UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

INTERPRETACIÓN Y ANÁLISIS DE DATOS DE LA RED CAN-BUS EN DISPOSITIVOS ANDROID.

INTERPRETATION AND DATA ANALYSIS OF CAN-BUS NETWORK ON ANDROID DEVICES.

Realizado por
SERGIO JOSÉ MARTÍNEZ PERALES

Tutorizado por
DAVID SANTO ORCERO

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, FEBRERO 2020

TRABAJO FIN DE GRADO

Interpretación y análisis de datos de la red CAN-BUS en dispositivos Android

Sergio José Martínez Perales

Tutor:
Prof. Dr. David Santo Orcero

Palabras Clave:
CAN BUS, ANDROID, OBD, OBDII, TELEMETRÍA, ELM327.

Resumen

En este TFG se ha desarrollado una aplicación que muestra, mediante una interfaz gráfica, varios parámetros relevantes sobre del funcionamiento de un vehículo en tiempo real. Además, permite al usuario representar los datos en gráficos de líneas para analizar el funcionamiento del vehículo. La aplicación se ha construido utilizando una API desarrollada en Java para comunicarse con un adaptador OBD mediante comandos a través de una conexión Bluetooth.

Para el desarrollo de la aplicación resultante se ha realizado una investigación acerca del funcionamiento del bus CAN que utilizan los vehículos de producción de hoy en día y del funcionamiento del sistema estándar OBD que integran todos los vehículos en Europa desde el año 2005, incluidos vehículos pesados.

Se ha repasado la arquitectura del sistema operativo Android y se ha descrito el proceso de diseño de la aplicación OBDII Dashboard, desarrollada en este TFG, mostrando los distintos casos de uso posibles y acompañando toda la memoria con un manual de usuario de la aplicación.

Abstract

This project describes the development of an application that provides a dashboard interface which shows relevant parameters about vehicle operation in real time. In addition, the application allows the user to represent the collected data using line charts in order to analyse if vehicle operation is correct. This application uses an OBD command Java API in order to communicate, using a Bluetooth connection, with vehicle Electronic Control Unit.

To achieve this target, some research job, about the CAN bus networks that today vehicles use and about the behaviour of the OBD standard system that vehicles from European Union include since 2005, including heavy weight vehicles like trucks, has been done.

In addition some research about Android OS architecture has been done and, finally, the design process of OBDII Dashboard application, developed in this project, has been described, showing all the possible use cases. A user guide has been included at the end of this dissertation.

Índice general

Capítulo 1

Introducción

La proliferación de sistemas electrónicos en la industria automotriz hace necesaria la existencia de protocolos de comunicación entre estos sistemas. En la actualidad, Controller Area Network bus (en adelante, CAN bus) se erige como protocolo estándar en la industria automotriz para la conexión de los sistemas electrónicos de los vehículos.

CAN bus es un protocolo orientado a buses de campo desarrollado por Bosch en la década de 1980. Algunas características deseables en un protocolo de comunicación para sistemas en automóviles son que sea robusto en cuanto a consistencia de datos, que sea flexible en cuanto a los tipos de sistemas que se comunican, que ofrezca una cierta garantía en tiempos de latencia y que sea capaz de detectar errores en los nodos de la red. CAN bus reúne todas estas características, permitiendo que las comunicaciones entre sistemas se realicen por medio de un bus serie, ahorrando así costes en la infraestructura de la red de comunicación y simplificándola enormemente frente a otras soluciones. Se encarga de conectar sensores, actuadores y una o varias unidades de control, de forma que son los sensores y las unidades de control las que escriben en el bus.

En Europa, desde la primera mitad de la década del 2000, según la Directiva 98/69EG, es de obligada incorporación en todo vehículo, ya sea con motor de gasolina o de gasóleo, un puerto de diagnóstico a bordo u OBD, por sus siglas en inglés. La gran mayoría de vehículos en la actualidad incluyen un puerto OBD II que permite extraer información de los distintos sensores leyendo los mensajes que estos envían por la red CAN bus. Existen multitud de dispositivos que se pueden utilizar para comunicarse con la red CAN bus mediante la interfaz que ofrece OBD II. Es común encontrar dispositivos basados en el ELM327, que es

un microcontrolador que permite conectar un ordenador al puerto OBD para, mediante comandos, obtener información de los sensores, leer códigos de error grabados en la unidad de control del motor – ECU por sus siglas en inglés – o incluso borrar los registros que la ECU tiene de esos errores.

Con el auge de los teléfonos móviles inteligentes, se hace interesante disponer de una aplicación que permita la conexión con un dispositivo OBD mediante conexión inalámbrica, y que se encargue de mostrar los datos más interesantes sobre los sensores disponibles en un vehículo, de manera que se pueda llevar un control más accesible para todo tipo de usuario sobre el funcionamiento de los sistemas del vehículo.

En este Trabajo de Fin de Grado (TFG, en lo que sigue) se ha creado una aplicación para Android que es capaz de mostrar, de forma intuitiva y sencilla, información útil sobre los sistemas de un vehículo que están conectados entre sí mediante la línea CAN bus. La aplicación se limita a leer el bus para recoger información de los distintos sensores y mostrarla de forma conveniente en un dispositivo que ejecute Android; no escribe en el bus ya que, de hacerlo, existe la posibilidad de que cualquier sistema de apoyo a la conducción falle.

Para desarrollar la aplicación, se ha estudiado el protocolo CAN bus, en el nivel físico y de enlace de datos, qué tipos de trama se pueden encontrar y cómo se utilizan para realizar la lectura de los datos de los sensores y actuadores. Esta tarea se complica con la dificultad inherente a la existencia de distintas implementaciones del protocolo según el fabricante, con lo que se han habilitado en la aplicación varias opciones para la conexión y un modo automático para la selección del protocolo adecuado.

Se ha estudiado el funcionamiento de OBD II como interfaz de conexión entre el dispositivo encargado de mostrar la información al usuario final y el propio CAN bus del vehículo. Se han revisado aspectos técnicos de la interfaz OBD II y se han explorado algunas de las opciones comerciales que hay en el mercado, con el objetivo de dar algunas pautas sobre la conveniencia de unos u otros adaptadores OBD. Además, se ha estudiado la forma en que un dispositivo puede comunicarse con un adaptador OBD utilizando los comandos OBD para hacer peticiones a la ECU de un vehículo.

La conexión con el dispositivo Android se establece utilizando un adaptador Bluetooth conectado al puerto de diagnóstico del vehículo, que recoge las tramas de la línea CAN bus y hace las peticiones necesarias. Es en el dispositivo Android donde se procesan dichas tramas para los distintos análisis y fun-

cionalidades, utilizando una librería ya existente para la comunicación con el adaptador OBD. La aplicación tiene una interfaz gráfica que sirve para ofrecer al usuario tanto un cuadro de mandos del vehículo como un ordenador de abordo. Las aplicaciones pueden ir desde el análisis del funcionamiento del vehículo hasta el diagnóstico de problemas específicos de los sistemas que lo integran utilizando la función de recogida de códigos de error o analizando los datos que la aplicación guarda en cada sesión.

Objetivos

El objetivo de este TFG es crear una aplicación para dispositivos Android que permite, mediante el puerto de diagnóstico, mostrar información relevante sobre el funcionamiento de un vehículo, ya sea para cuestiones de telemetría y análisis de funcionamiento de los sistemas que componen el vehículo durante su conducción, o para cuestiones de diagnóstico de errores y averías. La conexión Bluetooth con el puerto OBD se hace mediante un adaptador OBD que recoge y envía por Bluetooth las tramas de la línea CAN bus al dispositivo Android utilizando un lenguaje de comandos OBD.

La aplicación permite la conexión con dispositivos Bluetooth conectados al puerto de diagnóstico de un vehículo con el objetivo de establecer una conexión inalámbrica. Una vez establecida la conexión, la aplicación muestra un menú que permite realizar tareas de análisis o simplemente mostrar un cuadro de mandos configurable.

El estudio del protocolo CAN bus ha sido una parte importante del TFG, siendo éste el protocolo de comunicación que permite recoger la información que se muestra en la aplicación. La interfaz OBD ha sido también una parte fundamental para habilitar la comunicación entre el vehículo y el dispositivo Android.

Organización de la memoria

Se ha estructurado la memoria de la siguiente forma:

- En la presente introducción se enumeran los objetivos de este trabajo de fin de grado.

- En el segundo capítulo, se analiza el estado del arte.
- En el tercer capítulo se comentan las tecnologías empleadas.
- En el cuarto capítulo, se describe el análisis y el diseño del sistema que se ha implementado.
- En el quinto capítulo, se comentan las conclusiones a las que se ha llegado tras la realización de este trabajo.
- Se incluye una lista con las referencias consultadas y la bibliografía empleada.
- Se incluye un apéndice con un pequeño manual de usuario de la aplicación desarrollada.

Capítulo 2

Estado del arte

2.1. Introducción

El uso de la electrónica en la industria del automóvil se remonta a finales de la década de 1970. Al principio, los fabricantes se limitaban a incorporar algunos sensores a los motores con el objetivo de verificar su correcto funcionamiento. Para el manejo de estos sensores, se utilizaban unidades de control que permitían tomar decisiones según los valores reportados por los sensores en cada momento.

En 1966, para controlar las emisiones que ocasionaban problemas de polución en la ciudad de Los Ángeles, el estado de California de Estados Unidos exigió la implementación de sistemas de control de emisiones en todos los vehículos posteriores a este año.

En 1970, el Congreso de Estados Unidos creó la EPA (Environmental Protection Agency) y esta comenzó el desarrollo de estándares para el control de la emisión de gases en vehículos y la reducción de la contaminación.

En 1988, la California Air Resources Board determinó que, para todo vehículo de gasolina, el fabricante debía instalar un sistema de diagnóstico de emisiones y averías en cada nuevo vehículo sujeto a las regulaciones de la Sección 1968 o 1968.1, Título 13, del California Code of Regulations.

Debido a esta normativa, surgió la necesidad de utilizar una interfaz que estandarizara la monitorización de los componentes dedicados a controlar las emisiones en los vehículos. OBD fue la primera generación del sistema de diagnóstico OBD (On-Board Diagnostics). Su primera especificación es de 1983 y la implementación en vehículos norteamericanos comenzó en 1988 con motivo de

las regulaciones de la California Air Resources Board para reducir la contaminación del aire.

En 1996 llega la segunda generación del sistema OBD, OBD II, que implementa limites más estrictos en el control de emisiones contaminantes y provee de más funciones de diagnóstico al sistema OBD.

En Europa no se implanta OBD hasta el año 2000, en el que la Directiva 98/69EG obliga a los automóviles con motor de gasolina a incorporar un sistema de diagnóstico de emisiones, aplicándose esta misma directiva posteriormente a los automóviles con motor diésel en el año 2003 y a los camiones en el año 2005. La introducción de OBD en Europa se hace ajustándose al OBD II americano de 1996

OBD es un sistema que se encarga de monitorizar las emisiones de un vehículo y, para ello, se sirve de un protocolo de comunicación que se negocia con las unidades de control del motor y otros elementos del vehículo. En la actualidad, el protocolo de comunicación que utilizan la mayoría de fabricantes para comunicar sus sistemas en los vehículos es CAN, también conocido como CAN bus, por ser éste un protocolo para buses de campo.

El desarrollo del protocolo comenzó en 1983 por la empresa alemana Robert Bosch GmbH, también conocida como Bosch, y fue en 1986 cuando se lanzó al mercado oficialmente durante el congreso de la Sociedad de Ingenieros Automotrices, más conocida como SAE, por sus siglas en inglés. El protocolo se fue refinando hasta que en 1991, Bosch publicó la especificación CAN 2.0, que está disponible como documentación de libre acceso a través de Bosch.

En 2011, la empresa alemana, en cooperación con fabricantes de la industria del automóvil, comenzó el desarrollo de CAN-FD (de Flexible Data-rate), una especificación de CAN retrocompatible con la capacidad de transmitir datos a una velocidad mayor a la máxima posible utilizando la especificación CAN 2.0.

Las definiciones tanto del protocolo CAN clásico como del protocolo CAN-FD están recogidas en el conjunto de normas ISO 11898-1.

En la actualidad, en Europa, en las inspecciones periódicas que se realizan a los vehículos, como es la Inspección Técnica de Vehículos (ITV) en España, se realizan análisis por medio del puerto OBD de los vehículos para controlar los límites de las emisiones y cualquier tipo de irregularidad y manipulación sobre el vehículo inspeccionado. Además, mediante el Reglamento de Ejecución 2019/621 de la Unión Europea, a partir del 20 de mayo de 2020 será obligatorio, para todos los fabricantes de vehículos, facilitar todos los datos y documentación

técnica necesarios para realizar correctamente los análisis en las inspecciones técnicas.

2.2. Electrónica en el automóvil.

Desde hace unos 50 años, en la década de 1970, la mayor parte de las innovaciones en la industria del automóvil han venido de la mano de la electrónica. En 2002, más de un $25\,\%$ del valor de fabricación de un vehículo actual se debía a la electrónica del automóvil y alrededor de un $80\,\%$ de las innovaciones tenían implicación directa de la electrónica y el software.

En los últimos años, la inclusión de sistemas de asistencia a la conducción y de confort no ha hecho más que aumentar, dejando las innovaciones mecánicas relegadas a un segundo plano, aunque los avances en disminución de emisiones en la última década han sido bastante notables.

La abundancia de sistemas electrónicos a bordo de los automóviles de hoy en día origina unas necesidades de alta conectividad que, de haber seguido utilizando conexiones punto a punto entre sistemas, habrían originado serios problemas de espacio, peso y costes debidos a la gran cantidad de cableado necesario para satisfacer dichas necesidades. Se estima que, en un vehículo medio, cada $50\,\mathrm{kg}$ de peso extra en cableado incrementa el consumo en $0,2\,\mathrm{l}$ de combustible por cada $100\,\mathrm{km}$ recorridos. En un comunicado de prensa de Motorola en 1998, se calculó que, en un vehículo de gama media-alta, el uso de conexiones punto a punto suponía un extra de $15\,\mathrm{kg}$ por cada puerta.

La solución al problema de cableado de las unidades electrónicas en los automóviles que se comparaba en ese comunicado de prensa era el uso de redes de control que funcionan mediante protocolos de comunicación en serie. El uso de estas redes permite reducir el cableado, de manera simplificada, a simplemente un par de cables que conforman la red y los cables correspondientes para conectar cada componente electrónico a dicha red, un cable por cada cable que conforma la red, es decir, 2 cables por cada unidad de control y dos cables que conforman la red y a los que se conectan las unidades de control. En el comunicado de prensa de Motorola se observó que el uso de este tipo de redes reducía, por cada puerta, unos $15\,\mathrm{kg}$ en cableado y, además, permitía mejorar los sistemas incluidos en cada una de las puertas sin el sacrificio de aumentar el cableado de forma considerable.

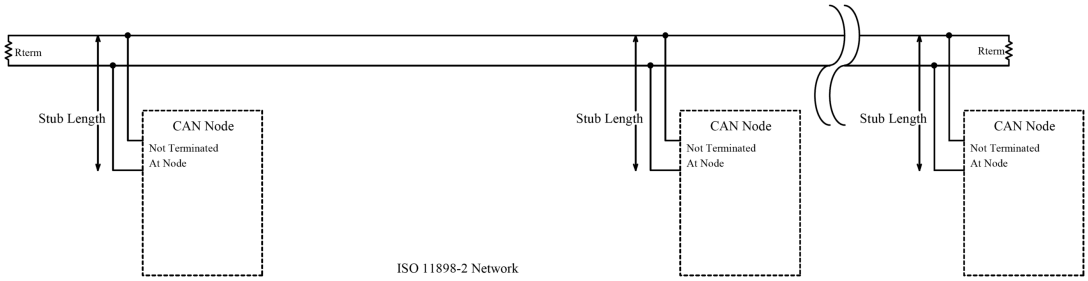


Figura 2.1: Red de control CAN. Fuente: Wikipedia.

De acuerdo con [LH02, p. 91], desde finales de 1998, los sistemas de conducción dinámica que típicamente habían sido controlados por sistemas puramente mecánicos o hidráulicos, han pasado a estar manejados por actuadores electrónicos. Sistemas como el ABS (del alemásn, Antiblockiersystem), el TCS (del inglés, Traction Control System) o el ESP (del alemán, Elektronisches Stabilitätsprogramm) son ahora más sofisticados, eficientes y fiables gracias a las unidades de control electrónicas incluidas en los vehículos de hoy en día.

