incorrecto.

Figura 2.5 Tablero de instrumentos Mazda3 [11]

El Mazda3 presenta las revoluciones en un amplio arco divido en octavos, donde la escala esta reducida para mostrar una sola cifra, empezando en 0 revoluciones y terminando en 8000 representada por un 8.

Imagen de la pantalla de un celular con letras y números

El contenido generado por IA puede ser incorrecto.

Figura 2.6 Tacómetro Mazda3 [11]

La esfera que queda en el medio del tablero se presenta distinta información, de la cual hay 4 medidas que no cambian, localizadas en el exterior de la esfera. Mientras se deja el centro de la esfera para información que el conductor puede cambiar con un mando físico del vehículo. La información presente en este tablero y que siempre está disponible es el cuentakilómetros total, la temperatura exterior, el consumo de combustible actual y la autonomía del vehículo. Además, la velocidad siempre está presente en el cuadro de una forma u otra. Esta información es clave para el conductor ya que presenta datos importantes a tener en cuenta en un viaje, que es el objetivo principal de un vehículo.

Una captura de pantalla de un celular con texto e imágenes

El contenido generado por IA puede ser incorrecto.

Figura 2.7 Visualizador de datos múltiples Mazda3 [11]

La temperatura se muestra en la esfera que se encuentra en el extremo derecho del tablero, compartiendo el espacio con el indicador de combustible del vehículo. Ambos tienen una escala divida en tercios, con una escala menor que divide esos tercios en la misma cantidad.

Forma

El contenido generado por IA puede ser incorrecto.

Figura 2.8 Medidor de temperatura del refrigerante del motor Mazda3 [11]

Imagen que contiene objeto, reloj

El contenido generado por IA puede ser incorrecto.

Figura 2.9 Medidor de combustible Mazda3 [11]

Por último, el Mazda3 cuenta con sensores ultrasónicos delanteros y traseros para la asistencia en carretera y aparcamiento. A la hora de estacionar el vehículo ofrece ayudas sonoras y visuales. Las ayudas visuales muestran la distancia a un obstáculo, en este caso cuanto más cerca está el obstáculo, menor es la distancia dibujada en el indicador que está formado por 4 arcos que están divididos en 3, según el ángulo de detección del obstáculo [12].

Imagen que contiene taza

El contenido generado por IA puede ser incorrecto.

Figura 2.10 Visualización de distancia para el estacionamiento [12]

### Tesla Model 3

Tesla es una de las principales marcas responsables de la revolución eléctrica en el mundo de la automoción. Sus vehículos destacan por un diseño moderno y minimalista, siendo de los primeros fabricantes en transformar el panel de instrumentos de los vehículos en pantallas, con el Tesla Model 3 teniendo dos pantallas como método de transmitir información al conductor del vehículo y el resto de los ocupantes. La pantalla que actúa como panel de instrumentos está colocada de manera horizontal dando espacio suficiente para incluir la información relevante.

La interfaz de este panel de instrumentos sigue la misma línea de diseño que tanto identifica a la marca, el minimalismo. La sección central se enfoca en las ayudas a la conducción por las que destaca el vehículo, el cual cuenta con sensores para la detección de peatones y otros vehículos. En la franja izquierda se muestra el navegador, que además de mostrar la ruta establecida, contiene en la zona inferior la autonomía actual del vehículo junto con la temperatura exterior. La franja derecha indica con un velocímetro digital la velocidad actual del vehículo, junto con la velocidad de crucero seleccionada y un indicador que muestra si el piloto automático del vehículo esta activado.

Imagen de la pantalla de un celular con la imagen de un video juego

El contenido generado por IA puede ser incorrecto.

Figura 2.11 Panel de instrumentos Tesla Model 3 [13]

El Tesla Model 3 cuenta con otra pantalla que principalmente muestra información de navegación y permite acceder a las funciones de entretenimiento del vehículo. Además, desde esta pantalla el conductor puede cambiar el modo de conducción [14].

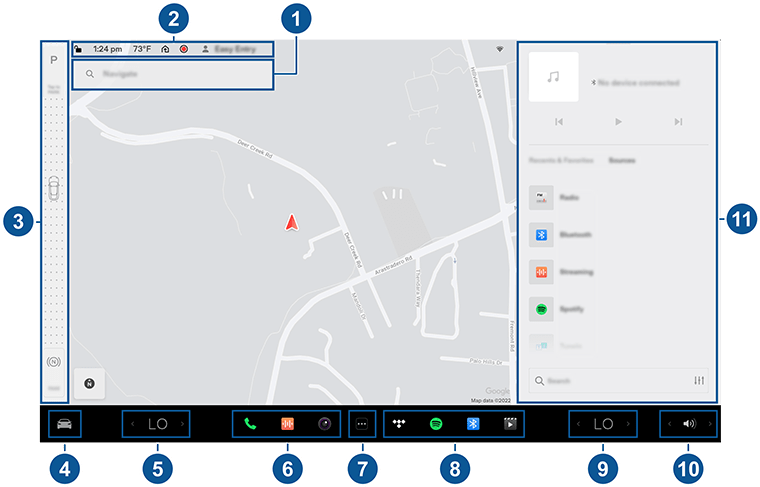


Figura 2.12 Pantalla táctil Tesla Model 3 [14]

### Mercedes Clase C

Mercedes en una de las marcas más conocidas de Europa y tiene una gran historia como fabricante de vehículos de carretera. El fabricante alemán se ha unido a la moda de sustituir el cuadro analógico de sus vehículos por una pantalla que contiene la información que antes era mostrada de manera analógica.

Un reloj en la parte de atrás de un carro

El contenido generado por IA puede ser incorrecto.

Figura 2.13 Visualizador del conductor Mercedes Clase C [15]

El Mercedes Clase C usa una distribución clásica del fabricante, en la que hay dos esferas principales en el cuadro. El velocímetro se encuentra en la izquierda, a diferencia de otros fabricantes. Además, la esfera del velocímetro cuenta con el indicador de nivel de combustible en la zona inferior de esta esfera, junto con el indicador de la autonomía estimada del vehículo.

En el lado opuesto del tablero hay otra esfera que contiene el cuentarrevoluciones del motor. Al igual que el velocímetro la zona inferior de la esfera se aprovecha para incluir la temperatura del motor. Y en la zona inferior izquierda de la esfera tenemos el indicador de la temperatura exterior.

Por último, en la franja central se deja una zona abierta que permite mostrar distinta información, por defecto se muestra el kilometraje del vehículo. Y en la zona superior de esta franja se muestran distintos iconos con información de las ayudas a la conducción junto con un velocímetro digital.

# DESARROLLO DEL SISTEMA

Una vez establecido el contexto del proyecto, se pueden empezar a desarrollar las distintas partes que conforman el sistema.

En este apartado se identifican los requisitos establecidos para el desarrollo del proyecto, se indican los componentes necesarios y se expone el proceso seguido para el desarrollo.

## Requisitos

Para cumplir con el objetivo del proyecto se debe de presentar la siguiente información al usuario, y dicha información debe de provenir del microcontrolador de forma alámbrica.

* **Velocidad**: El microcontrolador debe de ser capaz de entregar la medida de la velocidad en m/s para ser presentada en el salpicadero digital. El salpicadero debe de ser capaz de leer la medida entregada por el microcontrolador y actualizar el valor de esta en la representación adecuada con precisión.
* **Distancia Recorrida**: El microcontrolador debe de ser capaz de escribir la distancia recorrida acumulada en metros. El salpicadero debe ser capaz de registrar el valor de la medida y actualizar la figura que la representa.
* **Revoluciones por minuto**: El microcontrolador debe de ser capaz de enviar la medición en rpm. El salpicadero debe de ser capaz de interpretar la medición y representarla con la magnitud adecuada.
* **Temperatura exterior**: El microcontrolador debe de ser capaz de entregar la lectura de la temperatura exterior en Celsius. El salpicadero debe de ser capaz de leer la lectura de la temperatura y actualizar el valor de la representación.
* **Temperatura interior**: El microcontrolador debe de ser capaz de enviar la medición de la temperatura interior en grados Celsius. El salpicadero debe de ser capaz de leer la medición e interpretarla para mostrarla de forma correcta en una representación diferenciada a la de la temperatura exterior.
* **Señal de seguridad**: El microcontrolador debe de ser capaz de mandar una señal relacionada con la seguridad del vehículo. El microcontrolador debe de ser capaz de representar esta señal de manera adecuada para la seguridad de los ocupantes del vehículo.
* **Nivel de llenado del depósito**: El microcontrolador debe de ser capaz de mandar la medida de llenado del depósito de combustible. El salpicadero debe de ser capaz de leer y representar dicha medida.

## Componentes

En este apartado se exponen los distintos sensores elegidos para el proyecto, incluyendo en caso de ser necesario información sobre opciones descartadas con la debida justificación.

### Microcontrolador

Se estableció que se debía usar un microcontrolador programable para el desarrollo de la parte relacionada con la obtención de las mediciones de sensores. Esto descarto placas como la raspberry pi, ya que, aunque era posible usarse como placa para el desarrollo del proyecto se decidió que era demasiado potente para este proyecto.