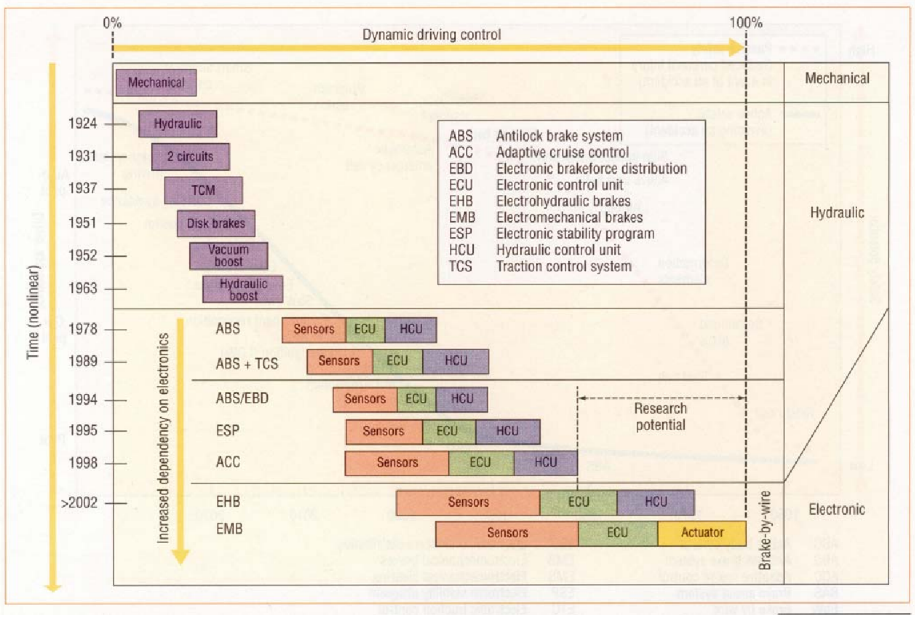


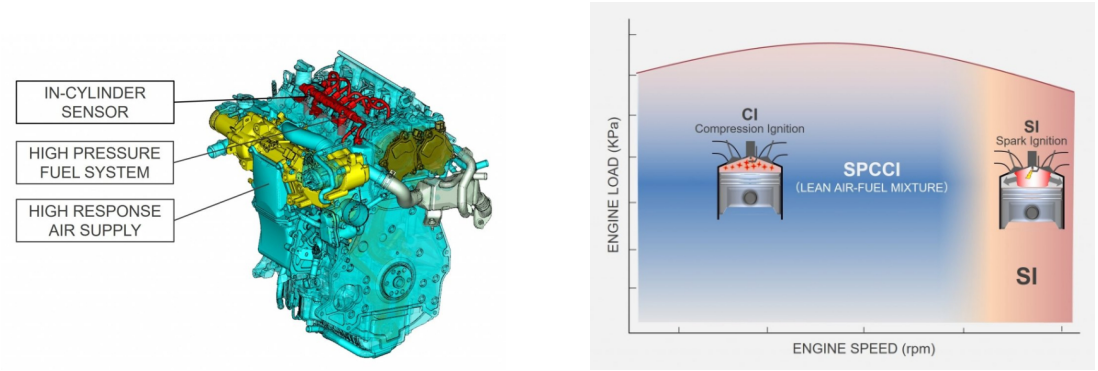
Figura 2.2: Evolución de los sistemas de control dinámico de conducción [LH02, Figura 2, p. 91].

Los sistemas electrónicos, según [NSL17, p. 25] se pueden clasificar en dominios funcionales, atendiendo a los componentes del vehículo involucrados y a los requerimientos de tiempo y capacidad de computación:

- **Dominio del tren automotriz:** Referido a todos los sistemas que intervienen en el movimiento longitudinal del vehículo. Por ejemplo, los sistemas que controlan el motor de acuerdo a las peticiones realizadas por el conductor como son las aceleraciones y las frenadas, mediante los pedales del acelerador y el freno respectivamente o los sistemas que controlan la transmisión, con todos los sistemas relacionados como el diferencial, la caja de cambios, etc. Estos sistemas se basan en parámetros como la temperatura ambiente, el nivel de oxigeno, las emisiones contaminantes del sistema de escape, la posición de los pedales de acelerador y freno, el número de revoluciones por minuto del motor y la velocidad del vehículo, etc.

  Sirviéndose de parámetros como los comentados anteriormente, los sistemas del dominio del tren automotriz pueden controlar parámetros como el consumo de combustible, parámetros relacionados con las ayudas a la conducción, o incluso los tiempos de apertura y cierre de válvulas en motores con sistemas de distribución variable. Sistemas de este estilo son responsables, en parte, de avances técnicos que mejoran la eficiencia de los motores y reducen, por tanto, las emisiones contaminantes, como ocurre con los motores SPCCI (Spark Controlled Compression Ignition) desarrollados por Mazda.

  Los exhaustivos controles necesarios para manejar este tipo de motores son posibles gracias a los sistemas electrónicos que se encargan de la gestión del funcionamiento de dichos motores. En este caso concreto, el sistema de control se sirve de sensores en el interior de cada cilindro, como se ve en el modelo 3D de un motor SPCCI que se muestra en la figura 2.3(a) (ver también [Gre]), que controlan la temperatura y compresión de la mezcla de gasolina y aire, permitiendo dos modos de funcionamiento como se observa en el esquema de la figura 2.3(b): ignición por compresión a bajas revoluciones por minuto (RPM) e ignición por chispa originada por una bujía a altas RPM.



(a) Motor con tecnología SPCCI que utiliza sensores en cada cilindro.

(b) Modos de funcionamiento de un motor con tecnología SPCCI.

Figura 2.3: Fuente: https://www.greencarcongress.com

- **Dominio del chasis:** Referido a todos los sistemas que se encargan del control de la interacción del automóvil con el firme. Elementos como las ruedas o la suspensión forman parte de este dominio. Sistemas de ayuda a la conducción como son los sistemas de control de tracción, sistemas de control 4WD (4 Wheel Drive) para vehículos con tracción a las cuatro ruedas, sistemas de control de estabilidad ESP o sistemas de prevención de bloqueo de ruedas en la frenada ABS son algunos ejemplos de los sistemas que se pueden conseguir con el uso de sistemas electrónicos. No son exclusivos de vehículos con sistemas electrónicos de control sofisticados. Por ejemplo, el concepto del ABS se remonta a la década de 1950, en la que los sistemas electrónicos de control en vehículos aun no eran posibles con la tecnología de la época. En la actualidad, el mismo ABS se sirve de sensores que detectan la velocidad de giro de cada rueda y las condiciones de adherencia del firme, permitiendo, mediante técnicas de lógica difusa, decidir cuánta presión de frenado debe llegar a cada freno con el objetivo de no bloquear ninguna de las ruedas y optimizar así la frenada consiguiendo parar el vehículo en el menor espacio posible y permitiendo al conductor hacer cambios de dirección si fuese necesario. El ABS es un buen ejemplo de sistema en el que intervienen sensores de dos dominios distintos, siendo por tanto primordial que sea posible la comunicación entre unidades de control de los distintos dominios.

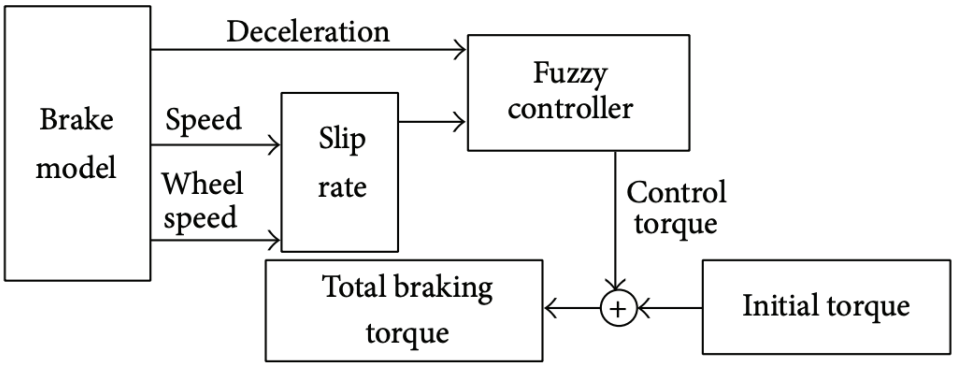
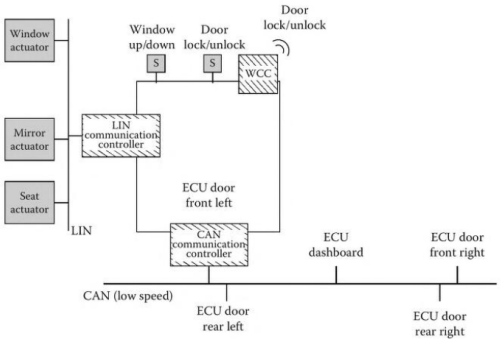


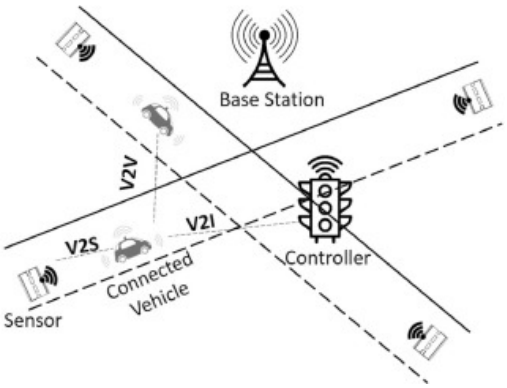
Figura 2.4: Esquema de un sistema ABS [ZLQ14, FIGURE 8].

- **Dominio de la carrocería:** Referido a todos los sistemas integrados que no están relacionados con el control de los movimientos del vehículo. Algunos de estos sistemas serían las luces del vehículo, los motores de apertura y cierre de ventanas, los actuadores de apertura y cierre de seguridad de las puertas, posición de asientos, posición de espejos, limpiaparabrisas, etc. Entre estos, hay sistemas que requieren que se cumplan una serie de restricciones de tiempo, por lo que surge la necesidad de la existencia de subsistemas de sensores y componentes mecatrónicos conectados a una pequeña red de alta velocidad que cumple con los requisitos exigidos para la tarea a realizar y que, a su vez, se conectan por medio de una unidad de control a la red de baja velocidad donde se conectan los sistemas del dominio de la carrocería.

  Un buen ejemplo de sistema con altas restricciones de tiempo sería el sistema de bloqueo y desbloqueo de puertas del vehículo, que debe responder, en ocasiones, a una señal de entrada que llega al sistema de forma inalámbrica y que se transfiere a través de la red CAN de baja velocidad a la unidad de control de bloqueo y desbloqueo de puertas, donde sensores y actuadores mecatrónicos se comunican en una red de alta velocidad para ejecutar la acción necesaria. En la figura 2.5(a) se muestra un esquema en el que se diferencian estos distintos medios utilizados para comunicar las acciones de bloqueo y desbloqueo de puertas.



(a) Esquema del control de puertas de un vehículo [NSL17, p.30].



(b) Comunicación de un vehículo con la vía u otros vehículos [BJ19, Figure 8].

Figura 2.5

- **Dominio multimedia, telemático y de interfaz hombre-máquina:** Referido a los sistemas multimedia del vehículo, los sistemas que permiten al conductor conocer el estado de algunos sistemas del vehículo a través del cuadro de mandos y otros sistemas como el sistema de pagos automáticos en telepeajes. En este dominio es donde se centran algunas de las claves de las conocidas como Smart Cities, la comunicación del vehículo con las calles, carreteras y otros vehículos, como se muestra en la figura 2.5(b), se llevarían a cabo por sistemas que se engloban en este dominio.

- **Dominio de seguridad activa y pasiva:** Referido a sistemas de seguridad pasiva como los cinturones de seguridad o los airbags y a sistemas de seguridad activa como pueden ser los sistemas de control de crucero adaptativos, los sistemas de prevención y aviso de colisión o los sistemas ayuda para evitar las salidas del carril de circulación. Sistemas como el airbag se basan en valores recogidos por otros sensores pertenecientes a dominios como el del tren automotriz o el del chasis para accionarse en fracciones de segundo después de que el vehículo sufra una colisión. En la figura 2.6 se ilustran algunos de los sensores utilizados por sistemas de seguridad activa.

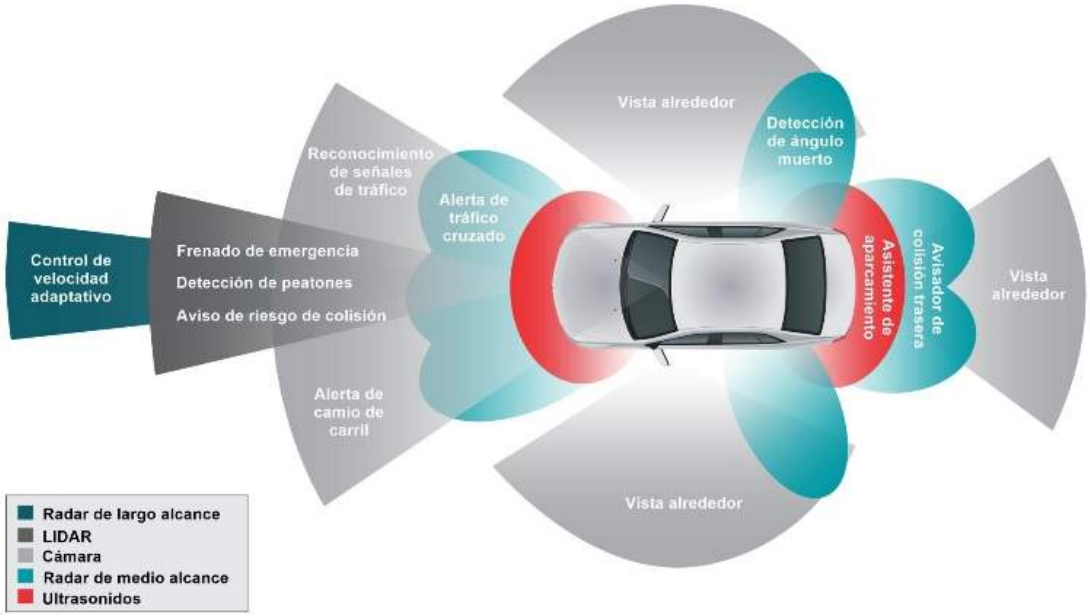


Figura 2.6: Sensores utilizados por sistemas de seguridad activa. Fuente: https://www.top10motor.com.

- **Dominio de diagnóstico:** Referido a todos los sistemas involucrados en el diagnóstico de emisiones y averías del vehículo. Un ejemplo claro de sistema que se engloba en este dominio es el sistema OBD, que permite hacer diagnósticos de emisiones y averías en tiempo real y que, además, ofrece una memoria en la que se almacenan códigos de error conocidos como Diagnostic Trouble Codes (DTCs, en adelante) que ayudan a indicar averías registradas por el sistema OBD.

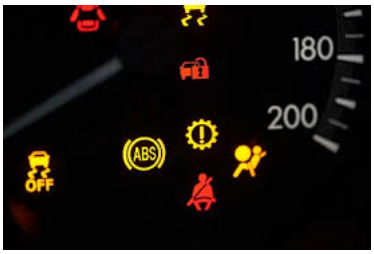


Figura 2.7: Indicadores de avería disponibles en algunos vehículos. Fuente: https://industryreports24.com

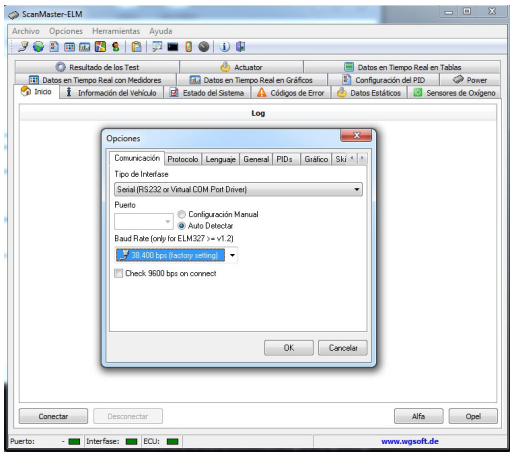
Todos estos dominios se conectan y comunican entre sí por medio de redes CAN de alta y baja velocidad, según los requerimientos de tiempo y ancho de banda de cada dominio o de cada sistema específico.

2.3. Software de diagnóstico OBD

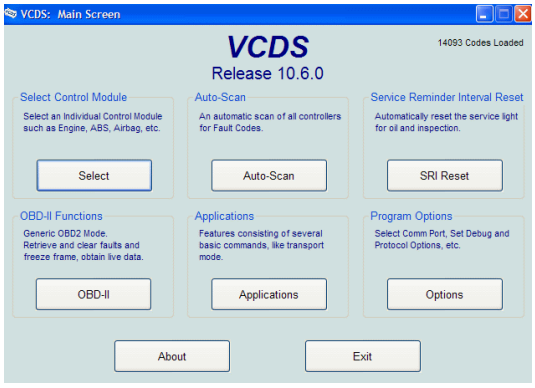
En la actualidad, existen multitud de formas de comunicarse con el sistema de diagnóstico de un vehículo a través de un adaptador OBD y una conexión USB, WiFi, Bluetooth o de cualquier otro tipo, dependiendo del adaptador utilizado y del dispositivo utilizado para la recogida de datos del OBD. Las opciones más relevantes en cuanto a software de lectura de datos OBD están disponibles en la web de ELM Electronics.

Algunas de esas opciones son:

- **ScanMaster, para sistemas Windows**, desarrollado por la compañía desarrolladora alemana WGSoft. Permite la conexión mediante un adaptador OBD con conexión serie RS232-USB o por medio de Bluetooth y es capaz de ejecutar todo tipo de funciones contempladas en el sistema OBD.
- **VCDS, para sistemas Windows**, más conocida como VAG-COM. Es la aplicación específica del grupo Volkswagen y permite utilizar el protocolo propietario de Volkswagen para comunicarse con el sistema OBD de todos los vehículos de las marcas pertenecientes al grupo.



(a) Captura de la pantalla de inicio de la aplicación ScanMaster.



(b) Captura de la pantalla principal de la aplicación VCDS.

Figura 2.8

- **Torque, para sistemas Android**. Permite la conexión por medio de Bluetooth con un adaptador OBD con módulo Bluetooth y es capaz de integrar plugins desarrollados por una comunidad de usuarios de la aplicación, de manera que existen, incluso, módulos que permiten grabar video con la cámara del dispositivo Android y superponer en el video las mediciones recogidas en tiempo real por el sistema OBD.

Figura 2.9: Captura del cuadro de mandos mostrado por la aplicación Torque.

A la vista de las aplicaciones revisadas en este TFG, se puede observar como todas cumplen con unos mínimos de funcionalidad en cuanto al sistema OBD, pero ninguna luce una interfaz sencilla y llamativa que además permita al usuario entender el estado de cada uno de los parámetros mostrados de un solo vistazo, como sería deseable, además de ofrecer un modo de análisis de los datos recogidos.