Durante la fase inicial de proyecto se intentó usar una placa STM32F407VE la cual fue seleccionada por su gran número de pines GPIO, buenas características de rendimiento y su precio competitivo. Cuando se empezó con el desarrollo, se encontraron distintas dificultades para trabajar sobre ella. Principalmente el hecho de que no tiene integrado el hardware necesario para la depuración ni reprogramación del microcontrolador. También hubo problemas para encontrar documentación y guías que usaran la placa o alguna similar. Con la única documentación disponible siendo lo diagramas ofrecidos por el fabricante.

Un circuito electrónico

El contenido generado por IA puede ser incorrecto.

Figura 3.1 Vista de perspectiva de la placa STM32F407VE [16]

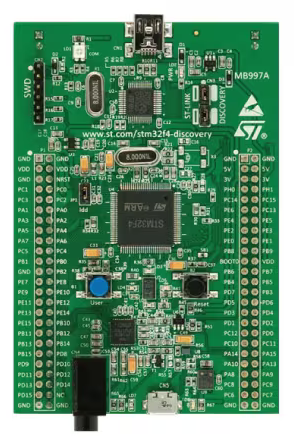
Por eso se deicidio usar la placa de desarrollo STM32F407G Discovery. Las principales razones para elegir esta placa son: la inclusión de una herramienta de reprogramación y depuración llamada ST-LINK, la gran cantidad de pines GPIO, y la posibilidad de configurar una conexión serie, además de la popularidad de esta placa en el entorno de la programación de microcontroladores [17].

Figura 3.2 Microcontroladora STM32F407G-DISC1 [17]

### Sensor de ultrasonidos

Se usarán los sensores de ultrasonidos HC-SR04 para simular los sensores de proximidad de los vehículos, proporcionando de esta manera la señal de seguridad. Se usarán 2 unidades de este sensor para recoger la distancia que haya ha obstáculos como otros vehículos, personar u objetos. Estos componentes representaran los distintos grupos de sensores que se suelen encontrar en la parte delantera y trasera de los vehículos.



Figura 3.3 Sensor de ultrasonidos HC-SR04 [18]

Estos sensores tienen un rango efectivo que va de los 2cm a los 400cm, con una precisión de 3mm. Esto los convierte en una opción razonable y asequible para el desarrollo del proyecto.

El sensor consta de 4 pines, Vcc donde debe recibir una alimentación de 5V, un pin Gnd para conectarlo a tierra, un pin Trig o Trigger para accionar el sensor y un pin Echo donde se recibirá la respuesta.

Para explicar su funcionamiento debemos ver el diagrama de tiempos proporcionado por el fabricante.

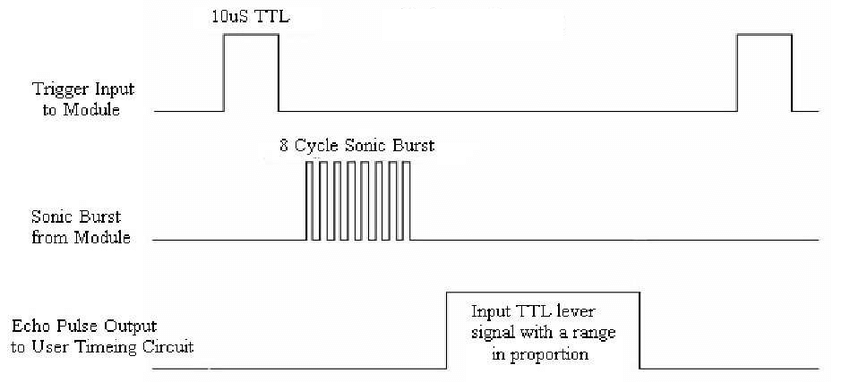


Figura 3.4 Diagrama de tiempos sensor HC-SR04 [19]

Para iniciar la medición por parte del sensor deberemos de mandar una señal de 10μs al pin Trig, tras esto el sensor lanza 8 pulsos de ultrasonidos a una frecuencia de 40KHz. Por último, el sensor devuelve una señal en el pin Echo que indica el tiempo que ha tardado en recibir el pulso de vuelta. Entre cada señal se mande al sensor deben de transcurrir al menos 60 milisegundos según el fabricante [19].

Con esta información podemos calcular la distancia en cm que han recorrido los pulsos de ultrasonidos emitidos por los sensores. Para ello usaremos la siguiente formula.

Siendo *tiempo* el valor en microsegundos que ha durado la señal recibida en el pin echo y *1/58* el factor de conversión a cm proporcionado por el fabricante [19].

### Sensor de temperatura

Para el sensor de temperatura inicialmente se usó el modelo DHT11 muy usado en proyectos de este tipo. Pero tras implementar la funcionalidad de este sensor se descartó por su alta latencia, ya que generaba un bloqueo en la ejecución de más de un segundo [20].

Finalmente se decidió usar un termistor, es decir, una resistencia variable, cuyo valor de resistencia varía en función de la temperatura de esta. El termistor usado se trata de un PT100 de 3 pines que utiliza platino como material sensible y cuya resistencia eléctrica a 0 °C es de 100 Ω [21]. Su principio de funcionamiento se basa en que la resistencia del platino aumenta de manera predecible y casi lineal con la temperatura, lo que permite relacionar la medida de resistencia con la temperatura real. Al tener 3 pines mitiga parte del error introducido por la resistencia añadida del cable que va de la punta del sensor donde se encuentra la resistencia hasta los terminales donde se conecta el sensor.

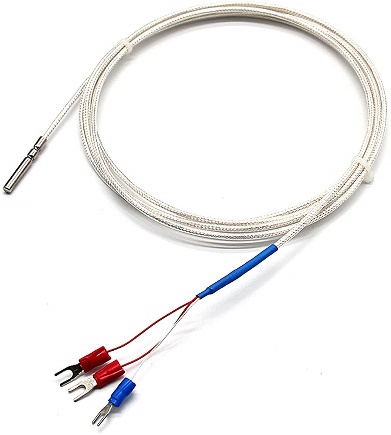


Figura 3.5 Sensor de temperatura PT100 [22]

Para poder usar este tipo de sensor se optó por usar un conversor de valor de resistencia a digital, conocido como Conversor RTD.

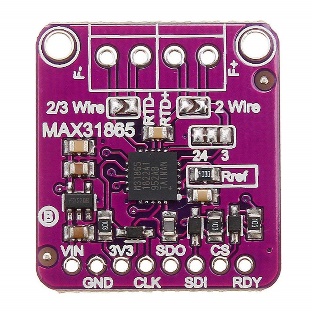


Figura 3.6 Conversor de señal RTD a digital [23]

El Conversor elegido fue el MAX31865 cuyo funcionamiento se basa en medir la variación de resistencia que presenta el sensor con la temperatura, y convertirla en un valor digital que pueda ser interpretado por el microcontrolador a través de la interfaz SPI. Para ello, el chip aplica una corriente de activación al RTD y compara la caída de voltaje resultante con una resistencia de referencia de alta precisión. De esta manera obtiene una medida relativa que luego pasa por un convertidor analógico-digital sigma-delta, lo que permite obtener una lectura muy precisa [24].

### Sensor fotoeléctrico

Para poder obtener la medida de revoluciones por minuto es necesario un elemento que tenga rotación. En los vehículos de combustión se suele usar el volante del cigüeñal y sus dientes para obtener esta medida [2]. Dado que no disponemos de un motor térmico para capturar esta información, lo mejor que se puede hacer es simular el comportamiento del volante del cigüeñal.

Este sensor se usará para sustituir el sensor de efecto Hall y así medir las revoluciones por minuto a las que gira el motor. Se trata de un sensor DAOKAI LM393, este usa un haz de luz infrarroja emitido contra un receptor para detectar las interrupciones del haz [25].

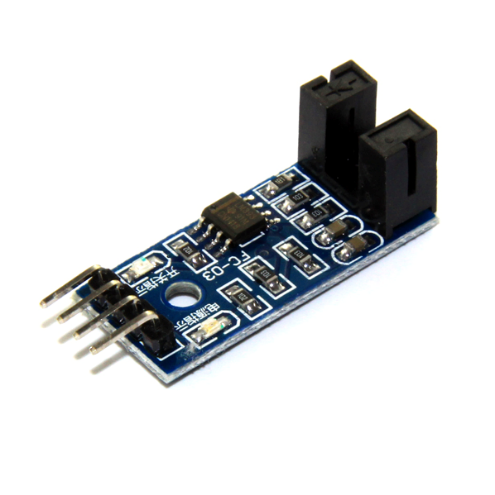


Figura 3.7 Sensor fotoeléctrico Daokai LM393 [26]

Para simular el efecto de los dientes del volante del cigüeñal añadimos una rueda perforada con 20 huecos para interrumpir el haz del sensor, sabremos cuando ocurre una revolución.



Figura 3.8 Rueda perforada con 20 ranuras [27]

La rueda se conectará a un motor de corriente continua, que no se incluye como parte del proyecto ya que no será controlado por ninguno de los sistemas a desarrollar y simplemente se usará para comprobar el correcto funcionamiento de este sensor concreto.