En este TFG se ha diseñado una interfaz gráfica sencilla e intuitiva que permite al usuario conocer el estado del vehículo de un solo vistazo al cuadro de mandos mostrado y analizar el funcionamiento del mismo estudiando los datos recogidos, resultando en la creación de una aplicación para Android cuyo nombre es OBDII Dashboard.

Capítulo 3

Tecnologías empleadas

En el desarrollo de este TFG se han utilizado una serie de tecnologías que han sido clave para la construcción de la aplicación resultante.

En este capítulo se hace una enumeración de las tecnologías utilizadas, describiendo las necesidades que cubren y repasando su utilidad en cada caso. La información aquí presente se puede consultar en las siguientes referencias [Bus; Esd; Obd; Elm; Blu; Gps; Anda; Andc; Ora].

3.1. El protocolo CAN bus

CAN bus es un protocolo bus serie orientado a buses de campo que se define en el conjunto de estándares ISO 11898. Este conjunto de estándares define las capas física y de enlace del protocolo en sus distintos tipos. Atendiendo a este conjunto de estándares, existen dos tipos de buses, CAN bus de alta velocidad (hasta $1\ \mathrm{Mbit/s}$) y CAN bus de baja velocidad tolerante a fallos (hasta $125\ \mathrm{kbit/s}$).

Se describirá el funcionamiento de las capas física y de enlace del modelo OSI (Open System Interconnection), profundizando en esta segunda capa y estudiando los distintos tipos de trama contemplados en el protocolo CAN bus.

En primer lugar, la capa física define la forma en que se transmite la información entre los nodos de la red. La conexión entre los dispositivos se establece por medio de dos cables (que son los que conforman el bus) por los que se transmiten dos señales, CAN_H y CAN_L, respectivamente. Los cables son de par de cobre trenzado, que protege al bus de posibles interferencias electromagnéticas provenientes de sistemas externos. Además pueden ser apantallados o no.

Cuando los dos cables están sujetos a la misma tensión, se dice que el bus

está en estado recesivo. Contrariamente, si hay una diferencia de tensión entre los dos cables, se dice que el bus está en estado dominante. Esta diferencia de tensión entre las señales CAN_H y CAN_L debe ser de al menos $1,5$ V.

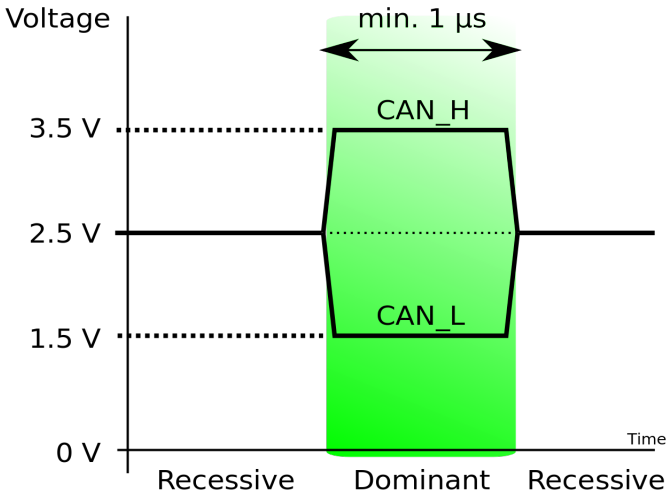


Figura 3.1: Estados de la red según la tensión [Bus].

En cuanto a la longitud del bus y la tasa de transferencia, existen limitaciones que dan lugar a dos tipos de buses:

- Buses de hasta 40 metros de longitud, cuya tasa de transferencia llega hasta 1 Mbit/s.
- Buses de hasta 1 kilómetro con una tasa de transferencia de hasta 50 kbit/s en las configuraciones de mayor longitud y, típicamente, con una tasa de transferencia de hasta 125 kbit/s en configuraciones medias.

Cada nodo de la red requiere varios elementos para poder comunicarse con el resto de nodos:

- Un controlador CAN, que se encarga de recibir y transmitir mensajes, guardando los bits que se transmiten en serie por el bus o escribiéndolos del mismo modo en este.
- Un procesador que procesa los mensajes recibidos por el controlador CAN y decide qué mensaje enviar a través de este si fuera necesario.
- Un transceptor, encargado de adaptar la señal del bus a los niveles aceptados por el nodo y viceversa.

A nivel de la capa de enlace, existen cuatro tipos de trama que se pasará a describir en el resto de la sección: la trama de datos, la trama remota, la trama de error y la trama de sobrecarga.

La trama de datos tiene dos formatos posibles:

- El formato base, que cuenta con 11 bits dedicados a la identificación de la trama.
- El formato extendido, que cuenta con 29 bits para la identificación de la trama.

En ambos formatos, los bits de identificación se utilizan como representación de la prioridad de la trama dentro del bus. Por medio de estos bits, y aplicando una función lógica determinista como es la función AND, se pueden detectar colisiones en la red y se pueden resolver por supervivencia de la trama con identificador de mayor prioridad. Para ello, hay que tener en cuenta que el estado del bus que se mencionó anteriormente (dominante o recesivo) define el valor de bit. Es decir, cuando la red está en estado recesivo, se considera un bit con valor 1 y, cuando la red está en estado dominante, se considera un bit con valor 0. Como su propio nombre indica, el valor predominante, a igualdad de posición dentro del identificador de una trama y a efectos de definición de prioridad, es el valor de bit dominante, el 0. Esto es, una trama con un identificador con valor binario más bajo (más ceros a la izquierda), es más prioritaria que una con valor binario más alto.

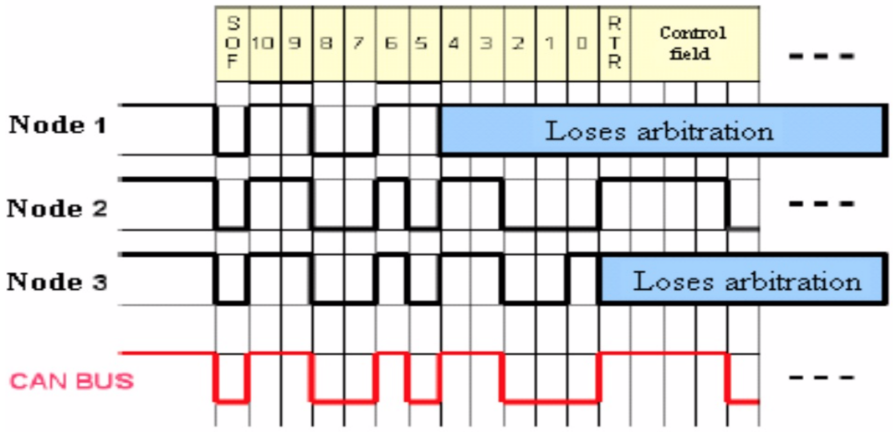


Figura 3.2: Arbitraje en CAN bus, en este caso la señal del Nodo 2 es la que prevalece. Fuente: canbus.pl.

De este modo, se determinan las prioridades para cada mensaje, de forma que todos los nodos de la red pueden transmitir y recibir mensajes al mismo tiempo, contando con un arbitraje. Este arbitraje es, a grandes rasgos, el definido en el párrafo anterior, los nodos se pondrán de acuerdo para transmitir mensajes según la prioridad. Al final del arbitraje solo quedará un nodo transmitiendo su trama en la red: el que transmita el mensaje con mayor prioridad.

Se observa entonces que el protocolo CAN utiliza un control de acceso al medio compartido del tipo CSMA/CD+CR, es decir, acceso múltiple con detección de portadora, detección de colisión y resolución de colisión.

Teniendo en cuenta la forma en que se priorizan los mensajes dentro de la red, se puede determinar la latencia en la transmisión de mensajes entre los nodos de la red y se puede prescindir, a nivel del modelo OSI, de procedimientos para el control de acceso al medio.

En cuanto al resto de campos del formato base, se distinguen:

- El campo de *petición de transmisión remota* o RTR por sus siglas en inglés, con longitud de un bit y dedicado a indicar si la trama es de datos o es una trama remota.
- El campo de *bit de extensión de identificador*, dedicado a indicar si el formato es base o extendido.
- El campo de *bit reservado*, que tiene longitud de 1 bit.
- El campo de *código de longitud de datos* o DLC por sus siglas en inglés, encargado de indicar el tamaño del campo de datos en bytes, con un valor máximo de 8.
- El propio *campo de datos*, que tiene la longitud indicada en el campo DLC.
- El campo CRC, que contiene un *código de redundancia cíclica* que permite verificar que el mensaje se ha transmitido correctamente.
- El campo *delimitador* del CRC, que tendrá valor recesivo para indicar el final del código.
- El campo de *acuse de recibo* o ACK por la abreviación del término en inglés, que será recesivo para el transmisor de la trama y dominante para los receptores.

- El campo *delimitador del acuse de recibo*, que debe ser recesivo.
- El campo de *final de trama*, que deberá ser recesivo también.

Puede darse el caso de que, en la red, un nodo requiera datos de otro nodo de la red. Para estos casos, existe otro tipo de trama en la que el campo de petición de transmisión remota (RTR) está informado con valor 1, recesivo. Este tipo de trama se conoce como trama remota y se diferencia de una trama de datos fundamentalmente en que la primera tiene el campo RTR con valor recesivo y no tiene campo de datos, es decir, tras el campo DLC, tiene el campo CRC.

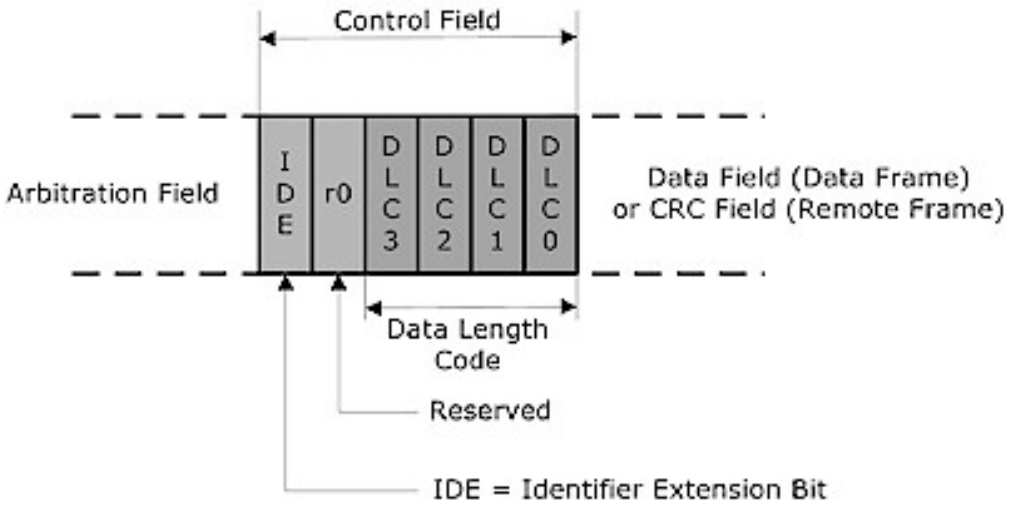


Figura 3.3: Trama de datos frente a trama remota. Fuente: esd-electronics.

Si algún nodo de la red tiene mucha carga en un momento dado, éste puede enviar a la red una trama de sobrecarga (véase la figura 3.4), que no tiene la estructura de una trama de datos ni de una trama remota, y que sirve para aumentar el retardo entre envío de tramas dentro del bus. Este tipo de tramas se identifican por estar compuestas por dos campos: un flag de 6 bits dominantes que se envían en el intervalo que hay entre dos tramas y un delimitador de 8 bits recesivos. Cuando el resto de nodos de la red detectan una trama de sobrecarga, responden con flags de sobrecarga, parando así el tráfico de la red. La longitud del campo flag de sobrecarga puede ser de 6 a 12 bits.

Cuando se transmite una trama que no es ni de datos, ni de sobrecarga, ni remota, se considera que la trama es una trama de error. Una trama de error, como la trama de sobrecarga, está formada por dos campos: un flag de error,

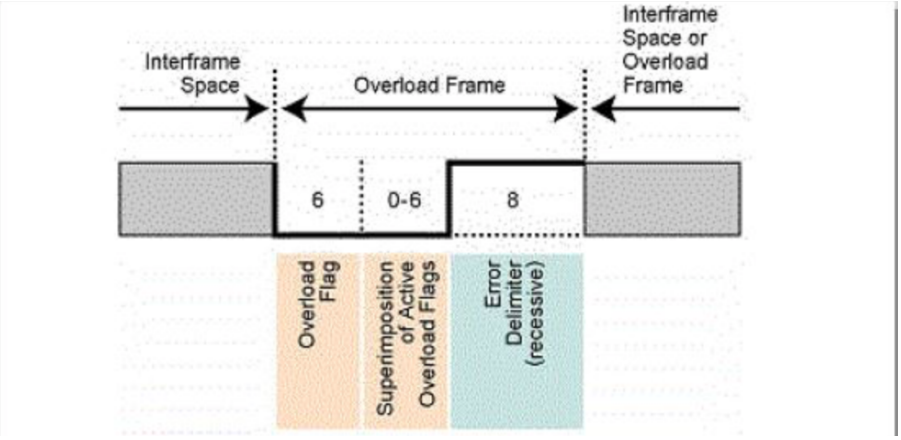


Figura 3.4: Trama de sobrecarga. Fuente: canauto.wordpress.com.

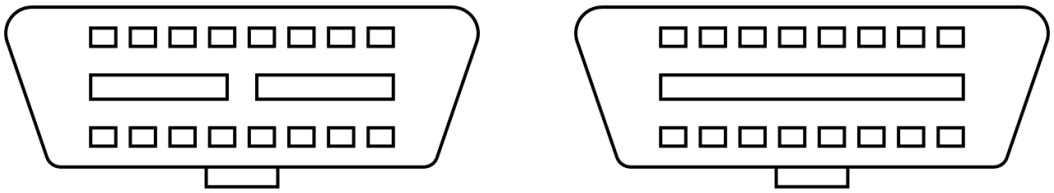
que puede ser activo si los 6 bits que lo componen son dominantes, o pasivo si los 6 bits que lo componen son recesivos y un delimitador de error, que se compone de 8 bits recesivos. Si un nodo de la red detecta un mensaje erróneo, éste provoca que los demás nodos también transmitan tramas de error.

Tanto las tramas de datos como las tramas remotas están separadas por, al menos, tres bits recesivos, de forma que cuando se detecta un bit dominante, se considera que es el inicio de una nueva trama. Las tramas de error no respetan la separación entre tramas y las tramas de sobrecarga solo se envían en el espacio entre tramas, es decir, tras los tres bits recesivos.

3.2. La interfaz OBD II

Una forma sencilla de comunicarse a través de una red CAN bus en un vehículo es a través del puerto de diagnóstico a bordo u OBD, por sus siglas en inglés. La mayoría de vehículos europeos actuales disponen de un puerto OBD que implementa el estándar OBD II.

El estándar OBD II surge como sistema dedicado al control de emisiones y diagnóstico de fallos en un vehículo. OBD II se especifica en la norma ISO 15031. Esta norma describe dos tipos de conectores con la misma forma en D y la misma disposición de pines. Estos dos tipos son el tipo A, en el que la alimentación del conector funciona a 12 voltios y 4 amperios y el tipo B, en el que la alimentación del conector funciona a 24 voltios y 2 amperios. Salvando esta diferencia y una pequeña ranura en el centro del conector de tipo B, ambos tipos tienen la misma



(a) Conector OBD II tipo A (b) Conector OBD II tipo B

Figura 3.5: Fuente: Wikipedia.

forma y la disposición de pines de ambos tipos es la misma. El tipo A es el que se utiliza normalmente en los vehículos tipo turismo. Los pines 6 y 14 del puerto OBD II se corresponden con las señales CAN_H y CAN_L, respectivamente.

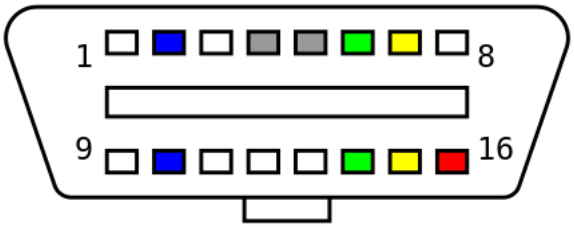


Figura 3.6: Pines del conector OBD tipo A. En verde CAN_H y CAN_L. Fuente: Wikipedia.

Además de los tipos de conector y la disposición de pines, el estándar OBD II también especifica una lista de códigos de error estandarizada con la que se pueden detectar fallos en el vehículo, una lista de comandos para consultar datos en vivo a las unidades de control del vehículo y varios modos para consultar un histórico de errores, borrar los errores que se muestran actualmente, etc.

En el mercado existen multitud de dispositivos, conocidos como adaptadores OBD, que permiten utilizar el estándar OBD II junto con el protocolo CAN bus para comunicarse con un vehículo a través del conector OBD. Entre los adaptadores OBD más económicos, están los que utilizan versiones clonadas del micro controlador ELM327. El micro controlador ELM327, cuyo diagrama de bloques se puede ver en la figura 3.8, es uno de los micro controladores más populares de los utilizados para la comunicación entre un ordenador y el sistema de diagnóstico a bordo de un vehículo. Este micro controlador pretende ser una interfaz de conexión entre el sistema OBD de los vehículos e interfaz de conexión en serie estándar RS232. Ofrece detección automática del protocolo de comunicación utilizado por el sistema OBD al que se conecta y además es capaz de comunicarse

utilizando 9 protocolos diferentes, como se puede ver en la figura 3.7, entre otras características interesantes. La configuración se realiza mediante comandos AT (o conjunto de comandos Hayes), que se enviarán por medio de una terminal o cualquier otro medio que permita conectar por medio del puerto serie RS232.

Protocol	Description
0	Automatic
1	SAE J1850 PWM (41.6 kbaud)
2	SAE J1850 VPW (10.4 kbaud)
3	ISO 9141-2 (5 baud init)
4	ISO 14230-4 KWP (5 baud init)
5	ISO 14230-4 KWP (fast init)
6	ISO 15765-4 CAN (11 bit ID, 500 kbaud)
7	ISO 15765-4 CAN (29 bit ID, 500 kbaud)
8	ISO 15765-4 CAN (11 bit ID, 250 kbaud)
9	ISO 15765-4 CAN (29 bit ID, 250 kbaud)
A	SAE J1939 CAN (29 bit ID, 250* kbaud)
B	User1 CAN (11* bit ID, 125* kbaud)
C	User2 CAN (11* bit ID, 50* kbaud)

*user adjustable

Figura 3.7: Protocolos de comunicación que se pueden utilizar con el ELM327. Fuente: www.elmelectronics.com.

Para manejar las conexiones en el puerto serie, el micro controlador ELM327 utiliza un controlador UART (Universal Asynchronous Receiver-Transmitter), que se encarga de recoger los datos recibidos y transmitirlos por la interfaz de conexión en serie RS232.

Los adaptadores OBD que utilizan estas versiones clonadas del micro controlador ELM 327 suelen declarar en sus especificaciones el uso de versiones más actualizadas de este micro controlador, pero realmente utilizan la versión 1.0, lo cual disminuye el número de funciones soportadas. Estos adaptadores OBD tienen la ventaja de equipar, en muchos casos, un módulo Bluetooth que permite la

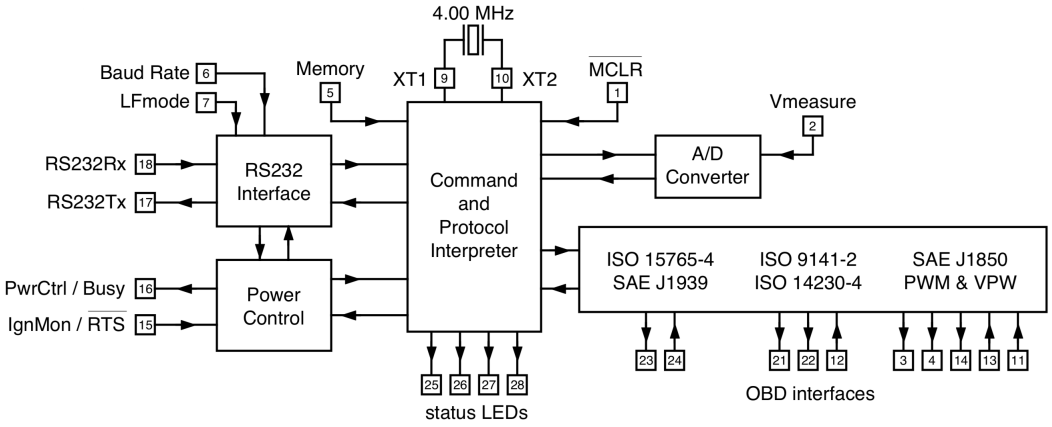


Figura 3.8: Diagrama de bloques del ELM327. Fuente: www.elmelectronics.com.

conexión inalámbrica con ordenadores o smartphones. Este módulo Bluetooth, que habilita la conexión serie con el adaptador OBD, se conecta al controlador UART y actúa como si fuese una interfaz RS232.

La forma en que se obtienen los datos del vehículo en OBD II es a través de los PIDs, Parameter IDs en inglés. Estos PIDs son códigos que se utilizan para solicitar datos al vehículo, de forma que existe un estándar que define muchos de estos códigos, el SAE J1979. Además de los códigos definidos por el estándar, cada fabricante puede definir una serie de PIDs específicos para sus vehículos.

A continuación se describe el proceso de solicitud y respuesta de datos al vehículo a través de OBD II y se muestra un ejemplo descriptivo.

La forma en que un adaptador OBD solicita a un vehículo un dato concreto es a través de CAN bus, enviando solicitudes OBD con una ID de difusión $7\mathrm{DF}_{16}$ (el subíndice tras el ID indica que esta está codificada en hexadecimal, en adelante se indicará cualquier valor hexadecimal siguiendo la misma notación) que llega a todas las ECUs que estén conectadas a la red CAN. A la ID del mensaje le sigue un campo de un byte que indica el número de bytes útiles restantes en el mensaje enviado y posteriormente un campo de un byte que indica el servicio OBD que se quiere utilizar y de uno a cinco bytes con el PID que se solicita.

La respuesta es un mensaje con ID en el rango de $7\mathrm{E8}_{16}$ a $7\mathrm{EF}_{16}$, dependiendo del ID de la ECU que responda. Las ECU tendrán un ID dentro del rango $7\mathrm{E0}_{16}$ a $7\mathrm{E7}_{16}$ y su respuesta tendrá el ID correspondiente sumando el valor 8_{16} a esta ID.

El campo de un byte dedicado a indicar el número de bytes útiles del resto

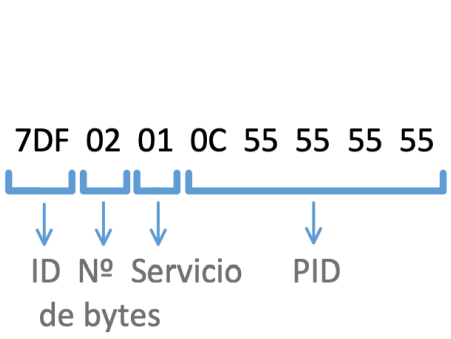
del mensaje sigue funcionando igual en el mensaje de respuesta. En cuanto al campo que indica el servicio OBD utilizado, en este caso es igual que el de la solicitud, pero sumando el valor 40_{16}.

El resto del mensaje estará determinado por el PID de respuesta correspondiente. La fórmula necesaria para obtener el valor final y la descripción de cada PID se puede consultar la tabla definida por el estándar SAE J1979.

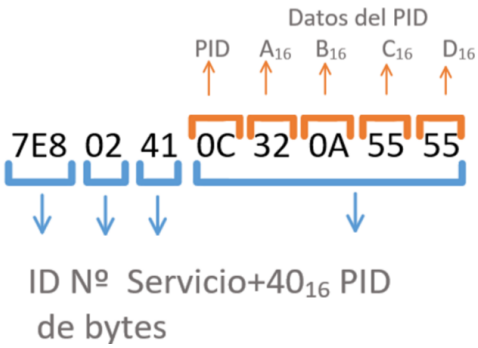
En las figuras 3.9(a) 3.9(b) se muestra un ejemplo descriptivo de una solicitud de datos sobre las revoluciones por minuto del motor a través de OBD II y la respuesta de la ECU. En el supuesto de que, en el momento de la petición, el número R de revoluciones por minuto del motor sea $3202,5$ RPM, los mensajes de petición y respuesta serán los mostrados a continuación. La fórmula para el cálculo del número de revoluciones es

$$R = \frac{256p(y,5)_{10} + p(y,6)_{10}}{4},$$

donde $p(y,i)$ representa el valor i-ésimo de un mensaje de respuesta y a un mensaje de solicitud OBD, x (véase la figura 3.9(b)).



(a) Mensaje de petición. Se solicita información en tiempo real sobre el número de revoluciones por minuto.



(b) Mensaje de respuesta. Aquí, $p(y,5) = A$, $p(y,6) = B$, $p(y,7) = C$ y $p(y,8) = D$.

Figura 3.9: Los valores de los bytes están expresados en base hexadecimal.

En el mensaje de petición se observa cómo se especifica el servicio 01_{16} para obtener datos en tiempo real y cómo se solicita información sobre las revoluciones por minuto del motor utilizando el PID $0C_{16}$, según la tabla de PID especificada por el estándar [Pid].

En la respuesta se observa cómo al byte dedicado a indicar el servicio se le suma el valor 40_{16} y cómo a este le siguen los bytes que indican el PID destinado a obtener información sobre las revoluciones por minuto del motor, $0C_{16}$, y el valor específico en tiempo real en los bytes de datos del PID, $p(x, 5)$ y $p(x, 6)$, con valores 32_{16} y $0A_{16}$, respectivamente.

Haciendo uso de la fórmula especificada en el estándar, el número R de revoluciones por minuto del motor se obtiene del siguiente modo, utilizando los valores decimales:

$$R = \frac{(256 \cdot 50) + 10}{4} = 3202,5.$$

La tabla con los PID especificados por el estándar se puede encontrar en la referencia [Pid].

3.3. La conexión Bluetooth

Bluetooth es una tecnología de conexión inalámbrica dedicada al intercambio de datos entre dispositivos separados por distancias cortas utilizando señales de longitud de onda corta, en el rango de los $2{,}4\text{GHz}$. Fue estandarizado por el Institute of Electrical and Electronics Engineers (IEEE) como IEEE 802.15.1, pero es el Bluetooth Special Interest Group el encargado del desarrollo y mantenimiento de las nuevas especificaciones de Bluetooth. Surgió como una alternativa inalámbrica a las conexiones serie RS232. Actualmente es ampliamente utilizado en dispositivos periféricos y multimedia para conectar con un PC o un smartphone de forma inalámbrica. Cuando varios dispositivos se conectan entre sí utilizando la conexión Bluetooth, se crean redes conocidas como Redes de Área Personal. Este tipo de redes está muy extendido en la actualidad. Esto permite que sea fácil y poco costoso encontrar dispositivos que equipen un módulo Bluetooth para dotar de funcionalidades inalámbricas de corto alcance a dichos dispositivos.

Los smartphones forman parte de este grupo de dispositivos que incorporan módulos Bluetooth para realizar conexiones y transferencias de datos poco pesadas en entornos de poca distancia. En el desarrollo de la aplicación OBDII Dashboard se ha empleado un dispositivo Android dotado de conectividad Bluetooth.

En cuanto a la arquitectura de cualquier controlador Bluetooth, se debe describir atendiendo a dos ámbitos, el del software y el del hardware:

- En cuanto al **software**, los dispositivos que intervienen en una conexión Bluetooth se comunican mediante una interfaz llamada Host Controller Interface (HCI), que se encarga de ofrecer una serie de comandos para que los dispositivos puedan conocer el estado y acceder a los registros hardware para poder enviar y recibir datos correctamente. A niveles más altos, los dispositivos utilizan protocolos como el Service Discovery Protocol (SDP), encargado de la búsqueda de dispositivos dentro del rango de la conexión Bluetooth, el Radio Frequency Communication (RFCOMM), que es el encargado de emular las conexiones al puerto serie o el Telephony Control Protocol (TCS, no confundir con Traction Control System), encargado de los controles de telefonía. Estos protocolos interactúan con el protocolo L2CAP (Logical Link Control and Adaptation Protocol), que es el responsable de la lectura de los paquetes que se envían por la red Bluetooth.
- En cuanto al **hardware**, se tienen dos piezas principales que son:
  - La radio, que es responsable de modular y transmitir las señales analógicas para conectar con otros dispositivos.
  - El controlador digital, que se encarga de ejecutar un Link Controller (o controlador de enlace). Este es el responsable del procesamiento, por ejemplo, del audio en las conexiones con sistemas multimedia.

Se puede subir en capas de abstracción hasta al menos dos niveles más, esto da lugar a la pila de protocolos Bluetooth representada en la figura 3.10

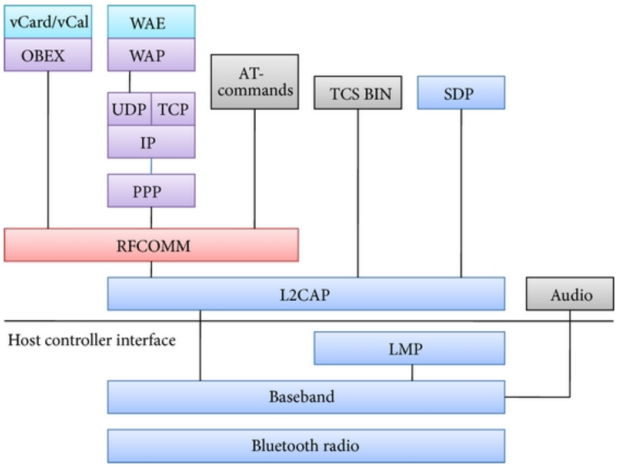


Figura 3.10: Pila de protocolos Bluetooth.

Desde su lanzamiento por el Bluetooth Special Interest Group en 1998, se han ido revisando y mejorando las características hasta llegar a la actual versión 5.1:

- **Bluetooth 1.0 y 1.0B:** Son las primeras versiones de Bluetooth. Estas versiones sufrían bastantes problemas a la hora de operar con otros dispositivos y tenían problemas de seguridad a la hora de establecer la conexión.
- **Bluetooth 1.1:** En esta versión, se resuelven muchos errores encontrados en la versión 1.0B y, además, se añade la posibilidad de establecer conexiones sin cifrar. Se añade también un indicador de la potencia de la señal.
- **Bluetooth 1.2:** Se mejora la velocidad de conexión y de detección de otros dispositivos cercanos, se mejora la resistencia a interferencias por medio de técnicas de salto de frecuencias evitando usar frecuencias con mucho tráfico y se aumenta la velocidad de transmisión hasta los $721\mathrm{kbit/s}$, entre otras mejoras.
- **Bluetooth 2.0:** Lanzada en 2005. La mejora principal de esta versión es la inclusión de Enhanced Data Rate (EDR) para una mayor velocidad de transferencia de datos. Sin contar las mejoras aportadas por EDR, las mejoras en Bluetooth 2.0 con respecto a las versiones anteriores son mínimas.
- **Bluetooth 2.1:** Esta versión, lanzada en 2007, mejora la seguridad en el momento del emparejamiento y también introduce mejoras en los filtros en el momento de la detección de dispositivos.
- **Bluetooth 3.0:** Aumenta la velocidad de transferencia utilizando como medio de transferencia de datos una red WiFi 802.11, de forma que mantiene la conexión con otros dispositivos mediante Bluetooth y pasa a utilizar la red WiFi 802.11 para transferir los datos.
- **Bluetooth 4.0:** Con el nombre de Bluetooth Smart, esta versión fue lanzada en 2010 incluyendo los protocolos de Bluetooth clásico, Bluetooth de alta velocidad y Bluetooth de baja energía. Hasta el año 2013 se fueron publicando revisiones del protocolo que mejoraban el consumo de los dispositivos.
- **Bluetooth 4.1:** Es una mejora solamente en la parte del software. Entre algunas nuevas funcionalidades añadidas, se encuentra la capacidad de permitir a un dispositivo tener varios roles simultáneamente.

- **Bluetooth 4.2:** Introduce cambios orientados al Internet of Things (IoT), mejorando en áreas como la seguridad en conexiones de baja energía y en el proceso de enlace con otros dispositivos.
- **Bluetooth 5.0:** A mediados de 2016 se lanza esta versión que incluye características mayormente orientadas al IoT y a mejorar las velocidades de transferencia en el modo de baja energía.
- **Bluetooth 5.1:** En enero de 2019 se presenta esta versión que incluye características utilizadas para la localización y seguimiento de dispositivos. Estas características abren un extenso abanico de posibilidades para los dispositivos IoT.

3.4. El sistema GPS

Desarrollado por el Departamento de Defensa de los EE. UU., el sistema GPS (por Global Positioning System) permite determinar la posición 3D de cualquier objeto en el planeta Tierra. La precisión del sistema puede llegar hasta los centímetros, aunque normalmente los dispositivos que utilizan GPS utilizan variantes con una precisión algo menor, normalmente de unos pocos metros.

Su funcionamiento se basa en la trilateración, método que consiste en determinar la posición relativa de un objeto utilizando la geometría de los triángulos. A diferencia de la triangulación, la trilateración se ayuda de, al menos, dos localizaciones conocidas y la distancia entre ellas y el sujeto a posicionar.

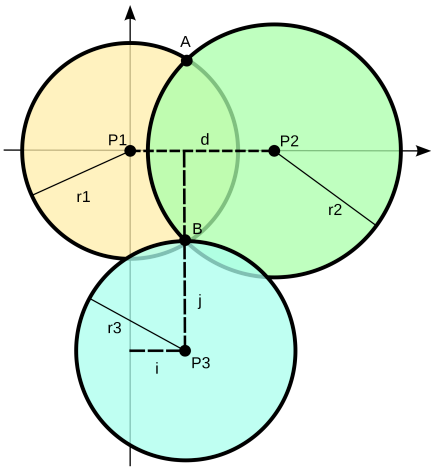


Figura 3.11: Trilateración. Fuente: Wikipedia

En el sistema GPS, las localizaciones conocidas son las de redes de satélites situados de 20 a 180 kilómetros de altura, con trayectorias sincronizadas para cubrir siempre toda la superficie del planeta. Para devolver la posición GPS, un dispositivo receptor recoge los datos y muestra la posición dada en 3 variables: latitud, longitud y altitud. Con estos tres valores se puede situar con exactitud cualquier objeto en el planeta, aunque la precisión dependerá del número de satélites utilizados, los sistemas de posicionamiento consultados y el post-procesado que se haga con ellos.

Existen 4 sistemas de posicionamiento global populares que pueden utilizarse para posicionar objetos sobre el planeta:

- **Navstar GPS**: Es el sistema GPS de los Estados Unidos. Formado por una constelación de 24 satélites, es el más popular de todos.

- **GLONASS**: Este sistema, desarrollado por la Union Soviética y administrado hoy en día por la federación Rusa, surgió como contrapartida al GPS norteamericano y hoy en día se usa junto con este para aportar precisión extra en el geoposiciconamiento en sistemas comerciales.

- **Galileo**: Es el sistema de posicionamiento global europeo, desarrollado por la Agencia Espacial Europea conjuntamente con la Unión Europea. Su lanzamiento fue en diciembre de 2016 y desde entonces hasta el año 2020 se espera que la constelación de satélites esté completa en órbita. Se puede utilizar conjuntamente con el sistema Navstar GPS para mejorar la fiabilidad.