### Sensor de aceleración

Para obtener información como velocidad y distancia recorrida se va a usar el sensor de aceleración de 3 ejes del que dispone la placa. El acelerómetro de 3 ejes LIS3DSHTR, el cual proporciona la aceleración detectada en 3 ejes (x, y, z) [28]. Se ha decido usar este sensor ya que viene incluido en la propia placa, además esto posibilita la toma de medidas independientemente del vehículo en el que se pueda montar esta placa con esta implementación.

Diagrama, Dibujo de ingeniería

El contenido generado por IA puede ser incorrecto.

Figura 3.9 Vista de los ejes de detección de aceleración [28]

Para poder hacer uso de la aceleración primero debemos de tener en cuenta la aceleración ejercida por la gravedad terrestre. Además, hay que tener en cuenta el factor del ruido en las mediciones del acelerómetro, el cual habrá que mitigar.

### Potenciómetro

Para simular el sensor que mide la cantidad de combustible que hay en un tanque vamos a usar un potenciómetro. Ya que si reducimos los componentes de este tipo de sensor llegamos a que básicamente se trata de un potenciómetro, es decir, una resistencia variable conectada a un flotador [8].

En este caso se usa un potenciómetro de 200Ω del cual ya se disponía. Esto implica que las lecturas irán de los 0Ω a los 200Ω, pero se espera que el valor recibido por el microcontrolador pueda ser algo mayor por la resistencia añadida del cable y ruido proveniente de otros componentes cercanos, además de posibles defectos del propio potenciómetro.

Imagen que contiene tabla

El contenido generado por IA puede ser incorrecto.

Figura 3.10 potenciómetro rotatorio [29]

## Diseño del salpicadero

Antes de comenzar el proceso de desarrollo se debe plantear el diseño que se quiere alcanzar durante la creación del salpicadero. Para realizar este diseño se usa la plataforma *Figma* [30] para crear el prototipo del tablero.

Se decide usar elementos a los que el usuario ya está acostumbrado. Esto quiere decir que se usaran diales para representar las revoluciones por minuto, la velocidad del vehículo, la temperatura del motor y el nivel de llenado del vehículo. Mientras que para la distancia recorrida y la temperatura exterior se usan cuadros de texto con las unidades correspondientes. Por último, para representar la distancia de seguridad se usará una imagen del vehículo, en esa imagen se representará la distancia que hay en la parte delantera y trasera del vehículo con arcos de distinto color. El color ayuda a que el conductor no tenga que fijar su vista en el indicador para identificar si el aviso es para la zona delantera o trasera del vehículo.

Diagrama

El contenido generado por IA puede ser incorrecto.

Figura 3.11 Prototipo del diseño del salpicadero

El diseño se puede dividir en tres áreas, la primera zona contiene los diales para la velocidad y las revoluciones por minuto del vehículo. Se sitúa en la zona superior para que el conductor no tenga que apartar la vista de la carretera. Los diales contendrán la escala adecuada para representar la velocidad del vehículo y las revoluciones por minuto del motor.

La segunda zona contiene el indicador de la distancia a obstáculos. Para indicar que un obstáculo está a una distancia muy cercana al vehículo se mostraran el número máximo de arcos posibles, generando de esta forma una alerta visual llamativa. A partir de la distancia mínima los arcos irán desapareciendo a medida que la distancia con un obstáculo aumente, los arcos irán desapareciendo a intervalos de distancia continua de más lejanos del vehículo hasta que al alcanzar la distancia máxima desaparezca el arco más cercano al vehículo. Este comportamiento es el mismo para el indicador frontal y trasero.

Por último, está la zona que se encuentra debajo de los diales de la velocidad y las revoluciones por minuto. En este cuadro de información se encuentra información relevante sobre el vehículo pero que no afecta de forma directa e inmediata a la conducción de este. Los diales de nivel de combustible y temperatura están divididos en cuartos. En el caso del nivel de combustible se usa la *E* para indicar que el tanque está vacío y *F* indica que está lleno. Mientras que en el dial de temperatura se usa *C* para indicar que la temperatura del motor es fría y *H* para marcar que está caliente. La distancia recorrida se indica con un máximo de 5 dígitos siguiendo el estilo de un odómetro analógico, y la temperatura exterior se indica en grados Celsius.

# IMPLEMENTACIÓN

En este apartado se expone el funcionamiento y el proceso de implementación seguido para cumplir los requisitos y alcanzar los objetivos del proyecto. Esto incluye la explicación de cómo se configuran elementos del microcontrolador, así como implementación de código para ambas partes del proyecto.

## Microcontrolador

Para el desarrollo del código se ha empleado el entorno de desarrollo integrado STM32CubeIDE [31]. El cual incluye las herramientas necesarias para compilar el código C y permite configurar más fácilmente el microcontrolador. Esto incluye configurar la velocidad del reloj del microcontrolador, los pines donde se conectan los distintos sensores, así como generar el código básico para las configuraciones realizadas.

El entorno de desarrollo genera un archivo de código fuente *main.c* donde se encuentra la función *main()* que ejecuta la funcionalidad principal y es donde principalmente se encuentra el código desarrollado para la implementación. En este archivo se generan automáticamente zonas delimitadas donde se incluye el código del usuario y que se mantiene entre los cambios de configuración y la consecuente regeneración del código fuente.

### Comunicación alámbrica

La primera parte que se implementó fue la comunicación alámbrica, ya que esta funcionalidad se encarga de hermanar las dos partes del proyecto. Si el microcontrolador no es capaz de comunicarse con el sistema que ejecuta el salpicadero, no tiene sentido invertir tiempo en el resto de las funcionalidades.

Para la comunicación alámbrica se usa la interfaz *USART2* en modo asíncrono. Y se configura con los siguientes parámetros. *Baud Rate* en 115200 Bits/s, un *Word Length* de 8 Bits, sin Bit de paridad y el parámetro de *Stop Bits* a uno. Esta configuración es importante, ya que si los parámetros no son los mismos no se podrá leer la información enviada desde el salpicadero.

La comunicación se lleva a cabo usando un conversor de USB a TTL UART. Este conversor permite convertir la señal enviada por el microcontrolador a través de los pines PA2 y PA3, encargados de transmitir y recibir la información respectivamente. Estos pines se configuran automáticamente al activar la configuración de la interfaz serie de la placa según indica el manual con las especificaciones.

Un circuito electrónico

El contenido generado por IA puede ser incorrecto.

Figura 4.1 Conversor USB a TTL UART [32]

Se van a transmitir un total de 8 variables, de las cuales 6 son de tipo *float* que ocupan 4 bytes y las otras 2 son de tipo *uint32\_t* que ocupa un tamaño de 8 bytes. En total transmitiremos un total de 40 bytes. Los datos se transmitirán como un *string* de caracteres separados por comas, para ello guardaremos los valores en un *buffer* de 41 bytes. Ese byte extra se añade para cubrir el espacio que ocupen las comas y otros caracteres especiales. Las mediciones siguen el siguiente orden: distancia frontal, distancia trasera, temperatura exterior, temperatura interior, revoluciones por minuto, velocidad, nivel de combustible, distancia recorrida.



Figura 4.2 Declaración del buffer para el envío de datos

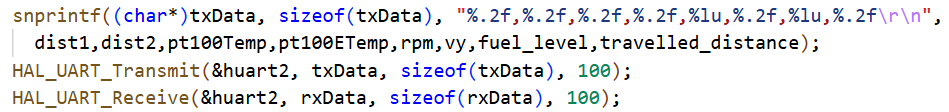


Figura 4.3 Código encargado de la comunicación serie

### Sensores de proximidad

Como se expone en el apartado 3.2.2 donde se describe el funcionamiento del sensor HC-SR04, hay que esperar un mínimo de 60 ms entre cada medición del sensor. Para tener un margen de sobra la lógica que ejecuta el ciclo de lectura del sensor se ejecuta solo si han pasado más de 100 ms. Para esto se hace uso de la función *HAL\_GetTick()*, que forma parte de la capa de abstracción de Hardware proporcionada como librería por parte del entorno de desarrollo. La función devuelve el número de milisegundos que han pasado desde el inicio del sistema. Cada vez que se ejecuta el código se guarda el valor actual que devuelve la función mencionada, de este modo en cada iteración del bucle principal se puede comparar el valor en ese instante con el de la última ejecución de la mencionada sección de código.

*if (HAL\_GetTick() - delay\_hcsr04 >= 100)*

A ambos sensores se les manda la misma señal de Trigger, de esta forma ambos toman la medición de una forma coordinada. Esta señal se manda a través del Timer TIM11 de la placa. Este Timer se ha configurado para que un tick dure 2.5μs y el pulso se mantenga a lo largo de 4 ticks, lo que equivale a un pulso de 10μs. El Timer realiza este proceso cada 100ms.

Estos son los cálculos utilizados para configurar el Timer correctamente:

Donde *APB2* es la frecuencia del reloj al que está conectado el Timer TIM11 y *Ttick* es el tiempo que dura un tick. *fTIM11* es la frecuencia con la que queremos que se complete un ciclo, en este caso es 10Hz, que son 100ms. Tras fijar esos valores se calcula el *Prescaler* y el *CounterPeriod* que representan los valores de configuración del Timer que tiene el mismo nombre. De esta forma quedaría configurado el comportamiento del Trigger [33].