- **Beidou**: En su primera versión, Beidou era un sistema de posicionamiento chino que ofrecía cobertura limitada a China y zonas cercanas. En la actualidad, el sistema se encuentra en una transición desde su segunda versión (conocida como COMPASS) a la tercera, en la que se espera que haya un total de 35 satélites en órbita, consiguiendo así una mayor precision.

En la actualidad, son muchos los dispositivos que permiten la conexión a, al menos, dos de estos sistemas, con el fin de aumentar cuanto sea posible la precisión.

3.5. Android

Para esta sección se ha consultado la referencia [Andb]. Basado en el Kernel de Linux, Android es un sistema operativo orientado principalmente a dispositivos con pantalla táctil como son los smartphones o las tablets. Es un sistema operativo de código abierto y, actualmente, es el sistema operativo más utilizado en el mundo con una cuota de mercado del $40,39\,\%$ en agosto de 2019, según el portal de análisis web https://gs.statcounter.com, como se observa en la figura 3.12.

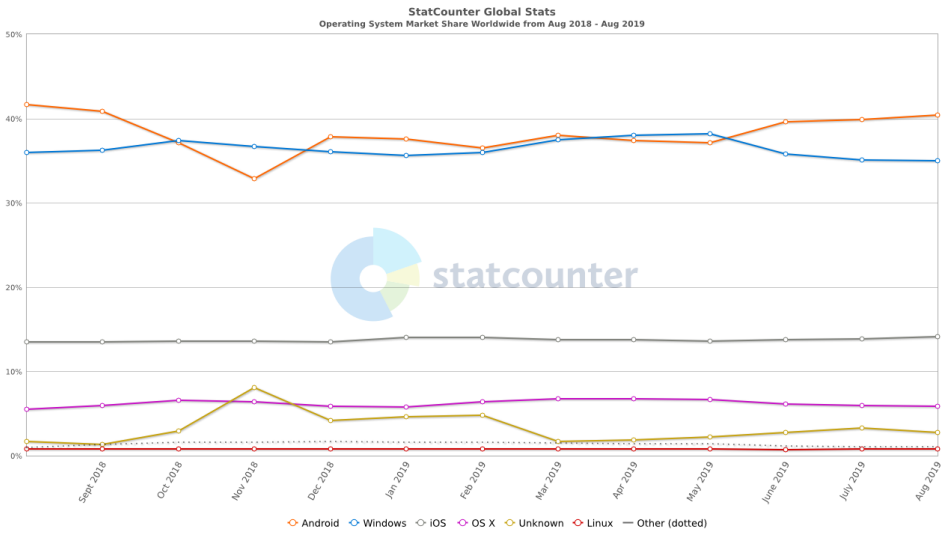


Figura 3.12: Comparación de cuota de mercado de sistemas operativos.

Debido a su gran cuota de mercado, se hace interesante el desarrollo de aplicaciones sobre este sistema operativo, ya que se garantiza la compatibilidad con un mayor número de dispositivos. Por este motivo, en este TFG, se ha decidido el uso de este sistema operativo como base para el desarrollo de la aplicación.

Profundizando un poco en la arquitectura del sistema, se observa una estructura de capas apiladas que se irá describiendo a continuación:

- **Capa de aplicaciones:** Las aplicaciones están escritas en el lenguaje Java y corren sobre un entorno Java. Entre la máquina virtual Java sobre la que corren estas aplicaciones y las mismas aplicaciones hay una capa intermedia conocida como JNI, Java Native Interface, que permite la interoperabilidad entre las aplicaciones y librerías escritas en otros lenguajes como C, C++ o incluso lenguaje ensamblador. Esta característica puede ser muy

beneficiosa para aplicaciones que necesitan de un alto rendimiento como son las aplicaciones de cámara, que aplican filtros en tiempo real a las imágenes recogidas por el sensor. Este tipo de aplicaciones se aprovechan de JNI utilizando rutinas escritas en lenguaje ensamblador o aprovechando el lenguaje C o C++, que es mucho más cercano a la capa hardware.

- **Entorno de trabajo Java:** Es el conjunto de APIs mediante el que los desarrolladores pueden acceder a las características de Android para desarrollar sus aplicaciones. De esta forma se simplifica el uso de componentes y servicios del sistema.

  Algunos de estos componentes y servicios son los siguientes:

  - **Vistas del sistema:** Permite construir interfaces gráficas para las aplicaciones utilizando elementos como listas, vistas de rejilla, cajas de texto, botones, etc.
  - **Gestor de recursos:** Ofrece a los desarrolladores un medio para gestionar recursos como cadenas de texto, imágenes, diseños de la interfaz gráfica y otros gráficos, etc.
  - **Gestor de notificaciones:** Permite a las aplicaciones mostrar al usuario cambios de estado, mensajes recibidos, acciones a ejecutar, indicaciones y muchas más opciones posibles.
  - **Gestor de actividad:** Maneja el ciclo de vida de las aplicaciones dentro del sistema, siendo el componente encargado del inicio y finalización de la ejecución de las aplicaciones. Ofrece una pila de navegación hacia atrás que las aplicaciones pueden utilizar para para gestionar la navegación dentro del conjunto de actividades definidas dentro de las mismas. También es el componente encargado de la navegación entre distintas aplicaciones dentro del sistema.
  - **Gestor de contenidos:** Ofrece a los desarrolladores una manera de gestionar los datos producidos por sus aplicaciones y de acceder a datos producidos por otras aplicaciones.

  Cabe destacar que las aplicaciones desarrolladas para Android pueden codificarse tanto en lenguaje Java como en el lenguaje Kotlin.

- **Librerías C/C++:** La capa de abstracción del hardware y la capa del entorno de ejecución de Android requieren de algunas librerías nativas escritas en C/C++. Esta capa ofrece al sistema estas librerías y permite a los desarrolladores, a través del Android NDK (Android Native Development Kit), el uso de estas librerías para el desarrollo de sus aplicaciones. Algunos ejemplos de aplicaciones que requieren el uso de estas librerías son los reproductores multimedia, los videojuegos, las aplicaciones web o las aplicaciones de procesamiento de audio, video o imágenes. Este tipo de aplicaciones pueden beneficiarse del uso de librerías nativas en C/C++ como Webkit para desarrollos web, OpenMAX AL para procesamiento de audio, video e imagen, Media Framework para el desarrollo de reproductores multimedia u OpenGL ES para el desarrollo de videojuegos y aplicaciones gráficas en 2D y 3D. Para el uso de algunas de estas librerías, el entorno de trabajo Java ofrece al desarrollador APIs que hacen uso de estas librerías nativas, facilitando el trabajo de desarrollo por parte del desarrollador.

- **Entorno de ejecución:** Desde la versión 5 de Android, conocida con el nombre de Lollipop, cada aplicación corre sobre su propia instancia de Android Runtime (ART). ART esta diseñado para permitir la ejecución de múltiples máquinas virtuales en entornos con baja memoria principal disponible, ejecutando ficheros DEX, que es un bytecode diseñado específicamente para Android y que está optimizado para un consumo de memoria mínimo.

  Algunas de las características de ART son:

  - **Compilación just-in-time (JIT) y ahead-of-time (AOT)**: Dada la necesidad de fluidez en el sistema por la alta interactividad de las aplicaciones de hoy en día, se hace interesante poder contar con la opción de compilación de código AOT, que permite compilar una sola vez, en el momento de la instalación de la aplicación, el código bytecode DEX, evitando las esperas y el consumo de recursos que la compilación JIT traen consigo. La compilación JIT es interesante también para los casos en que se requiere la ejecución de un código optimizado conforme las necesidades de la aplicación en cuestión. Este tipo de compilación se hace durante la ejecución de la aplicación, haciendo que la interactividad necesaria en la mayoría de aplicaciones se vea resentida en

términos de tiempo de espera entre ejecución de una rutina y otra. Para aplicaciones menos orientadas al uso interactivo es conveniente utilizar esta opción de compilación frente a AOT. Por estos motivos ART incluye ambas opciones, ofreciendo al desarrollador una gran versatilidad para la compilación de sus aplicaciones.

- **Garbage Collector (GC) optimizado:** Se encarga de la gestión de memoria de las aplicaciones que corren sobre ART, eliminando los objetos que ya no se utilizan y compactando la memoria cuando es necesario, dejando espacio para nuevos objetos en memoria. Para saber más sobre el GC de ART, se puede consultar la referencia [Aos].
- **Soporte para depuración de aplicaciones y herramientas de análisis de rendimiento y gestor detallado de excepciones.**

El entorno de ejecución incluye además un conjunto de librerías clave que ofrecen gran parte de la funcionalidad del lenguaje de programación Java en su última versión, actualmente Java 8. Estas funcionalidades son las que aprovecha el entorno de trabajo Java.

- **Capa de abstracción del hardware:** Es una capa que permite abstraerse del hardware del dispositivo, ofreciendo al entorno de ejecución Java las distintas funcionalidades disponibles en cada dispositivo. Consiste en módulos provistos de múltiples librerías, cada uno de los cuales implementa una interfaz específica para un tipo de hardware concreto, como pueden ser el módulo de GPS o el de Bluetooth. Cuando una funcionalidad de alto nivel requiere el uso de cualquier componente hardware del dispositivo, Android carga el módulo correspondiente a ese componente y ofrece a las capas de nivel superior librerías con funciones específicas para usar dicho componente. De este modo, los desarrolladores no tienen necesidad de preocuparse de la funcionalidad de bajo nivel si no lo necesitan por restricciones específicas de la aplicación que estén desarrollando.

- **Núcleo del sistema:** El núcleo del sistema Android es Linux. Esto implica que características fundamentales del sistema operativo Android, tales como el procesamiento multihilo o el manejo de memoria a bajo nivel recaen sobre el núcleo de Linux. Una ventaja de utilizar Linux como núcleo del sistema es la de facilitar a los fabricantes de dispositivos el desarrollo de drivers, al ser Linux un núcleo ampliamente utilizado y conocido. Otra de

las ventajas es poder aprovechar todas las características de Linux en lo que a seguridad se refiere. El núcleo está diseñado para evitar que el funcionamiento de las aplicaciones pueda causar problemas a otras aplicaciones, al sistema operativo o incluso al dispositivo.

Algunas de las características de seguridad de Linux que el sistema operativo Android incluye son:

- **Sistema de permisos basado en usuarios:** El modelo de permisos basado en usuarios es una característica de seguridad que se encarga de restringir el acceso a según qué partes del sistema o de los datos para según qué usuarios.
- **Aislamiento de procesos:** Cada proceso se aísla de cualquier otro limitando la comunicación inter-proceso. No se permite que un proceso A acceda a la memoria de otro proceso B, a no ser que se establezca un espacio de memoria compartida o se inicie una comunicación inter-proceso mediante sockets locales o sockets de Internet, que permitirían, estos últimos, la comunicación inter-proceso mediante una conexión a Internet.
- **Aislamiento de recursos:** Android es un sistema operativo multiusuario, por este motivo el núcleo del sistema se encarga de proteger los recursos de los usuarios de forma que sea el usuario quien decide qué recursos pueden ser compartidos con otros usuarios y qué recursos no. En particular, el núcleo Linux se asegura de proteger varios recursos:
  - Un usuario A no puede acceder a los ficheros de un usuario B.
  - Un usuario A no puede utilizar el espacio de memoria de un usuario B.
  - Un usuario A no puede aprovechar recursos CPU de un usuario B.
  - Un usuario A no puede aprovechar dispositivos de un usuario B, como podrían ser el GPS o el Bluetooth.

  Beneficiándose de estas características, Android asigna un identificador de usuario (UID) para cada aplicación que se lanza y la ejecuta en su propio proceso. Mediante esta UID, Android genera lo que se conoce como un sandbox a nivel de núcleo del sistema. Un sandbox –

cuya traducción del inglés es caja de arena, en clara referencia al área en que los niños pueden jugar sin peligros alrededor y sin peligro de causar daños a los alrededores –, es un área de la memoria en la que puede correr una aplicación con todos los recursos necesarios para su ejecución, estando este separado del área de memoria del resto de aplicaciones y partes del sistema. Aprovechando el aislamiento de recursos basado en usuarios y grupos de usuarios, el núcleo del sistema consigue limitar la comunicación entre aplicaciones y el acceso a determinadas partes del sistema operativo. De esta característica se deriva el sistema de privilegios de Android, que previene la ejecución de determinadas funciones o el acceso a determinadas partes del sistema a según qué aplicaciones y según qué usuarios. Este sandbox es posible gracias a lo que se conoce como SELinux, de Security-Enhanced Linux, que es el módulo que proporciona los mecanismos para aplicar políticas de seguridad y de control de accesos dentro del núcleo Linux.

El núcleo del sistema también se encarga de toda la gestión de energía del sistema y del acceso al hardware a través de los ya mencionados drivers que los fabricantes desarrollan para el núcleo Linux.

La figura 3.13 representa la pila de capas que conforman la arquitectura del sistema operativo Android, desde abajo en el núcleo Linux, hacia arriba hasta llegar a la capa de aplicaciones.

En cuanto a seguridad propia del sistema operativo, Android separa el sistema del resto de datos haciendo uso de particiones de la memoria secundaria. En concreto, el núcleo del sistema, así como las librerías del sistema operativo, el entorno de trabajo de las aplicaciones, el entorno de ejecución de Android y las aplicaciones en sí, se almacenan por separado del resto de datos en la partición /system, que es de solo lectura.

Además, Android incluye una verificación en el arranque del sistema operativo que se encarga de garantizar la integridad del software del dispositivo arrancando desde un componente hardware de confianza y verificando cada parte del software criptográficamente antes de ejecutarla hasta llegar finalmente a arrancar el sistema operativo.

Por defecto, en Android solamente el núcleo del sistema y un pequeño conjunto de aplicaciones clave corren con permisos de usuario root. En Linux, el

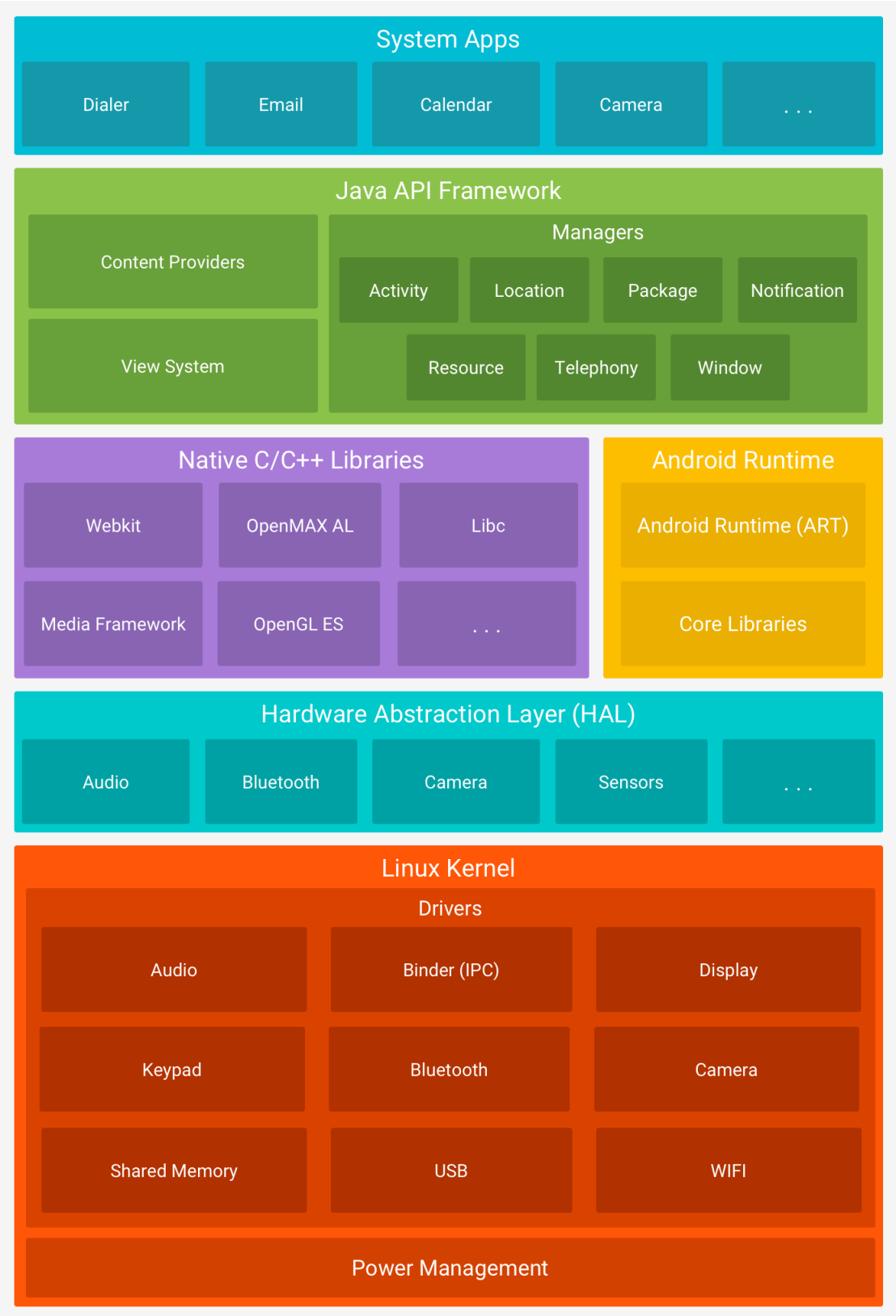


Figura 3.13: Arquitectura de capas de Android. Fuente: https://developer.android.com/guide/platform.

usuario root es el usuario que tiene todos los privilegios del sistema, es decir, puede modificar cualquier parte del sistema sin restricciones. A pesar de esto, Android no restringe a ningún usuario o aplicación con privilegios de usuario root la capacidad de modificar el sistema operativo, el núcleo u otras aplicaciones. Usuarios que se aprovechen de estos privilegios, como pueden ser desarrolladores que necesiten modificar el sistema con cualquier fin, pueden incluso instalar otros sistemas operativos en el dispositivo o estudiar el comportamiento de las aplicaciones sin restricción alguna, teniendo acceso incluso a datos cifrados guardados en el dispositivo, lo que supondría un problema de seguridad importante.