Para la medición de la duración de la señal que indica la distancia medida por los sensores necesitaremos usar otro Timer. En este caso el TIM8 el cual se configura con *Reset Mode* en el apartado *Slave Mode,* esto quiere decir que se configura para reiniciarse cada vez que detecta una señal, en este caso la señal mandada por el TIM11. De esta forma siempre estará listo para recibir la señal generada por el sensor. Al tener dos sensores también se configuran 2 canales de los cuatro disponibles como *Input Capture* *direct*, lo que significa que usaremos los pines predefinidos como puntos de lectura. Para el TIM8 estos pines son PC7 y PC6 [33] [34] [35].

Finalmente llamaremos a la función *HAL\_TIM\_ReadCapturedValue()*, donde se especifican como parámetros la referencia del Timer que se va a leer y el canal de dicho Timer. Tras dividir el valor leído entre 58 se obtiene la distancia en centímetros que hay delante del sensor.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.4 Código encargado de la funcionalidad del sensor HC-SR04

### Sensores de temperatura

Inicialmente se usaba un sensor DHT11 para la obtención del valor de la temperatura. Una vez implementada la funcionalidad correspondiente se comprobó que el proceso de obtención de medidas bloqueaba la ejecución durante un segundo aproximadamente.

Se buscaron soluciones para poder continuar usando el sensor sin eliminar toda la implementación de este. Se barajo el uso de RTOS para tener una forma de ejecutar el código de forma que no se bloqueara la ejecución. Ya que cuando se utiliza un RTOS, el sistema operativo toma el control de la planificación y ejecución de tareas. De tal modo que cada tarea se define como un hilo separado, y el RTOS se encarga de asignar tiempo de CPU a cada una según prioridades y reglas de planificación. Permitiendo programar la funcionalidad de forma concurrente.

Se considero que el tiempo necesario para aprender e implementar el uso de un RTOS era demasiado elevado para este proyecto y se cambió el sensor a un termistor pt100 cuya implementación tiene una ejecución más corta y ofrece resultados con más precisión.

Los sensores de temperatura están conectados a un conversor RTD como ya se explicó previamente en el punto 3.2.3. La comunicación entre estos conversores y la placa se realiza a través de la interfaz de comunicación *SPI*.

La comunicación SPI funciona mediante el intercambio de datos en serie entre un dispositivo maestro, que será el microcontrolador, y uno o varios dispositivos esclavos. El maestro controla la comunicación generando una señal de reloj que sincroniza la transmisión y recepción de datos. Cada bit de información se transfiere simultáneamente desde el maestro al esclavo y viceversa, usando líneas separadas para enviar y recibir datos [36].

La temperatura del interior del vehículo es recibida a través de la interfaz *SPI2*, mientras que la temperatura del motor es obtenida de la interfaz *SPI3*. Para configurar estas interfaces hay que poner el modo *Full-Duplex Master* y configurar la opción *Clock Polarity* como *Low* junto con *Clockl Phase* como *2 Edge*. Esto significa que leeremos los datos durante el flanco de bajada de la señal [35].

Se usa la librería *max31865* [37] que nos permite inicializar fácilmente la interfaz SPI y realiza la conversión del valor de resistencia a Celsius directamente. Desde el bucle principal se llama a la función *MAX31865\_readTempC()* que da la medición actual, se comprueba que el valor no es incorrecto y se pasa por un filtro para compensar posibles fallos de lectura y así suavizar los valores de temperatura.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.5 Código encargado de la lectura del sensor pt100

### Sensor de aceleración

El acelerómetro incluido en la placa está conectado al *SPI1*, para su lectura solo es necesario configurarlo en modo *Full-Duplex Master*. Para interactuar con el acelerómetro y leer los valores se usa la librería *LIS3DSH* [38] que permite la interacción con el sensor de aceleración.

Para obtener la aceleración actual se llama a la función *LIS3DSH\_GetDataScaled()* que nos devuelve los valores de la aceleración en cada eje. Como se ve en la Figura 4.8 y por la orientación del sensor en la placa, el movimiento del vehículo se ejerce en el eje *y*.

La lectura de la aceleración se devuelve como la milésima parte de *g*. Para la conversión a *m/s2* solo es necesario multiplicar por 9.81 y dividir entre 1000. En este caso se multiplica por 0.00981 simplificando el código.

Para obtener una lectura sin interferencias de la gravedad terrestre la alineación de la placa debería de ser perpendicular a esta. Es prácticamente imposible cumplir con esto, para mitigar esto se ignoran las aceleraciones que estén por debajo de los 0.5m/s2.

Además, para mitigar el posible ruido del sensor y suavizar las lecturas se aplica un filtro de paso bajo, atenuando los posibles fallos de lectura que suelen presentarse como picos en los valores de aceleración, ya sea con valor positivo o negativo.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.6 Implementación del filtro

Una vez se tiene un valor aceptable de la aceleración, se pasa a el cálculo de la información derivada de la aceleración. Vamos a derivar la velocidad en el eje *y*, gracias a esto podremos obtener la distancia recorrida.

Para el cálculo de la velocidad simplemente se obtiene el diferencial de tiempo entre cada calculo, para multiplicarlo por la aceleración y de esta forma conseguir la velocidad. La velocidad se guarda en una variable que acumula el valor, de esta manera se aplican los cambios de velocidad cuando se acelera o frena.

Una vez se tiene la velocidad se puede calcular la distancia recorrida, solo hay que obtener el diferencial de tiempo de cada iteración del bucle y multiplicarlo por el valor absoluto de la velocidad, ya que la velocidad negativa solo indica la dirección del movimiento.

### Sensor fotoeléctrico

Para el sensor fotoeléctrico solo hace falta configurar un pin como *External Interrupt*, es decir que cuando el sensor detecte un corte en el haz de luz mandara una interrupción al pin que se ha configurado.

En el archivo generado por el propio entorno de desarrollo encargado de las rutinas para las interrupciones, se declara una variable como contador, a la que se va sumando valores cada vez que ocurre una interrupción en el pin configurado.

Finalmente, en el archivo principal declaramos una función que se ejecute cada vez que se genera una interrupción de algún Timer. En este caso el TIM6, que está configurado para generar una interrupción cada segundo [35]. Cuando eso ocurra solo tendremos que multiplicar el numeró de interrupciones detectadas por el sensor por 60 segundos y dividirlo entre las 20 ranuras que tiene una revolución. Esto último se reduce a una multiplicación por 3. Por último, se reinicia a 0 el contador de interrupciones del sensor fotoeléctrico.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.7 Código encargado de calcular el valor de las rpm

### Potenciómetro

El potenciómetro dará el valor para el nivel de combustible, para esto hay que usar un conversor de valor analógico a digital o *ADC.* Esto se puede hacer a través de una de las interfaces *ADC* con las que cuenta la placa. Para el proyecto se usará la interfaz *ADC1*, en el modo IN0, lo que configura el puerto PA0 como pin de entrada de la señal [35].

En el bucle principal se llama a la función *HAL\_ADC\_GetValue()* que simplemente devuelve el valor de la entrada convertido. Como el valor de resistencia simplemente indica la posición que tiene el mecanismo del potenciómetro no es necesario hacer ningún tratamiento adicional de esta medida.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.8 Código encargado de la obtención del valor del potenciómetro

### Diagrama de conexiones

Una vez todos los sensores han sido configurados el diagrama de pines queda de la siguiente manera.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Figura 4.9 Esquema de pines STM32F407

El diagrama de conexiones muestra todos los sensores y periféricos conectados a la placa. Este diagrama se ha hecho con la herramienta *fritzing* [39] que permite hacer diagramas y otros esquemas de elementos electrónicos.

Imagen de la pantalla de un video juego

El contenido generado por IA puede ser incorrecto.

Figura 4.10 Diagrama de conexiones

## Salpicadero digital

El salpicadero digital se desarrolla con *Visual Studio Code* y esta desarrollado en el leguaje Python. Se usa este leguaje por la familiaridad que se posee con su sintaxis y por integrar la mayoría de las herramientas necesarias para el desarrollo del proyecto, como las librerías para crear una interfaz gráfica. Se usa la librería estándar de Python *tkiner* para esta interfaz.

Si bien es cierto que Python es un lenguaje popular, no todas sus librerías tienen la misma base de uso y documentación. Por eso el proceso de aprendizaje sobre esta librería se entiende como parte del desarrollo de este proyecto. Principalmente habrá que entender cómo se permite organizar la interfaz, como funciona el proceso de dibujado de los distintos elementos necesarios para construir la interfaz, y como se pueden actualizar esos elementos.

Tkinter es la biblioteca estándar de Python para la creación de interfaces gráficas. Actúa como un *wrapper*, es decir, una interfaz que permite utilizar las herramientas gráficas Tcl/Tk desde Python, ofreciendo una forma sencilla y multiplataforma de desarrollar aplicaciones con ventanas, botones, menús, cuadros de texto, y otros elementos gráficos comunes. Internamente, Tkinter traduce las llamadas de Python a comandos Tcl/Tk, por lo que hereda muchas de las características y limitaciones de ese sistema [40].