Para evitar el acceso a datos de otros usuarios, los dispositivos Android llevan de fábrica el bootloader, que es la pieza de software que inicia el arranque del sistema operativo, bloqueado. Este bootloader se arranca desde el componente hardware de confianza, que generalmente puede ser modificado únicamente por el fabricante del dispositivo. En ocasiones, los fabricantes permiten del desbloqueo del bootloader, de modo que un usuario puede solicitar al fabricante una clave con la que el hardware de confianza permitirá desbloquear el bootloader, permitiendo hacer cambios en este y, a partir de ese punto, permitir que se cargue un sistema operativo no verificado. Algunos fabricantes, aun permitiendo el desbloqueo del bootloader, añaden más componentes hardware para proteger algunos contenidos específicos, como puede ser el Digital Rights Management (DRM) para los contenidos de video o los datos referentes al NFC (del inglés, Near field communication), usados en aplicaciones de pago a través de NFC por ejemplo. De esta forma, se permite la modificación del sistema operativo pero se restringe el acceso a ciertas funcionalidades si se detecta que se han hecho cambios no autorizados por el fabricante en el software del dispositivo. La forma en que se evita que el desbloqueo del bootloader dé acceso a los datos cifrados de un usuario es ejecutando una función que elimina los datos del usuario. Si el método utilizado para conseguir los privilegios de usuario root no es el ofrecido por el fabricante y, por el contrario, se utilizan agujeros de seguridad del sistema, entonces la función de borrado de los datos de usuario podría saltarse, lo cual llevaría a un escenario en el que un posible usuario no autorizado acceda a los datos de otro usuario dentro del sistema.

En la siguiente sección se repasan las características del entorno de desarrollo oficial de Android utilizadas en el desarrollo de este TFG.

3.6. Android Studio

Android Studio es el Entorno de Desarrollo Integrado (IDE) oficial de Android. Está basado en el editor y las herramientas de desarrollo de Intellij IDEA. Además del editor de código, Android Studio ofrece muchas características que facilitan el desarrollo de proyectos para el sistema operativo Android:

- **Un sistema de construcción de proyectos basado en Gradle:** Se encarga de definir los pasos necesarios para compilar un proyecto y obtener la aplicación final, asegurándose de que se cubren todas las dependencias, etc.
- **Un emulador:** Ayuda al desarrollador a hacer pruebas en un dispositivo emulado con el objetivo de ver el resultado final del desarrollo de la aplicación en los casos en los que no se disponga de un dispositivo físico que ejecute Android.
- **Un entorno unificado de desarrollo:** Permite desarrollar aplicaciones para todos los tipos de dispositivos que ejecutan Android en la actualidad, desde smartphones hasta smartwatchs.
- **Lanzamiento instantáneo:** Conocido como Instant Run, permite hacer modificaciones en el código o en los recursos utilizados por la aplicación y ver los cambios sin necesidad de volver a compilar dicha aplicación.
- **Soporte para C++ y Android NDK:** Facilita la programación de módulos escritos en C++ y la integración de estos en las aplicaciones escritas en Java o Kotlin.
- **Integración con GitHub:** Proporciona integración con el sistema de gestión de versiones GitHub. Esto permite llevar un histórico de cambios en los proyectos y permite compartir el código del proyecto en una red social de desarrolladores mundialmente conocida, con las ventajas en trabajo colaborativo que ello implica.
- **Compleción de código:** Ofrece sugerencias de compleción del código que se escribe en tiempo real, se encarga de ayudar al desarrollador a mantener un estilo de codificación y analiza que la sintaxis utilizada es correcta, señalando errores y sugiriendo posibles soluciones, como se puede ver en la figura 3.14.

- **Editor de diseños:** Ofrece un completo editor de diseños con numerosas ayudas y opciones para el desarrollador. Permite editar los diseños en XML o directamente desde una vista previa, al estilo WYSIWYG. En la figura 3.15 se puede ver un ejemplo de diseño con Android Studio.

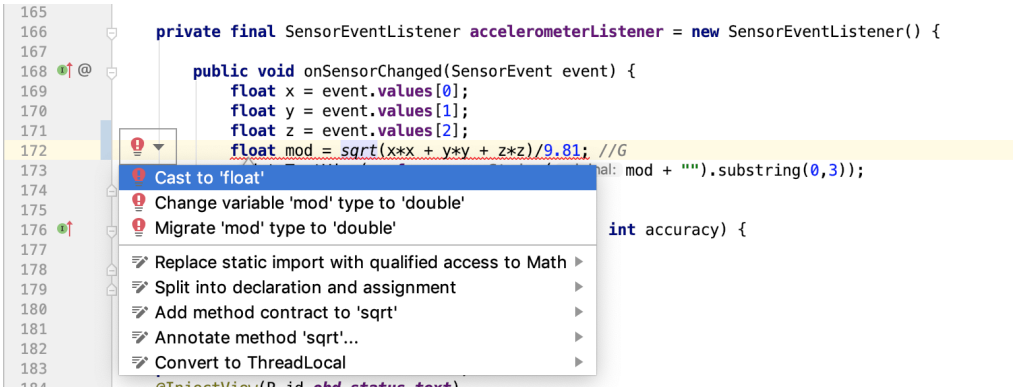


Figura 3.14: Sugerencias y análisis de código en Android Studio.

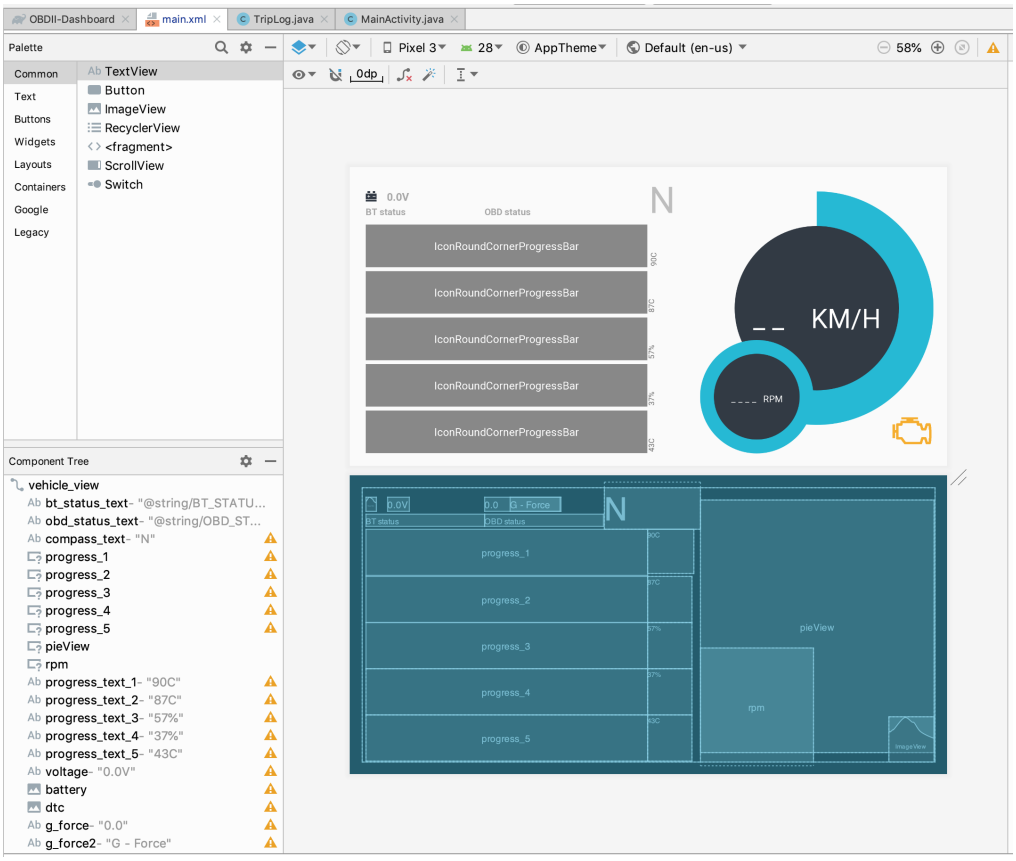


Figura 3.15: Android Studio en la vista Design de un Layout.

Capítulo 4

Análisis y diseño de la aplicación OBDII Dashboard

4.1. Introducción

Existen multitud de aplicaciones de código propietario que son capaces de realizar consultas a través de un adaptador OBD a las ECU de un vehículo. En este TFG se ha desarrollado una aplicación de código abierto con una interfaz gráfica clara y fácil de configurar.

En los capítulos anteriores se ha estudiado la red CAN mediante la que se comunican los distintos sistemas electrónicos dentro de un vehículo y se ha analizado el funcionamiento del sistema de diagnóstico OBD incluido desde hace décadas en todos los automóviles del mercado. Además se ha repasado la arquitectura del sistema operativo Android, explorando el funcionamiento del sistema atendiendo a las diferentes capas que lo conforman y describiendo la herramienta oficial de desarrollo de aplicaciones para dicho sistema operativo.

En este capítulo se describe el desarrollo de la aplicación OBDII Dashboard, analizando las librerías que se han utilizado.

Tras establecer las bases del desarrollo, se describirá el proceso de diseño de la interfaz y la forma en que se relacionan todos los elementos con los datos recogidos por la aplicación para mostrar la información al usuario a través de los elementos gráficos.

4.2. La base de la aplicación

La aplicación OBDII Dashboard, desarrollada en este TFG, está construida sirviéndose de varias librerías. Una de ellas es la librería OBD II Java API, creada por un desarrollador cuyo alias es Pires[Pir13]. Otra es la librería de gráficas HelloCharts for Android, desarrollada por un desarrollador cuyo alias es lecho[Lec]. Por último, dos librerías de elementos gráficos, Android - RoundCornerProgressBar del desarrollador Tailandés akexorcist[Ake15] y Plain-pie desarrollada por Alejandro Zürcher[Zür18] en 2018. Todas las librerías están disponibles en GitHub y han sido publicadas bajo licencias Apache 2.0.

4.2.1. OBD II Java API

La primera librería se dedica a facilitar la comunicación con un adaptador OBD. Está estructurada en 4 partes:

- **Commands:** Esta parte de la librería está orientada a todo lo que está relacionado con los comandos que se envían al sistema OBD, ofreciendo algunas clases abstractas e interfaces que ayudan en el polimorfismo de datos, facilitando la implementación posterior de aplicaciones que envían comandos a un adaptador OBD. Además de estas clases abstractas e interfaces, en este punto se definen las clases que representan a los principales comandos OBD que se pueden enviar para solicitar información a las ECU de un vehículo. Estas clases se clasifican en relación al ámbito en el que se puede englobar el comando concreto al que representan:

  - **Control:** En este grupo se engloban comandos que devuelven información de control referente a averías, a información de identificación del vehículo o a configuración del motor.

    - **DistanceMILOnCommand:** Esta clase representa al comando Distance MIL On, cuyo PID es $01_{16}\ 21_{16}$. Este comando devuelve, en kilómetros, la distancia recorrida desde que la luz indicadora de avería (MIL) se encendió. La distancia d en km se calcula tomando los bytes $p(y, 5)$ y $p(y, 6)$ de la respuesta y para hacer el siguiente cálculo:

$$d = 256p(y, 5)_{10} + p(y, 6)_{10},$$

donde, al igual que en el capítulo anterior, $p(y, i)$ representa el valor del i-ésimo byte del mensaje de respuesta OBD, y.

- **DistanceSinceCCCommand:** Esta clase representa al comando Distance since codes cleared, cuyo PID es 01_{16} 31_{16}. Se calcula del mismo modo que el comando Distance MIL On, y su respuesta indica la distancia, en kilómetros, que se ha recorrido desde que se limpiaron los códigos de avería almacenados en la memoria de la ECU dedicada al diagnóstico.
- **DtcNumberCommand:** Esta clase representa al comando Monitor status since DTCs cleared, es decir, el estado del sistema de diagnóstico desde que los códigos de avería se borraron por última vez. Su PID es el 01_{16} 01_{16}, y devuelve el estado del indicador MIL (que será 1 o 0) seguido del número de códigos de error, $\#\mathrm{DTCs}$, almacenados actualmente en la memoria. El cálculo se hace tomando el byte $p(y, 5)$ de la respuesta y, y operando con cada bit, $p^i(y, 5)$, $i = 0, \ldots, 7$, que lo conforma, del siguiente modo:

$$\text{MIL On} = 1 \iff p^7(y, 5)_2 = 1,$$

$$\#\mathrm{DTCs} = \sum_{i=1}^{7} p^i(y, 5)_2 2^i.$$

- **EquivalentRatioCommand:** Esta clase representa al comando Equivalent Air-Fuel Ratio, cuyo PID es 01_{16} 44_{16}. Esta ratio r se calcula tomando los bytes $p(y, 5)$ y $p(y, 6)$ del mensaje de respuesta y, y haciendo la siguiente operación:

$$r_{\mathrm{EAF}} = \frac{2}{65536}(256p(y, 5)_{10} + p(y, 6)_{10}).$$

  El valor de esta ratio r_{EAF} indica qué proporción de aire se utiliza en relación a la cantidad de combustible utilizado en la combustión empleada por el motor, ya sea por compresión o por chispa.
- **IgnitionMonitorCommand:** Esta clase representa un comando específico del circuito ELM327 que forma parte del adaptador OBD. Devuelve el estado de si el circuito ha entrado en modo de bajo consumo mediante el comando AT LP o no. El comando al que representa la clase es el AT IGN.

- **ModuleVoltageCommand:** Esta clase representa al comando Control Module Voltage, cuyo PID es $01_{16}\ 42_{16}$. Este comando devuelve, en voltios, la tensión V que llega al módulo de control, que es la que la batería del vehículo entrega a todos los dispositivos electrónicos, con lo que permite medir el voltaje de la batería. La tensión se calcula tomando los bytes $p(y, 5)$ y $p(y, 6)$ y realizando el siguiente cálculo:

$$V = \frac{256p(y, 5)_{10} + p(y, 6)_{10}}{1000}.$$

- **PendingTroubleCodesCommand:** Esta clase representa al comando Pending Diagnostic Trouble Codes, cuyo PID es 07_{16}. Este comando es el encargado de recoger los DTCs del actual ciclo de conducción o el anterior. Una lista de DTC se puede encontrar en [KFB14]. La rutina de lectura de códigos parará al recibir el DTC P0000.
- **PermanentTroubleCodesCommand:** Esta clase representa al comando Permanent Diagnostic Trouble Codes, cuyo PID es $0A_{16}$. Este comando es el encargado de recoger los DTC que no están sujetos al borrado mediante las peticiones con comandos OBD. El funcionamiento es exactamente igual al comando Pending Trouble Codes.
- **TimingAdvanceCommand:** Esta clase representa al comando Timing Advance, cuyo PID es $01_{16}\ 0\mathrm{E}_{16}$. Devuelve el ángulo α del cigüeñal en grados en el que se enciende la chispa de la bujía antes de que el pistón llegue al punto muerto superior. Este avance en el encendido es necesario ya que, desde el momento en que la bujía enciende la chispa al momento en que el pistón llega al punto muerto superior, que es el de mayor compresión, pasan aproximadamente unos $2\mathrm{ms}$. El cálculo de los grados que el sistema OBD devuelve se hace tomando el byte $p(y, 5)$ de la respuesta y y operando con ellos del siguiente modo:

$$\alpha = \frac{p(y, 5)_{10}}{2} - 64.$$

- **TroubleCodesCommand:** Esta clase representa al comando Show

Stored Diagnostic Trouble Codes, cuyo PID es 03_{16}. Este comando es el encargado de recoger de la memoria del sistema OBD la lista de códigos de avería que las diferentes ECUs han detectado desde el último borrado. El funcionamiento es exactamente igual al comando Pending Trouble Codes.

- **VinCommand:** Esta clase representa al comando VIN, cuyo PID es $09_{16}\ 02_{16}$. Este comando devuelve una cadena de 17 caracteres en codificación ASCII que contiene el número de identificación del vehículo (VIN).