Dado que las herramientas gráficas usadas se desarrollaron al final de la década de los 80, no se espera desarrollar una interfaz impactante o bonita. El objetivo principal de esta interfaz es mostrar claramente toda la información recogida por el microcontrolador.

Las herramientas principales que se usan de esta librería son la estructura *grid* que permite ordenar las partes de nuestra interfaz siguiendo un patrón de filas y columnas. El otro componente será el denominado *Canvas* que es la forma en la que agruparemos los distintos objetos gráficos que forman cada pieza de la interfaz.

### Dial de rpm y velocidad

Para representar las revoluciones por minuto y la velocidad obtenida a través de la comunicación con el microcontrolador, se usa la figura del dial. Ambas magnitudes se representan de esta manera en el mundo real, por eso se usa la misma representación.

Con el fin de generar esta representación se crea una clase llamada *GaugeCanvas* que hereda de la clase *Canvas* definida en la librería que estamos usando. Esta representación está formada por un arco con la escala y una aguja que indica el valor en esa escala.

Como parámetros del constructor se aceptan los siguientes campos:

* **master**: este parámetro indica el elemento padre que contiene al objeto que se cree a partir de esta clase.
* **width**: este parámetro indica el ancho total en pixeles que ocupara el objeto en la interfaz.
* **height**: este parámetro indica la altura total en pixeles que tendrá el objeto en la interfaz.
* **max\_value**: este parámetro marca el valor máximo a representar en el dial.
* **step**: este parámetro sirve para indicar las unidades que habrá entre cada valor representado en el dial.
* **gauge\_text**: este parámetro perime definir una línea de texto que se colocara por encima de la perilla de la que nace la aguja del indicador.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Figura 4.11 Constructor de la clase GaugeCanvas

La clase cuenta con tres métodos, *draw\_gauge()*, *draw\_needle()* y *update\_value()*. Este último es el encargado de actualizar el valor al que indica la aguja sin parar del máximo o caer por debajo de 0 que es el mínimo. A continuación, vamos a ver la lógica que siguen las dos funciones encargadas de dibujar el contenido del dial.

La librería incluye una función *create\_arc()* que permite dibujar una elipse dando como parámetros dos tuplas de puntos que definan el rectángulo que la contiene. También se define que parte del arco de la elipse se quiere dibujar, en este caso se quiere obtener algo similar a un semicírculo, es decir, queremos la parte superior de la elipse que está situada en un arco de 0º a 180º.

La otra parte de este arco es la escala, para dibujarla se colocan líneas usando la función *create\_line()*. Para poder dibujar las marcas es necesario calcular las coordenadas de los puntos que indican el inicio y el fin de cada recta, y para poder hacer ese cálculo es necesario obtener el ángulo al que va colocada cada marca. Esto se hace usando la siguiente formula:

Donde *θmarca* es el ángulo de la recta que representa la marca, *θinicial*indica donde empieza el arco, *i* representa el valor que se va a representar en la escala, *N* representa el valor máximo que puede tener la escala, y *Δθ* indica la extensión en grados del arco.

El resultado se transforma a radianes, ya que la librería *math* de Python espera radianes para realizar cálculos y no grados.

Para calcular la distancia aplicamos las siguiente formulas:

Donde *x* e *y* son las coordenadas del punto, *x0* e *y0* son las coordenadas del centro del arco, y *θ* es el ángulo previamente calculado.

De esta forma obtenemos los puntos sobre el arco que se dibuja, y para obtener los puntos que completan las rectas que representan la escala solo debemos hace el mismo calculo anterior, modificando el radio para crear un arco más pequeño. Por último, para añadir las unidades en la escala se usan las mismas formulas, pero con un radio menor para que estas se puedan leer.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.12 Método draw\_gauge de la clase GaugeCanvas

Si se aplican las mismas fórmulas, primero para calcular el ángulo para un valor de velocidad o rpm, y luego se calcula un punto para ese ángulo con un radio menor que los anteriores, se puede dibujar una recta desde el centro hasta ese punto. Obteniendo de esta manera la aguja del dial.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.13 Método draw\_needle de la clase GaugeCanvas

Imagen que contiene dispositivo, medidor

El contenido generado por IA puede ser incorrecto.

Figura 4.14 Dial revoluciones por minuto del salpicadero digital

Un conjunto de letras blancas en un fondo blanco

El contenido generado por IA puede ser incorrecto.

Figura 4.15 Dial de velocidad del salpicadero digital

### Indicador de distancia de seguridad

Se sigue el mismo procedimiento para dibujar el indicador de la distancia de seguridad, se crea una clase que contendrá los distintos elementos de la representación. La clase llamada *DistanceCanvas* y que contiene los métodos para crear la representación hereda como la clase creada para el dial de la clase *Canvas*.

Como parámetros del constructor se aceptan los siguientes campos:

* **master**: este parámetro indica el elemento padre que contiene al objeto que se crea a partir de esta clase.
* **width**: este parámetro indica el ancho total en pixeles que ocupara el objeto en la representación.
* **height**: este parámetro indica la altura total en pixeles que tendrá el objeto en la interfaz.
* **front\_distance**: este parámetro indica la distancia frontal inicial a representar.
* **back\_distance**: este parámetro indica la distancia trasera inicial a representar.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.16 Constructor de la clase DistanceCanvas

La clase tiene tres métodos *draw\_arc\_distances()*, *draw\_scene()* y *update\_distances()*. Esta última simplemente actualiza los valores de los atributos que almacenan el valor de la distancia y llama a la función *draw\_scene()*. Esta otra función elimina los arcos que estén dibujados y llama a la función *draw\_arc\_distances()* para que represente las distancias actuales. A continuación, se explica el funcionamiento y la lógica de esta función.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.17 Método draw\_scene de la clase DistanceCanvas

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.18 Método update\_distances de la clase DistanceCanvas

En este caso se usa una imagen para representar al vehículo y se dibujan arcos a medida que la distancia disminuye, alertando de la proximidad de objetos. Para conseguir este comportamiento debemos limitar la distancia a partir de la cual se empezarán a dibujar estos arcos, en este caso 50cm, es decir, cualquier medición por encima de 50cm no se indicará, y cualquier medida que este por debajo tendrá una representación. Esto se realiza con la función *min()*, que devuelve el valor más pequeño de los que se le dan como argumento. En este caso a la función se le da el valor de la distancia actual que queremos representar y 50, que es la distancia a partir de la cual se dibuja una representación. Para obtener el número de arcos a dibujar se utilizan las siguiente formulas:

Donde *dacotada*es la distancia obtenida a través de la función *min()*, *dlim* es la distancia limite a partir de cual se empiezan a dibujar arcos, en este caso 50cm. La variable *ratio* indica en un valor entre 0 y 1 la proporción que hay entre la distancia a mostrar y la máxima. Donde *arcos* indica el número de arcos que se dibujaran y se obtiene de la función *round()*, a la que se le pasa como parámetro el producto del *ratio* y el número máximo de arcos representado por *arcosmax*. De esta forma se obtiene el número de arcos que se deben dibujar en la interfaz.

Para dibujar los arcos se aumenta el tamaño del rectángulo que define le elipse por una cantidad fija, de este modo conseguimos un aumento de tamaño y dejamos un espacio visual que marca el intervalo de los arcos.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.19 Método draw\_arc\_distnaces de la clase DistanceCanvas



Figura 4.20 Indicador de distancia de seguridad del salpicadero digital

### Cuadro de información del vehículo

Este elemento del salpicadero virtual contiene la información respectiva a la distancia recorrida, nivel del depósito de combustible, temperatura del motor y temperatura del interior del habitáculo.

Como en los apartados anteriores los elementos que forman parte del cuadro se han agrupado en una clase llamada *RectangleDashboardCanvas*. Aunque en este caso también se ha creado una clase llamada *Gauge* para especificar el comportamiento de los indicadores de combustible y temperatura del motor, ya que se usa un dial para representarlos.

La clase *RectangleDashboardCanvas* acepta como parámetros del constructor estos tres campos:

* **master**: este parámetro indica el elemento padre que contiene al objeto que se crea a partir de esta clase.
* **width**: este parámetro indica el ancho total en pixeles que ocupara el objeto en la representación.
* **height**: este parámetro indica la altura total en pixeles que tendrá el objeto en la interfaz.

****

Figura 4.21 Constructor de la clase RectangleDashboardCanvas

La clase tiene cuatro métodos *update\_odometer()* que se encarga de actualizar el valor que se reciba para la distancia recorrida, *update\_temperature\_text()* se encarga de actualizar el texto que representa la temperatura exterior, *update\_fuel\_level()* llama al método de actualización del objeto que representa el dial del nivel de combustible, *update\_engine\_temp\_level()* llama al método de actualización del objeto que representa el dial de la temperatura del motor.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 4.22 Métodos de la clase RectangleDashboardCanvas

El indicador del nivel de combustible y el indicador de la temperatura exterior son dos elementos de texto creados con la función *create\_text()*. Estos elementos se crean en el constructor de la clase.