- **Engine:** En este grupo se engloban comandos que devuelven información directa de distintos parámetros de funcionamiento del motor:
  - **AbsoluteLoadCommand:** Esta clase representa al comando Absolute Load Value, cuyo PID es $01_{16}\ 43_{16}$. Este comando devuelve, en tantos por ciento, el valor L de carga absoluto. La carga se calcula tomando los bytes $p(y, 5)$ y $p(y, 6)$ de la respuesta y para hacer el siguiente cálculo:

$$L = \frac{100}{255}(256p(y, 5)_{10} + p(y, 6)_{10}).$$

  - **LoadCommand:** Esta clase representa al comando Calculated Engine Load, cuyo PID es $01_{16}\ 04_{16}$. Este comando devuelve, en tantos por ciento, el valor EL de carga del motor. La carga se calcula tomando el byte $p(y, 5)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{EL} = \frac{100}{255}p(y, 5)_{10}.$$

  - **MassAirFlowCommand:** Esta clase representa al comando Mass Air Flow, cuyo PID es $01_{16}\ 10_{16}$. Este comando devuelve, en g/s, la cantidad de aire que entra al motor por unidad de tiempo. Este valor se conoce como MAF, y puede utilizarse para calcular el consumo de combustible instantáneo en conjunción con la velocidad del vehículo teniendo en cuenta que la mezcla estequiométrica de aire combustible en un motor de gasolina es de $14,7\ \mathrm{g}$ de aire por cada gramo de gasolina. En motores diésel, esta proporción es de $14,5\ \mathrm{g}$ de aire por cada gramo de gasóleo. Se calcula tomando los bytes $p(y, 5)$ y $p(y, 6)$ de la respuesta y para hacer el

siguiente cálculo:

$$\mathrm{MAF} = \frac{256p(y,5)_{10} + p(y,6)_{10}}{100}.$$

- **OilTempCommand:** Esta clase representa al comando Engine Oil Temperature, cuyo PID es $01_{16}\ 5C_{16}$. Este comando devuelve, en grados centígrados, el valor T de temperatura del aceite del motor. La temperatura se calcula tomando el byte A de la respuesta para hacer el siguiente cálculo:

$$\mathrm{T} = p(y,5) - 40.$$

- **RPMCommand:** Esta clase representa al comando Engine RPM, cuyo PID es $01_{16}\ 0C_{16}$. Este comando devuelve, en revoluciones por minuto, el valor R del régimen de revoluciones del motor. El régimen de revoluciones del motor se calcula tomando los bytes $p(y,5)$ y $p(y,6)$ de la respuesta y para hacer el siguiente cálculo:

$$R = \frac{256p(y,5)_{10} + p(y,6)_{10}}{4}.$$

- **RuntimeCommand:** Esta clase representa al comando Run Time Since Engine Start, cuyo PID es $01_{16}\ 1\mathrm{F}_{16}$. Este comando devuelve, en segundos, el tiempo RT que ha transcurrido desde que se puso en marcha el motor. Este tiempo se calcula tomando los bytes $p(y,5)$ y $p(y,6)$ de la respuesta para hacer el siguiente cálculo:

$$\mathrm{RT} = 256p(y,5)_{10} + p(y,6)_{10}.$$

- **ThrottlePositionCommand:** Esta clase representa al comando Throttle Position, cuyo PID es $01_{16}\ 11_{16}$. Este comando devuelve, en tantos por ciento, la posición TP del pedal del acelerador, desde 0, es decir, sin pisar, a 100, totalmente pisado. Este porcentaje se calcula tomando el byte $p(y,5)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{TP} = \frac{100}{255}p(y,5)_{10}.$$

- **Fuel:** En este grupo se engloban comandos que devuelven informa-

ción sobre parámetros relacionados con el sistema de combustible del vehículo:

- **AirFuelRatioCommand:** Esta clase representa al comando Fuel-Air Commanded Equivalence Ratio, cuyo PID es $01_{16}\ 44_{16}$. Este comando devuelve una ratio r_{AF} (menor que 2) de la mezcla de aire y gasolina requerida por el motor. La mezcla estequiométrica de aire y gasolina es $14,7:1$. Por ejemplo cuando el valor solicitado por el sistema de control de combustible es $0,97$, entonces la proporción aire/gasolina en la mezcla será $14,7:0,97$. El valor $r_{\mathrm{AF}} = 0,97$ ese será el valor devuelto por el sistema como respuesta a este comando. Esta ratio se calcula tomando los bytes $p(y,5)$ y $p(y,6)$ de la respuesta para hacer el siguiente cálculo:

$$r_{\mathrm{AF}} = \frac{2}{65536}(256p(y,5)_{10} + p(y,6)_{10}).$$

- **ConsumptionRateCommand:** Esta clase representa al comando Engine Fuel Rate, cuyo PID es $01_{16}\ 5\mathrm{E}_{16}$. Este comando devuelve, en $\mathrm{L/h}$, el volumen Cons de combustible consumido por el motor por cada hora. Esta cantidad se calcula tomando los bytes $p(y,5)$ y $p(y,6)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{Cons} = \frac{256p(y,5)_{10} + p(y,6)_{10}}{20}.$$

- **FindFuelTypeCommand:** Esta clase representa al comando Find Fuel Type, cuyo PID es $01_{16}\ 51_{16}$. Este comando devuelve, según la tabla 4.1. , el tipo de combustible que utiliza el vehículo. Para obtener el valor correspondiente de la tabla, se utiliza el valor del byte $p(y,5)_{10}$.

Valor	Tipo de combustible
0	No disponible
1	Gasolina
2	Metanol
3	Etanol
4	Diésel
5	GLP
6	GNC
7	Propano
8	Eléctrico
9	Bifuel/Gasolina
10	Bifuel/Metanol
11	Bifuel/Etanol
12	Bifuel/GLP
13	Bifuel/GNC
14	Bifuel/Propano
15	Bifuel/Electricidad
16	Bifuel/Híbrido
17	Híbrido/Gasolina
18	Híbrido/Etanol
19	Híbrido/Diésel
20	Híbrido/Eléctrico
21	Híbrido/Eléctrico con motor de combustión
22	Híbrido/Regenerativo
23	Bifuel/Diésel

Tabla 4.1: Tipos de combustible. Extraídos del enum FuelTypes de la OBD Java API.

- **FuelLevelCommand:** Esta clase representa al comando Fuel Tank Level Input, cuyo PID es $01_{16}\ 2\mathrm{F}_{16}$. Este comando devuelve, en tantos por ciento, la cantidad FuL de combustible disponible en el tanque de combustible. Esta cantidad se calcula tomando el byte $p(y, 5)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{FuL} = \frac{100}{255} p(y, 5)_{10}.$$

- **FuelTrimCommand:** Esta clase representa a los comandos relacionados con el Fuel Trim, cuyos PID son el 01_{16} 06_{16}, el 01_{16} 07_{16}, el 01_{16} 08_{16} y el 01_{16} 09_{16}. Este comando devuelve, en tantos por ciento, la riqueza FuT de la mezcla de combustible, según las necesidades del motor, indicadas por los sensores de O_2 del vehículo. Este porcentaje se calcula tomando el byte $p(y, 5)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{FuT} = \frac{100}{128} p(y, 5)_{10} - 100.$$

  Un valor negativo indica que la mezcla es demasiado rica en combustible, con lo que el sistema reduce la cantidad de combustible a utilizar en la mezcla. Por el contrario, un valor positivo, indica que la mezcla es demasiado pobre en combustible, con lo que el sistema aumenta la cantidad de combustible a utilizar en la mezcla.

- **WidebandAirFuelRatioCommand:** Esta clase representa a los comandos Wideband Air Fuel Ratio, cuyos PID son el 01_{16} 34_{16}, el 01_{16} 35_{16}, el 01_{16} 36_{16}, el 01_{16} 37_{16}, el 01_{16} 38_{16}, el 01_{16} 39_{16}, el 01_{16} $3\mathrm{A}_{16}$ y el 01_{16} $3\mathrm{B}_{16}$. Este comando devuelve una ratio r_{WAF} que indica la cantidad de oxígeno presente en los gases de escape. También devuelve la información devolviendo un valor I en mA. Los cálculos se hacen tomando los bytes $p(y, 5)$, $p(y, 6)$ $p(y, 7)$ y $p(y, 8)$ de la respuesta y para hacer los siguientes cálculos:

$$r_{\mathrm{WAF}} = \frac{2}{65536}(256 p(y, 5)_{10} + p(y, 6)_{10}).$$

$$I = \frac{256 p(y, 7)_{10} + p(y, 8)_{10}}{256} - 128.$$

- **Pressure:** En este grupo se engloban comandos que devuelven información sobre parámetros relacionados con la presión de algunos fluidos utilizados para el funcionamiento del vehículo:

  - **BarometricPressureCommand:** Esta clase representa al comando Barometric Pressure, cuyo PID es 01_{16} 33_{16}. Este comando devuelve, en kPa, el valor P de presión barométrica absoluta. La presión se calcula tomando el byte $p(y, 5)$ de la respuesta y para hacer el

siguiente cálculo:

$$P = p(y, 5)_{10}.$$

- **FuelPressureCommand:** Esta clase representa al comando Fuel Pressure, cuyo PID es 01_{16} $0\mathrm{A}_{16}$. Este comando devuelve, en kPa, el valor FuP de presión de combustible. La presión se calcula tomando el byte $p(y, 5)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{FuP} = 3p(y, 5)_{10}.$$

- **FuelRailPressureCommand:** Esta clase representa al comando Fuel Rail Pressure, cuyo PID es 01_{16} 23_{16}. Este comando devuelve, en kPa, el valor FuRP de presión de combustible en el riel de combustible. La presión se calcula tomando el byte $p(y, 5)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{FuRP} = 10(256p(y, 5)_{10} + p(y, 6)_{10}).$$

- **IntakeManifoldPressureCommand:** Esta clase representa al comando Intake Manifold Pressure, cuyo PID es 01_{16} $0\mathrm{B}_{16}$. Este comando devuelve, en kPa, el valor InP de la presión del colector de admisión. La presión se calcula tomando el byte $p(y, 5)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{IntakeManifoldPressure} = p(y, 5)_{10}.$$

- **PressureCommand:** Esta clase es una clase abstracta que extienden todas las clases relacionadas con comandos que devuelven valores de presión.

- **Temperature:** En este grupo se engloban comandos que devuelven información sobre parámetros relacionados con las temperaturas de algunos fluidos utilizados para el funcionamiento del vehículo:

  - **AirIntakeTemperatureCommand:** Esta clase representa al comando Intake Air Temperature, cuyo PID es 01_{16} $0\mathrm{F}_{16}$. Este comando devuelve, en $^{\mathrm{o}}\mathrm{C}$, el valor InT de temperatura del aire en la admisión. La temperatura InT se calcula tomando el byte $p(y, 5)$ de la

respuesta y para hacer el siguiente cálculo:

$$\mathrm{InT} = p(y, 5)_{10} - 40.$$

- **AmbientAirTemperatureCommand:** Esta clase representa al comando Ambient Air Temperature, cuyo PID es $01_{16}\ 46_{16}$. Este comando devuelve, en $^{o}\mathrm{C}$, el valor AmT de temperatura del aire ambiente. La temperatura se calcula tomando el byte $p(y, 5)$ de la respuesta y para hacer el siguiente cálculo:

$$\mathrm{AmT} = p(y, 5)_1 0 - 40.$$

- **EngineCoolantTemperatureCommand:** Esta clase representa al comando Engine Coolant Temperature, cuyo PID es $01_{16}\ 67_{16}$. Este comando devuelve, en $^{o}\mathrm{C}$, el valor EnT de temperatura del líquido refrigerante. La temperatura se calcula tomando el byte $p(y, 5)$ de la respuesta $y$para hacer el siguiente cálculo:

$$\mathrm{EnT} = p(y, 5)_{10} - 40.$$

- **TemperatureCommand:** Esta clase es una clase abstracta que extienden todas las clases relacionadas con comandos que devuelven valores de temperatura.

- **Protocol :** En este grupo se engloban comandos que devuelven información sobre parámetros relacionados con las temperaturas de algunos fluidos utilizados para el funcionamiento del vehículo. Se enumeran a continuación:

  - **AdaptiveTimingCommand:** Este comando representa el comando de tiempo de espera adaptativo, que por defecto está activado, y es la configuración recomendada. Si se usa el comando AT0 para desactivar el tiempo de espera adaptativo, entonces se tomará el tiempo especificado en el comando AT ST.
  - **AvailablePidsCommand:** Es la clase abstracta que extienden las siguientes tres clases.
  - **AvailablePidsCommand_01_20:** Devuelve los PID disponibles en el rango de PID del 01_{16} al 20_{16}.

- **AvailablePidsCommand_21_40:** Devuelve los PID disponibles en el rango de PID del 21_{16} al 40_{16}.
- **AvailablePidsCommand_41_60:** Devuelve los PID disponibles en el rango de PID del 01_{16} al 60_{16}.
- **DescribeProtocolCommand:** Esta clase representa al comando que muestra la descripción del protocolo utilizado en una conexión con un adaptador OBD.
- **DescribeProtocolNumberCommand:** Esta clase representa al comando que muestra el número correspondiente al protocolo utilizado durante la conexión con un adaptador OBD.
- **EchoOffCommand:** Esta clase representa al comando que permite desactivar el eco de respuesta en el adaptador OBD, de forma que al enviar una solicitud, el dispositivo no responda con la misma solicitud.
- **HeadersOffCommand:** Esta clase representa al comando que desactiva las cabeceras de los comandos OBD.
- **LineFeedOffCommand:** Esta clase representa al comando que desactiva el salto de línea en las respuestas recibidas por el adaptador OBD.
- **ObdProtocolCommand:** Esta es la calse abstracta que extienden los comandos DescribeProtocolCommand y DescribeProtocolNumberCommand.
- **ObdRawCommand:** Esta clase representa un comando no conocido que se envía a través del adaptador OBD. Puede utilizarse para enviar comandos específicos de un fabricante.
- **ObdResetCommand:** Esta clase representa al comando que reinicia la conexión OBD entre el adaptador y el sistema OBD del vehículo.
- **ObdWarmstartCommand:** Esta clase representa al comando que reinicia la conexión OBD entre el adaptador y el sistema OBD del vehículo y, seguidamente, establece la conexión de nuevo con el sistema.
- **ResetTroubleCodesCommand:** Esta clase representa al comando que elimina los DTCs y apaga la luz MIL.

- **SelectProtocolCommand:** Esta clase representa al comando que permite seleccionar un protocolo de conexión concreto entre el adaptador OBD y el sistema OBD del vehículo.
- **SpacesOffCommand:** Esta clase representa al comando que desactiva la auto-respuesta en el adaptador OBD.
- **TimeoutCommand:** Esta clase representa al comando que permite establecer un determinado tiempo de espera entre una petición OBD por parte del adaptador OBD y su respuesta por parte del sistema OBD del vehículo.

Además, esta parte incluye varias clases que no están englobadas en ninguno de estos grupos:

- **ObdCommand:** Esta clase es una clase abstracta que extienden todos los comandos anteriores. Define métodos para el envío del comando al adaptador OBD y para la recepción de la respuesta recibida por el adaptador OBD.
- **ObdMultiCommand:** Esta clase representa una lista de comandos que se envían al adaptador OBD y para los que se espera respuesta conjuntamente.
- **PercentageObdCommand:** Es la clase abstracta que extienden todas las clases que devuelven datos de tipo porcentual.
- **PersistentCommand:** Esta clase es una clase abstracta que extienden todas las clases que devuelven DTCs.
- **SpeedCommand:** Esta clase es representa al comando que devuelve la velocidad del vehículo, cuyo PID es $01_{16}\ 0\mathrm{D}_{16}$. Su valor S es devuelto en $\mathrm{km/h}$ y se calcula del siguiente modo:

$$\mathrm{S} = p(y, 5)_{10}.$$

- **SystemOfUnits:** Esta interfaz es implementada por todas las clases susceptibles a devolver unidades en el sistema imperial.

- **Enums:**

  - **AvailableCommandNames:** En este fichero se definen los valores de tipo String asignados a cada comando disponible para su utilización. Se utilizan en la definición de cada una de las clases que representan

a los comandos disponibles. La lista de comandos disponibles junto con su valor de tipo String se puede ver en la tabla 4.2.

- **FuelTrim:** En este fichero se definen los valores de tipo String asignados a cada comando de tipo Fuel Trim. Según el comando especificado, también se define a partir de este fichero el PID del comando utilizado. La lista de comandos Fuel Trim disponibles junto con su valor de tipo String se puede ver en la tabla 4.3.
- **FuelType:** En este fichero se definen los valores de tipo String asignados a cada tipo de combustible utilizado en la clase que representa al comando Fuel Type. La lista de tipos de combustible disponibles junto con su valor de tipo String se puede ver en la tabla 4.1.
- **ObdProtocols:** En este fichero se definen los valores hexadecimales asignados a cada protocolo. Estos valores se utilizan en las clases que representan a los comandos de consulta del protocolo utilizado Describe Protocol y Describe Protocol Number. La lista de protocolos disponibles junto con su valor hexadecimal y su descripción se puede ver en la tabla 4.4.

- **Exceptions:**

  - **BusInitException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD un mensaje del tipo "BUS INIT... ERROR".
  - **MisunderstoodCommandException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD un mensaje del tipo '?', que indica un comando no soportado.
  - **NoDataException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD un mensaje del tipo"NO DATA".
  - **NonNumericResponseException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD una respuesta sin número y se esperaba que los tuviera.
  - **ResponseException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD un mensaje de error genérico.
  - **StoppedException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD un mensaje del tipo "STOPPED".

Valor	Valor String
AIR_INTAKE_TEMP	Air Intake Temperature
AMBIENT_AIR_TEMP	Ambient Air Temperature
ENGINE_COOLANT_TEMP	Engine Coolant Temperature
BAROMETRIC_PRESSURE	Barometric Pressure
FUEL_PRESSURE	Fuel Pressure
INTAKE_MANIFOLD_PRESSURE	Intake Manifold Pressure
ENGINE_LOAD	Engine Load
ENGINE_RUNTIME	Engine Runtime
ENGINE_RPM	Engine RPM
SPEED	Vehicle Speed
MAF	Mass Air Flow
THROTTLE_POS	Throttle Position
TROUBLE_CODES	Trouble Codes
PENDING_TROUBLE_CODES	Pending Trouble Codes
PERMANENT_TROUBLE_CODES	Permanent Trouble Codes
FUEL_LEVEL	Fuel Level
FUEL_TYPE	Fuel Type
FUEL_CONSUMPTION_RATE	Fuel Consumption Rate
TIMING_ADVANCE	Timing Advance
DTC_NUMBER	Diagnostic Trouble Codes
EQUIV_RATIO	Command Equivalence Ratio
DISTANCE_TRAVELED_ AFTER_CODES_CLEARED	Distance since codes cleared
CONTROL_MODULE_VOLTAGE	Control Module Power Supply
ENGINE_FUEL_RATE	Engine Fuel Rate
FUEL_RAIL_PRESSURE	Fuel Rail Pressure
VIN	Vehicle Identification Number (VIN)
DISTANCE_TRAVELED_MIL_ON	Distance traveled with MIL on
TIME_TRAVELED_MIL_ON	Time run with MIL on
TIME_SINCE_TC_CLEARED	Time since trouble codes cleared
REL_THROTTLE_POS	Relative throttle position
PIDS_01_20	Available PIDs 01-20
PIDS_21_40	Available PIDs 21-40
PIDS_41_60	Available PIDs 41-60
ABS_LOAD	Absolute load
ENGINE_OIL_TEMP	Engine oil temperature
AIR_FUEL_RATIO	Air/Fuel Ratio
WIDEBAND_AIR_FUEL_RATIO	Wideband Air/Fuel Ratio
DESCRIBE_PROTOCOL	Describe protocol
DESCRIBE_PROTOCOL_NUMBER	Describe protocol number
IGNITION_MONITOR	Ignition monitor

Tabla 4.2: Nombres de comandos disponibles. Extraídos del enum AvailableCommand-Names de la OBD Java API.