Figura 4.23 Indicador de distancia recorrida del Cuadro de información



Figura 4.24 Indicador de temperatura del habitáculo del Cuadro de información

La clase *Gauge* tiene los siguientes parámetros de construcción:

* **master**: este parámetro indica el elemento padre que contiene al objeto que se crea a partir de esta clase.
* **center**: este parámetro es una tupla que indica las coordenadas del centro de la representación del dial.
* **radius**: este parámetro indica el radio que tendrá el dial.
* **label\_left**: este parámetro indica el texto que se dibujara en el lado izquierdo del dial, es decir, es la etiqueta para el valor 0 del dial.
* **label\_right**: este parámetro indica el texto que se dibujara en el lado derecho del dial, es decir, es la etiqueta para el valor 1 del dial.

**Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.**

Figura 4.25 Constructor de la clase Gauge

La clase tiene 4 métodos *draw\_gauge()* quedibuja el arco del dial, *draw\_tick()* que dibuja la escala del dial, *place\_label()* que dibuja las etiquetas para los valores extremos del dial y *update()* que actualiza la posición de la aguja para representar el valor actual. A continuación, se explica la lógica de las dos primeras funciones.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 4.26 Método draw\_gauge de la clase Gauge

Los diales están representados por un arco que se extiende 270º, este acro empieza a partir de los 135º. Una vez creado el arco se dibuja la escala, la cual divide el arco en arcos de 67.5º, es decir, se divide el arco en cuartos. Para el cálculo de las marcas de la escala se usan las fórmulas (4.8), (4.9) y (4.10).

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.27 Método draw\_tick de la clase Gauge

Los valores que se pueden representar en el dial tendrán que estar comprendidos entre 0 y 1. Para el indicador del nivel de combustible 0 indica vacío y 1 lleno. Para el indicador de temperatura del motor 0 indica frio y 1 caliente.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.28 Método update de la clase Gauge

Imagen que contiene Forma

El contenido generado por IA puede ser incorrecto.

Figura 4.29 Dial de nivel de combustible del Cuadro de información

Imagen que contiene Forma

El contenido generado por IA puede ser incorrecto.

Figura 4.30 Dial de temperatura del motor del Cuadro de información

### Comunicación alámbrica

Para la comunicación alámbrica se hace uso de la librería *pyserial* [41] la cual proporciona la funcionalidad necesaria para la transferencia de información a través de un puerto serie. La configuración necesaria se guarda en un archivo de variables de entorno, lo cual facilita ejecutar el programa en sistemas distintos.

La configuración debe de ser la misma que la utilizada en el microcontrolador, a excepción del puerto serie. Este vendrá determinado por la interfaz a la que conectamos el microcontrolador y del sistema operativo. En este caso durante el desarrollo en Windows el adaptador usado es detectado como COM6. Con esta configuración se declara un objeto de tipo *Serial* que representa la interfaz serie, el cual declaramos como global para poder ser usado en la función que se encarga de leer y actualizar los valores de la interfaz.

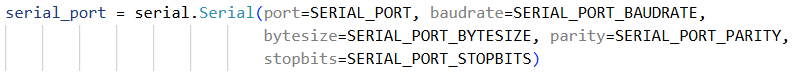


Figura 4.31 Declaración del puerto serie

La función *process\_data()* se encarga de leer los valores que llegan al puerto serie y utiliza los métodos específicos para actualizar cada parte de la interfaz. Del puerto serie se leen 41 bytes, que es el mismo número que se manda desde el microcontrolador.

Antes de actualizar los datos, se tienen que procesar. Para esto decodifican los bytes como un string, luego se eliminan los caracteres que no necesitamos como el salto de línea o valores nulos. Por último, se usa la función *split()* que permite transformar un string en una lista, donde cada elemento se extrae de la cadena usando el separador definido como argumento de la función.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.32 Función process\_data

### Función principal

Todos los componentes y funcionalidades del salpicadero virtual se integran en una función principal que se encarga de inicializar todos los componentes del salpicadero y de la comunicación alámbrica.

Se declaran las variables globales, el puerto serie, y los componentes de la interfaz. Primero se declara el objeto *window* que hace referencia a la ventana donde se dibujan los elementos de la interfaz. Se debe de definir el tamaño de la ventana y la posición en la que aparece en la pantalla. El tamaño de la pantalla se obtiene de variables de entorno que indican el ancho y el alto de la ventana. Con esta información podemos calcular la posición del punto que indica el inicio de la ventana con respecto de la pantalla. Teniendo en cuenta que el eje de la pantalla, o inicio de las coordenadas se encuentra en la esquina superior izquierda, y que el punto que se usa para indicar el comienzo de la ventana coincide con la esquina superior izquierda de esta, se usa la siguiente fórmula para que la ventana aparezca de una forma centrada en la pantalla.

Donde *c* es la coordenada del punto que queremos obtener, *s* es el la altura o el ancho dependiendo de la coordenada que queramos calcular, y *w* es la altura o ancho de la ventana que vamos a dibujar. Tras hacer esta calculo para el ancho y la altura obtenemos el punto que dibuja la ventana centrada en nuestra ventana.

Tras esto se definen los elementos de la interfaz en el siguiente orden: cuentarrevoluciones, velocímetro, indicador de distancia de seguridad y cuadro de información del vehículo.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4.33 Declaración de los elementos de la interfaz

Por último, se crea un hilo para la función de lectura y actualización de los valores en la interfaz y se llama a la función *mainloop()* del objeto *window*, bloqueando el hilo de ejecución principal y dibujando la interfaz en la pantalla.

Diagrama

El contenido generado por IA puede ser incorrecto.

Figura 4.34 Salpicadero digital completo

## Resultados

Para validar el funcionamiento de cada componente se desarrollaron una serie de pruebas. Todas ellas cuentan con un enlace a un video que muestra el desarrollo de la prueba en concreto. A continuación, se describen las pruebas y se incluyen las fotografías necesarias para dar contexto a las mismas.

* Prueba cuentarrevoluciones: Para verificar el correcto funcionamiento del cuentarrevoluciones se conecta un motor de corriente continua a un pin de 5V del microcontrolador, el motor cuenta con la rueda mostrada en la *Figura 4.7*. Se pudo comprobar que la medición obtenida por el microcontrolador y la representación mostrada en el salpicadero coincidían. <https://youtu.be/VIifqyUZO1o>



Figura 4.35 Montaje de sensor fotoeléctrico y motor DC

* Prueba velocímetro y distancia recorrida: Dado que el microcontrolador no está montado en una plataforma móvil, se instaló temporalmente en un vehículo teledirigido. Junto con unos cables de longitud considerable se mantuvo la conexión alámbrica al ordenador, y se trató de circular en línea recta. La velocidad máxima del vehículo radio control es de 8km/h o 2m/s. Se pudo observar como el microcontrolador indico una velocidad de 1.8m/s y el velocímetro quedo cerca de la marca de 2m/s. Teniendo como referencia una cinta métrica colocada en el suelo el vehículo radiocontrol se detuvo al alcanzar un metro, y como se observa en el video el indicador de distancia indico una distancia recorrida de un metro. <https://youtube.com/shorts/RY13U1XfVSU>

Una maleta en el suelo

El contenido generado por IA puede ser incorrecto.

Figura 4.36 Posición inicial del vehículo radiocontrol conectado al PC

Un ratón de computadora

El contenido generado por IA puede ser incorrecto.

Figura 4.37 Montaje del microcontrolador sobre el vehículo radiocontrol

Un piso de madera

El contenido generado por IA puede ser incorrecto.

Figura 4.38 Punto de referencia donde se alcanza un metro de distancia

* Prueba distancia de seguridad: Se coloca una caja de cartón delante de cada sensor junto con un metro para observar si se cumple el comportamiento esperado. Se comprueba que a la mínima distancia hay 4 arcos, teniendo en cuenta que hay como máximo 4 arcos y la distancia máxima configurada es 50cm deberíamos ver que aproximadamente cada 12cm se elimina un arco hasta llegar a los 50cm donde deberían de desaparecer todos. A una distancia de 13cm desaparece el primer arco, a los 24cm desaparece el siguiente, a los 38 desaparece el tercer arco y a los 49cm desaparece el ultimo arco, verificando el correcto funcionamiento de los sensores y el salpicadero. <https://youtu.be/1Elf1tIOCQQ>
* Prueba nivel de llenado del depósito de combustible: Se comprueba que para el valor mínimo de resistencia del potenciómetro se muestra como vacío el tanque de combustible, para el valor máximo que el tanque se muestra lleno, para la mitad del valor máximo se muestra el tanque a la mitad y que la transición entre estos valores no es demasiado abrupta. Verificando así la precisión del valor medido y el dibujado en la interfaz. <https://youtu.be/HR7XzW8RShM>
* Prueba temperatura motor y habitáculo: con un mechero se calientan los sensores hasta que se alcanzan los 50ºC, para el indicador del habitáculo se comprueba que se dibuja 50ºC en la interfaz. Y para el indicador de la temperatura del motor se comprueba que la aguja indica la mitad del dial, ya que el dial está configurado para mostrar valores entre 0 y 100. Tras esto se comprueba que el indicador de la temperatura se enfría hasta alcanzar la temperatura de la estancia. <https://youtu.be/YosbjsyeqZ4>

# ANÁLISIS DEL PROYECTO

En este capítulo se realizará un estudio del proyecto, más concretamente se centrará en el impacto del proyecto teniendo en cuenta la viabilidad de este. Este análisis se desarrollará centrado en el estudio del contexto socioeconómico en el que se exploraran las implicaciones sociales y económicas del desarrollo del proyecto.

Este análisis también incluirá un presupuesto detallado que desglosa el coste asociado a la compra de materiales para el desarrollo del microcontrolador, así como el coste relacionado con licencias de software y personal.

## Contexto socioeconómico

Como se ha comentado en la introducción los avances en tecnología y los cambios en como interactuamos con la tecnología han tenido su influencia en los vehículos que mueven nuestra sociedad. Ya sea para desplazarse al lugar de trabajo, para desarrollar una actividad económica o por placer, los vehículos han seguido evolucionando y mejorando en aspectos como la eficiencia y la seguridad. Y todas estas mejoras han llegado de mano de la tecnología que permite mejorar el rendimiento, eficiencia y analizar información en tiempo real para mejorar la seguridad de los vehículos que nos rodean.

En el ámbito económico, la introducción de nuevas tecnologías ha permitido la mejora de los vehículos haciéndolos más eficientes, tanto en consumo de combustible, como en un aumento de la vida útil de los componentes. Además, desde la UE y la DGT se han impulsado normativas que obligan a los fabricantes a incorporar elementos de seguridad y asistencia a la conducción [42]. Esto implica la expansión de un mercado en la que los fabricantes de vehículos buscaran las soluciones más eficientes a la hora de ser implementadas en sus modelos.

El avance de las tecnologías de conducción autónoma que están llegando a la industria auguran la necesidad de capturar más información para la mejora de estos sistemas. Si se tiene en cuenta que la actividad se desarrolla en un entorno altamente regulado, solo se puede esperar que las regulaciones obliguen a los fabricantes a mejorar estos sistemas, los cuales necesitaran del desarrollo e investigación de nuevas tecnologías relacionadas con la captura de información de los vehículos.

## Presupuesto

Para terminar con el análisis se realiza el estudio detallado de los gastos ocurridos durante el desarrollo del proyecto. A continuación, se desglosa el coste para el desarrollo del salpicadero virtual y la captura de datos con microcontrolador.

### Coste de elementos electrónicos

El coste de los elementos electrónicos usados para el desarrollo del proyecto incluye los sensores, el microcontrolador y el adaptador para la comunicación serie.

|  |  |  |
| --- | --- | --- |
| Concepto | Unidades | Importe |
| Placa de desarrollo STM32F407G-DISC1 | 1 | 56.99€ |
| Conversor USB a TTL | 1 | 15.49€ |
| HC-SR04 x3 | 1 | 7.99€ |
| MAX31865 PT100 | 2 | 11.99€ |
| DAOKAI LM393 x5 | 1 | 9.79€ |
| Potenciómetro | 1 | 3.62€ |

Tabla 5.1 Coste de elementos electrónicos

### Coste de licencias

Este coste incluye todas las licencias necesarias para el uso de software y subscripciones a servicios online que han sido usados para el desarrollo del proyecto. Aunque la mayoría de las licencias que se podrían necesitar para desarrollar el proyecto las proporciona la universidad, es importante tenerlas en cuenta.

|  |  |
| --- | --- |
| Concepto | Importe |
| Licencia Microsoft 365 para un mes | 10€ |

Tabla 5.2 Coste de licencias

### Coste de personal

Este coste refleja el tiempo que se ha consumido por cada persona implicada en el desarrollo del proyecto, teniendo en cuenta económicamente el valor aportado por cada persona y su relevancia el proyecto.

Dado que es el propio estudiante el que desarrolla este trabajo el coste ha sido de 0€. Pero para dar más contexto al presupuesto se va a añadir un cálculo orientativo. Tiendo en cuenta que la asignatura del TFG recoge un coste de 12 ECTS. Y que cada crédito equivale a 25 horas de trabajo se puede estimar un total de 300 horas. Si aplicamos las bases mínimas para la categoría profesional de Ingenieros y licenciados [43], estaríamos hablando de un salario base de 1929€ por mes.

Si suponemos una jornada laboral estándar, se estaría hablando de 38 días para completar el desarrollo del proyecto. Un mes suele tener unos 20 días laborables, es decir, en total serian 2 meses de trabajo.

|  |  |
| --- | --- |
| Concepto | Importe |
| Dos meses de salario bruto | 3858€ |

Tabla 5.3 Coste de personal

### Coste total

Teniendo en cuenta los costes reales de materiales, y los estimados para las licencias de software y el personal obtenemos el siguiente coste total.

|  |  |
| --- | --- |
| Concepto | Importe |
| Elementos electrónicos | 105,87€ |
| Licencias de software | 20€ |
| Coste de personal | 3858€ |
| **Total** | 3983.87€ |

Tabla 5.4 Coste total

El presupuesto total asciende a 3983,87€. Siendo el coste de personal lo más caro de todo el desarrollo. Esto se debe principalmente a que el coste humano suele ser lo más caro en este tipo de proyecto, es el ingeniero el que resuelve el problema. Y el resto de los gastos son medios para alcanzar esa solución.

También cabe destacar que el resto de los programas usados para el desarrollo del proyecto son de código abierto, lo cual quiere decir que la mayoría se pueden usar de forma gratuita por parte de particulares, y dependiendo de la licencia de cada uno también lo son para el uso por parte de empresas.

# CONCLUSIONES

Tras la finalización del proyecto de diseño de un salpicadero virtual en un PC con un microcontrolador. Se puede afirmar que los objetivos marcados se han cumplido de manera satisfactoria.

Los tres objetivos marcados se han desarrollado con éxito, y se ha obtenido un salpicadero virtual que presenta la información recibida de forma alámbrica desde el microcontrolador. Además, todos los parámetros marcados como información necesaria han sido representados en el salpicadero.

Se ha obtenido el conocimiento necesario para configurar todo lo relacionado con el microcontrolador y necesario para el desarrollo de esta parte del sistema. Que incluye la reprogramación de este, así como la escritura del código que captura la información y la transmite, o la conexión y configuración de distintos componentes.

También se ha conseguido transmitir información de forma alámbrica entre los dos dispositivos que forman parte del sistema. Esto contiene el conocimiento necesario para entender cómo funciona el protocolo de comunicación serie y todos los parámetros que lo afectan.

En cuanto al desarrollo del salpicadero se puede decir que cumple con su objetivo de presentar información de manera gráfica. Con un claro énfasis en la claridad y funcionalidad.

En conclusión, el trabajo ha cumplido con todos los objetivos, pero todavía hay margen para mejorar, lo que abre líneas futuras de trabajo. Este proyecto ha servido para establecer una base de conocimiento sobre la cual expandir el ecosistema de la obtención de datos de sensores en vehículos y la forma en la que esta información es presentada.

## Líneas futuras de trabajo

Aunque los objetivos del proyecto han sido completados, hay varias opciones de mejora que abren distintas líneas de trabajo para el futuro, con el fin de expandir el conocimiento obtenido del trabajo y aportar una mejor solución:

* Implementación del sistema en un vehículo a escala. Si se desarrollase, o aplicase una estructura física con la que representar un vehículo, y en la que poder integrar el sistema actual. Se conseguiría una plataforma más sencilla en la que presentar y probar las distintas funcionalidades del sistema. Además de posibilitar la opción de combinar distintas fuentes de datos, como por ejemplo la obtención de la distancia recorrida usando el método desarrollado en el sistema junto con la medición de distancia con un sensor en las ruedas del vehículo. Habilitando redundancia en la obtención de mediciones, así como la capacidad de comparar la precisión de cada implementación.
* Mejorar el diseño de la interfaz de usuario. La interfaz desarrollada para el salpicadero se centra en la funcionalidad, pero al igual que su simpleza puede ser un punto positivo también puede serlo negativo. Por ejemplo, no se suaviza de ninguna forma la transición entre los valores presentados al usuario, esto implica inmediatez en la actualización de los valores, pero no da ninguna información de la posible evolución de las distintas magnitudes. Lo cual no ayuda al usuario a predecir el comportamiento del vehículo.

# REFERENCIAS BIBLIOGRÁFICAS

[1] M.J. Mccourt. “1962-1973 Volkswagen Type 3 1500/1600”. Hemmings. Acceso: 28 de mayo de 2025. [En línea]. Disponible en: <https://www.hemmings.com/stories/1962-1973-volkswagen-type-3-1500-1600/>

[2] “¿Qué son las revoluciones por minuto (rpm) de un coche?”. RACE. Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://www.race.es/revoluciones-por-minuto-que-son>

[3] *Crankshaft Speed Sensor for Commercial Vehicles*. (s. f.). Acceso: 28 de agosto de 2025. [Imagen]. Disponible en: <https://www.bosch-mobility.com/en/solutions/sensors/crankshaft-speed-sensor/>

[4] B.M. MAPFRE. “Cómo funciona el velocímetro de un coche”. Motor. Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://www.motor.mapfre.es/accesorios/noticias-accesorios/como-funciona-el-velocimetro-de-un-coche/>

[5] *Wheel-speed Sensor*. (s. f.). Acceso: 28 de agosto de 2025. [Imagen]. Disponible: <https://www.bosch-mobility.com/en/solutions/sensors/wheel-speed-sensor/>

[6] “Temperature sensor for a wide temperature range”. Bosch Mobility. Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://www.bosch-mobility.com/en/solutions/sensors/temperature-sensor/>

[7] *Temperature Sensor NTC M12*. (s. f.). Acceso: 28 de agosto de 2025. [Imagen]. Disponible en: <https://www.bosch-motorsport.com/content/downloads/Raceparts/en-GB/53687307119081483.html>

[8] “Todo lo que necesitas saber sobre los medidores de gasolina”, *Farremetal Ltda*, s. f. Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://farremetal.com/2023/04/23/todo-lo-que-necesitas-saber-sobre-los-medidores-de-gasolina-como-funcionan-y-por-que-son-esenciales-para-tu-vehiculo/>

[9] *Sensor de nivel de combustible Ford E150 E250 E350 Van F150 F250 F350 Pickup 89-96*. (s. f.). Acceso: 28 de agosto de 2025. [Imagen]. Disponible en: <https://www.alpaimports.com/products/flotador-tanque-gasolina-89-96-ford-e150-e250-e350-van-f150-f250-f350-pickup>

[10] “Mazda3 5 Puertas: El Mild Hybrid definitivo”. Mazda España. Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://www.mazda.es/gama/mazda3/>

[11] Mazda. *Tablero de instrumentos y pantalla* (2025). Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://owners-manual.mazda.com/gen/es/mazda3/mazda3_8hs7sp19i/contents/05020700.html>

[12] Mazda. *Sistema de sensor de estacionamiento* (2025). Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://owners-manual.mazda.com/gen/es/mazda3/mazda3_8hs7sp19i/contents/05200100.html>

[13] Tesla. *Panel de instrumentos – Conducción*. Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://www.tesla.com/ownersmanual/models/es_mx/GUID-29A7E205-A689-41D1-B69C-3AE821CB70E7.html>

[14] Tesla. *Pantalla táctil* (2025). Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://www.tesla.com/ownersmanual/models/es_mx/GUID-518C51C1-E9AC-4A68-AE12-07F4FF8C881E.html#GUID-EAC53B57-E5F1-49C4-88AC-F1EC2F50CEBB>

[15] Mercedes. *Clase C Berlina Manual de instrucciones +* (2025). Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://www.mercedes-benz.es/passengercars/owners-manuals/es-es/pdf/mercedes-clase-c-berlina-2025-marzo-w206-mbux-instrucciones-de-servicio-1.pdf>

[16] T. Gravekamp. “STM32F407VET6 - STM32 F4VE V2.0”. STM32-base project. Acceso: 2 de enero de 2025. [En línea]. Disponible en: <https://stm32-base.org/boards/STM32F407VET6-STM32-F4VE-V2.0.html>

[17] “STM32F4DISCOVERY”. STMicroelectronics. Acceso: 24 de marzo de 2025. [En línea]. Disponible en: <https://www.st.com/en/evaluation-tools/stm32f4discovery.html>

[18] *Sensor de Distancia Ultrasónico HC-SR04*. (s. f.). Acceso: 12 de agosto de 2025. [Imagen]. Disponible en: <https://www.robotshop.com/es/products/sensor-de-distancia-ultrasonico-hc-sr04>

[19] *Ultrasonic Ranging Module HC - SR04*. Acceso: 5 de abril de 2025. [En línea]. Disponible en: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>

[20] Mouser. *DHT11 Humidty & Temperature Sensor*. Acceso: 10 de abril de 2025. [En línea]. Disponible en: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>

[21] *Data Sheet TF Pt100-3L* (2023). Acceso: 21 de abril de 2025. [En línea]. Disponible en: <https://www.stoerk-tronic.com/fileadmin/erp/dokumente/datenblaetter/en/900001_030.pdf>

[22] *Sensor de Temperatura RTD Platino – PT100 – 3 hilos 4×30mm*. (s. f.). Acceso: 12 de agosto de 2025. [Imagen]. Disponible en: <https://electronilab.co/tienda/sensor-de-temperatura-rtd-platino-pt100-3-hilos/>

[23] *MAX31865 RTD Platinum Resistance Temperature Detector Module*. (s. f.). Acceso: 12 de agosto de 2025. [Imagen]. Disponible en: <https://m.media-amazon.com/images/G/30/apparel/rcxgs/tile._CB483369954_.gif>

[24] *MAX31865 RTD-to-Digital Converter* (2015). Acceso: 12 de agosto de 2025. [En línea]. Disponible en: <https://www.analog.com/media/en/technical-documentation/data-sheets/max31865.pdf>

[25] “Technical Explanation for Photoelectric Sensors,” OMRON, CSM\_Photoelectric\_TG\_E\_8\_3. Acceso: 12 de agosto de 2025. [En líena]. Disponible en: <https://www.ia.omron.com/data_pdf/guide/43/photoelectric_tg_e_8_3.pdf>

[26] *Sensor fotoeléctrico Daokai LM393*. (s. f.). Acceso: 12 de agosto de 2025. [Imagen]. Disponible en: <https://cdn.shopify.com/s/files/1/0409/9041/products/gt_04ac7afe-7baf-4597-a7f9-7aa13133c359_grande.png?v=1475275249>

[27] *Codificador circular*. (s. f.). Acceso: 12 de agosto de 2025. [Imagen]. Disponible en: <https://m.media-amazon.com/images/I/41vOLwOceiL._UF350,350_QL80_.jpg>

[28] STMicroelectronics. *MEMS Digital Output Motion Sensor: Ultra-low-powered High-performance Three-axis “nano” Accelerometer* (2017). Acceso: 15 de mayo de 2025. [En línea]. Disponible en: <https://www.st.com/resource/en/datasheet/dm00040962.pdf>

[29] *Potentiometer.jpg - Wikimedia Commons*. (s. f.). Acceso: 12 de agosto de 2025. [Imagen]. Disponible en: <https://commons.wikimedia.org/wiki/File:Potentiometer.jpg>

[30] “Figma: The Collaborative Interface Design Tool”. Figma. Acceso: 10 de junio de 2025. [En línea]. Disponible: <http://www.figma.com>

[31] STMicroelectronics. *STM32CubeIDE*. (Versión 1.19.0). [En línea]. Disponible en: <https://www.st.com/en/development-tools/stm32cubeide.html>

[32] *DSD TECH SH-U09C5 Cable convertidor USB a TTL UART*. (s. f.). Acceso: 12 de agosto de 2025. [Imagen]. Disponible en: <https://m.media-amazon.com/images/I/71TnwREtkTL._AC_SX679_.jpg>

[33] STMicroelectronics. *Getting Started With TIM* (2024). Acceso: 5 de abril de 2025. [En línea]. Disponible en: <https://wiki.st.com/stm32mcu/wiki/Getting_started_with_TIM>

[34] STMicroelectronics. *UM1472 User Manual* (2025). Acceso 5 de abril de 2025. [En línea]. Disponible en: <https://www.st.com/resource/en/user_manual/um1472-discovery-kit-with-stm32f407vg-mcu-stmicroelectronics.pdf>

[35] STMicroelectronics. *RM0090 Reference Manual* (2024). Acceso: 5 de abril de 2025. [En línea]. Disponible en: <https://www.st.com/resource/en/reference_manual/dm00031020-stm32f405-415-stm32f407-417-stm32f427-437-and-stm32f429-439-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf>

[36] A. Pini. “Por qué y cómo usar la interfaz periférica serial para simplificar las conexiones entre distintos dispositivos”. DigiKey. Acceso: 25 de abril de 2025. [En línea]. Disponible en: <https://www.digikey.es/es/articles/why-how-to-use-serial-peripheral-interface-simplify-connections-between-multiple-devices>

[37] N. Askari. *MAX31865 Library*. (2023). [En línea]. Disponible en: <https://github.com/nimaltd/max31865>

[38] MYaqoobEmbedded. *STM32-Tutorials/Tutorial 27 - Motion 3-Axis Accelerometer LIS3DSH*. (2019). [En línea]. Disponible en: <https://github.com/MYaqoobEmbedded/STM32-Tutorials/tree/master/Tutorial%2027%20-%20Motion%203-Axis%20Accelerometer%20LIS3DSH>

[39] fritzing. *fritzing-app.* (Versión 0.9.6). [En línea]. Disponible en: <https://github.com/fritzing/fritzing-app>

[40] Python Software Foundation. “tkinter a Python interface to Tcl/Tk”. Python documentation. Acceso: 17 de mayo de 2025. [En línea]. Disponible en: <https://docs.python.org/es/3/library/tkinter.html>

[41] *pySerial*. (Versión 3.4). [En línea]. Disponible en: <https://github.com/pyserial/pyserial/tree/master>

[42] “Sistemas avanzados de ayuda a la conducción (ADAS)”. DGT. Acceso 28 de agosto de 2025. [En línea]. Disponible en: <https://www.dgt.es/muevete-con-seguridad/sistemas-avanzados-ayuda-conduccion/Sistemas-avanzados-de-ayuda-a-la-conduccion-ADAS-/>

[43] “Seguridad Social: Cotización / Recaudación de Trabajadores”. Seguridad Social. Acceso: 28 de agosto de 2025. [En línea]. Disponible en: <https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#36538>