Valor	Valor String
SHORT_TERM_BANK_1	Short Term Fuel Trim Bank 1
LONG_TERM_BANK_1	Long Term Fuel Trim Bank 1
SHORT_TERM_BANK_2	Short Term Fuel Trim Bank 2
LONG_TERM_BANK_2	Long Term Fuel Trim Bank 2

Tabla 4.3: Tipos de comandos Fuel Trim. Extraídos del enum FuelTrim de la OBD Java API.

Valor	Valor Hexadecimal	Descripción
AUTO	0	Auto
SAE_J1850_PWM	1	SAE J1850 PWM
SAE_J1850_VPW	2	SAE J1850 VPW
ISO_9141_2	3	ISO 9141-2
SO_14230_4_KWP	4	ISO 14230-4 KWP
ISO_14230_4_KWP_FAST	5	ISO 14230-4 KWP FAST
ISO_15765_4_CAN	6	ISO 15765-4 CAN
ISO_15765_4_CAN_B	7	ISO 15765-4 CAN B
ISO_15765_4_CAN_C	8	ISO 15765-4 CAN C
ISO_15765_4_CAN_D	9	ISO 15765-4 CAN D
SAE_J1939_CAN	A	SAE J1939 CAN
USER1_CAN	B	USER1 CAN
USER2_CAN	C	USER2 CAN

Tabla 4.4: Protocolos disponibles para el adaptador OBD. Extraídos del enum ObdProtocols de la OBD Java API.

- **UnableToConnectException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD un mensaje del tipo "UNABLE TO CONNEC".
- **UnknownErrorException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD un mensaje del tipo "ERROR".
- **UnsupportedCommandException:** Esta excepción se lanza cuando se recibe desde el adaptador OBD un mensaje del tipo "?", que indica un comando no soportado.

- **Utils:**
  - **CommandAvailabilityHelper:** Es una utilidad con la que se puede consultar la lista de comandos disponibles.

Como demostración de uso de esta librería, el desarrollador ha puesto a disposición de la comunidad de desarrolladores la aplicación para Android android-obd-reader[Pir13]. El proyecto puede ser consultado en el siguiente enlace que lleva al repositorio de GitHub: https://github.com/pires/android-obd-reader.

La aplicación consiste en una pantalla principal con varios campos de texto que muestran información sobre la velocidad del vehículo, la orientación, el consumo de combustible, el tiempo que el motor lleva en marcha e información sobre el estado del módulo GPS, el módulo Bluetooth y la conexión del dispositivo con el adaptador OBD. Además de estos campos de texto, se muestra, en el centro de la pantalla, una tabla en la que van apareciendo los comandos enviados por la aplicación y la respuesta recibida desde el adaptador OBD.

La aplicación desarrollada en este TFG utiliza un planteamiento similar, ejecutando un servicio en fondo que permite a la aplicación realizar comunicaciones entre lo que sucede en la interfaz gráfica y lo que sucede en el sistema OBD del vehículo sin que ello suponga bloquear el uso interactivo de la aplicación.

4.2.2. HelloCharts for Android

La segunda librería está dedicada a la representación de datos en distintos tipos de representaciones gráficas. En este TFG se ha optado por mostrar los datos en simples gráficas de líneas. La librería, sin embargo, permite múltiples tipos de representaciones gráficas de conjuntos de datos. Entre ellas, están las siguientes:

- **Line Chart**: Se trata de la representación de los datos en gráficas de líneas. Permite la representación de varios juegos de datos a la vez, lo cual facilita enormemente la comparación de distintos parámetros dando lugar a una herramienta de análisis muy valiosa, puesto que se puede observar cómo afecta la variación de los valores de cierto parámetro a los valores de cualquier otro.

- **Column Chart**: Permite representar los datos en gráficos de barras, pudiendo definir distintos colores para distintos juegos de datos en la misma gráfica. Es posible combinar este tipo de gráfica con las gráficas de líneas.

- **Pie Chart**: Permite representar un juego de datos en un gráfico de tarta para visualizar en qué proporción se distribuyen los valores de un determinado parámetro.

- **Bubble Chart**: Representa los datos mediante tres dimensiones, dos de ellas representadas mediante la posición de cada burbuja sobre los ejes y la restante representada mediante el tamaño de cada una de las burbujas.

- **Preview Line Chart**: Es la misma representación que en el gráfico de líneas, añadiendo la capacidad de desplazarse sobre el eje X de la gráfica, de forma que se facilita la representación de datos con un gran rango valores sobre este eje.

- **Preview Column Chart**: Al igual que la Preview Line Chart, añade a los gráficos de barras la capacidad de desplazarse sobre el eje X para facilitar la representación de datos con un rango muy grande de valores en el eje X.

El uso que se le ha dado a esta librería dentro de la aplicación es el de servir como forma de análisis de los datos obtenidos mediante la conexión con el adaptador OBD. Es una especie de telemetría que permite conocer la relación entre los distintos parámetros que el adaptador OBD es capaz de ofrecer y que puede contribuir a una análisis preciso de las condiciones del vehículo, pudiendo utilizarse para detectar incidencias en los sistemas que lo componen, entre otras cosas.

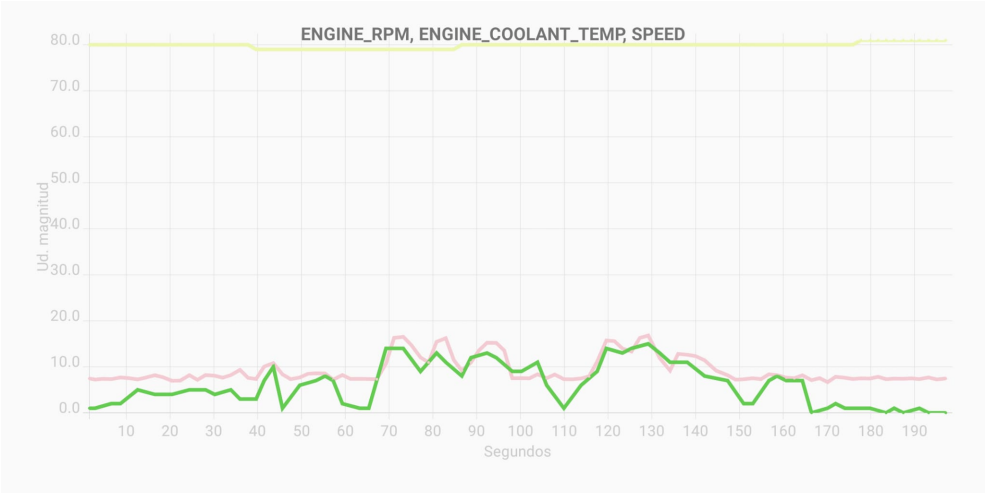


Figura 4.1: Varios parámetros representados mediante gráficos de líneas en una misma gráfica.[Lec]

4.2.3. Librerías de elementos gráficos

Las librerías gráficas que se han utilizado para mostrar los datos recibidos por la interfaz OBD son, en primer lugar, Android-RoundCornerProgressBar, que permite utilizar barras de estado vistosas con etiquetas en forma de iconos y con distintos colores, y, en segundo lugar, la librería plain-pie, que permite mostrar un gráfico en forma de tarta con unas características concretas que son útiles para mostrar datos en tiempo real sobre la velocidad del vehículo o las revoluciones por minuto a las que gira el motor.

4.2.3.1. La librería Android-RoundCornerProgresBar

La librería gráfica Android-RoundCornerProgressBar fue desarrollada por el desarrollador Tailandés con alias akexorcist[Ake15] en 2015 y está publicada en GitHub bajo una licencia Apache 2.0.

La decisión de utilizar en este TFG una librería como esta viene de la necesidad de encontrar elementos gráficos configurables y que sean fácilmente comprensibles por cualquier usuario. Android-RoundCornerProgressBar ofrece barras de progreso configurables tanto en color como en tamaño y, además, permite establecer iconos a cada una de las barras. Las barras de progreso generadas con esta librería constan de hasta 4 partes diferenciadas:

- **Barra de progreso principal:** Es la barra de progreso que queda al frente de

todo el elemento gráfico. Se puede configurar cuál es el valor máximo que puede tomar y se puede elegir si se quiere mostrar el progreso de forma invertida, de modo que la barra se irá rellenando de derecha a izquierda.

- **Barra de progreso secundaria:** Esta barra de progreso queda por detrás de la barra de progreso principal. Se puede configurar cuál es el valor máximo que puede tomar y se puede elegir si se quiere mostrar el progreso de forma invertida, de modo que la barra se irá rellenando de derecha a izquierda.

- **Fondo de la barra de progreso:** Este elemento queda por detrás del resto dentro del elemento gráfico.

- **Icono:** Este elemento es opcional, si se utiliza, el icono quedará a la izquierda de la barra de progreso. Se puede utilizar un icono propio si se desea.

La librería ofrece una serie de métodos que permiten configurar varios aspectos de las barras de progreso:

- **Colores:** La librería define métodos para modificar el color de todos los elementos de la barra de progreso. Se pueden definir, por separado, los colores de la barra de progreso principal, la barra de progreso secundaria, el fondo de las barras de progreso y el fondo del icono.

- **Iconos:** La librería define métodos con los que se puede modificar el icono mostrado por defecto por la librería, pudiendo utilizarse cualquier icono con formato png.

- **Tamaños:** Se pueden modificar los tamaños de las barras de progreso, del fondo y del icono. Además la librería también define métodos para modificar la separación entre la barra de progreso principal, la barra de progreso secundaria y el fondo. Incluye también métodos para aumentar la curvatura de las esquinas de todos los elementos gráficos, sin distinción entre ellos.

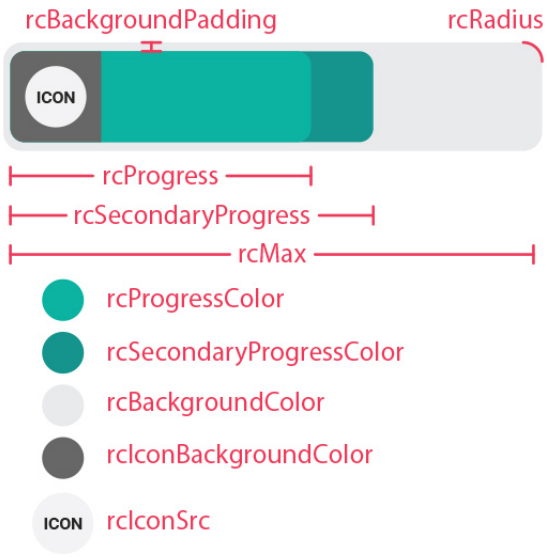


Figura 4.2: Parámetros configurables de las barras de progreso.[Ake15]

4.2.3.2. La librería plain-pie

La librería Plain-Pie fue desarrollada por Alejandro Zürcher[Zür18] en 2018 y también está publicada en GitHub bajo una licencia Apache 2.0.

La decisión de utilizar esta librería en este TFG viene de la necesidad de encontrar un elemento gráfico capaz de mostrar información sobre la velocidad o el régimen de revoluciones del motor de una forma parecida a la que se muestra esta misma información en los cuadros de mando de los vehículos comerciales actuales. Plain-pie ofrece vistas en forma de tarta que ofrecen la particular característica de mostrar información en el centro de la vista en forma de texto y con un fondo diferenciado del resto del elemento gráfico, con colores y tamaños configurables. Las partes del elemento gráfico son:

- **Elemento gráfico de progreso de la vista:** Esta es la parte principal de la vista. Indica gráficamente el porcentaje de la magnitud que se necesite mostrar. Para conseguir que sea coherente con cualquier rango de valores, hay que tener en cuenta que el máximo valor que puede tomar este elemento es 360, ya que el progreso en este elemento gráfico modifica el ángulo recorrido por el elemento gráfico de progreso. La librería incluye métodos que permiten añadir una animación al proceso de relleno del elemento gráfico de progreso.

- **Centro de la vista:** Esta parte se muestra en el centro del elemento gráfico de progreso y sobre ella es donde se muestra el texto de la vista. Se puede configurar el tamaño y color de esta parte del elemento gráfico.
- **Texto de la vista:** Esta parte se muestra sobre el centro de la vista. Se puede configurar para que muestre el porcentaje de progreso del elemento gráfico de progreso. También permite modificar el texto para mostrar un texto cualquiera, en cualquier color y tamaño.

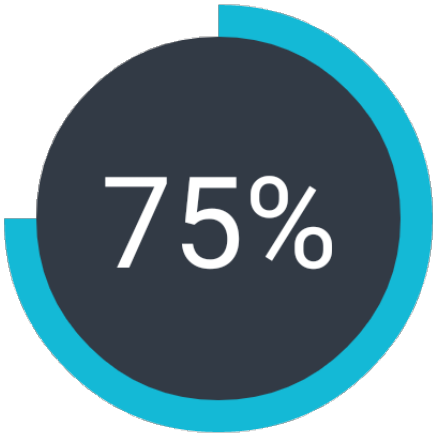


Figura 4.3: Ejemplo de elemento gráfico generado con la librería plain-pie.[Zür18]

4.3. OBDII Dashboard

En este punto, ya se han expuesto las bases del desarrollo de este TFG. Lo que sigue es la descripción del desarrollo de la aplicación OBDII Dashboard, repasando la creación de la interfaz gráfica y la experiencia de usuario, revisando los distintos modos disponibles y enfocando la descripción en el objetivo de convertir la aplicación en un cuadro de mandos para utilizar durante la conducción de un vehículo y una herramienta de análisis del funcionamiento del mismo.

En el momento de iniciar la aplicación, se han desarrollado una serie de actividades que permiten habilitar el módulo de Bluetooth del teléfono y establecer una conexión mediante Bluetooth con el adaptador OBD. La aplicación se encarga de comprobar si el dispositivo al que se conecta es un adaptador OBD, enviando una serie de comandos con los que, en caso de obtener respuesta, se

puede suponer que el dispositivo al otro lado de la comunicación es un adaptador OBD. Una vez se establece la conexión con el dispositivo elegido se pasa al menú principal de la aplicación, en el que, si el dispositivo conectado es un adaptador OBD, se iluminará en color azul el icono con la forma del conector OBD para indicar que, efectivamente, el dispositivo conectado es un adaptador OBD. Además se desbloquearán los iconos de los modos solo texto, cuadro de mandos y el de lectura y limpieza de errores. En caso de que el dispositivo conectado no sea un adaptador OBD, el icono indicador quedará en color negro, indicando que el dispositivo conectado no es un adaptador OBD y, además, se bloquearán los iconos de los modos solo texto, cuadro de mandos y el de lectura y limpieza de errores. Lo mismo ocurrirá si, al inicio de la aplicación, en lugar de elegirse la opción de conectar con un dispositivo Bluetooth, se elige la opción de modo sin conexión. Esta opción permite al usuario configurar la aplicación en caso de no disponer en algún momento de un adaptador OBD, además de dar la opción de usar el modo de análisis de datos, que ofrece un buscador de ficheros para elegir el log que se quiere visualizar y una interfaz que permite decidir qué parámetros quieren representarse mediante gráficas de barras.

Se describe a continuación el funcionamiento del modo de cuadro de mandos, que ofrece algunas configuraciones interesantes para el usuario y, posteriormente, se pasa a describir los distintos casos de uso posibles dentro de la aplicación.

En primer lugar, se han añadido 5 barras de progreso en las que se muestran los valores recibidos del adaptador OBD para determinados comandos. Por defecto se muestran:

- **Engine Coolant Temperature**
- **Engine Oil Temperature**
- **Engine Load**
- **Throttle Position**
- **Air Intake Temperature**

Las barras de progreso se pueden configurar en cuanto al comando que están mostrando. Para ello, se hace una pulsación larga sobre la barra de progreso que se quiere configurar. Para las barras de progreso, se han diseñado iconos representativos del comando que se muestra en cada una de ellas. Cada barra de

progreso tiene configurado un valor máximo acorde con el comando que muestra. Además, junto a cada barra de progreso, se muestra un texto que informa del valor representado en la barra de progreso correspondiente y de qué unidad se está utilizando.

En el código, se han creado estructuras de diccionario Hash para situar cada comando en su barra correspondiente, además de en el texto correspondiente. Estos diccionarios permiten, además, manejar la configuración de cada barra, estableciendo una relación entre el comando al que representan y ellas mismas. Lo mismo se aplica a los textos correspondientes a cada barra.

Se ha desarrollado para estas barras un método `OnLongClickListener()` que se encarga de configurar la interfaz gráfica asignando el comando correspondiente a cada barra, su valor máximo, el color que representa a ese comando y su icono.

Para los parámetros de velocidad y régimen de revoluciones del motor, se han utilizado dos elementos gráficos plain-pie. El elemento más grande se ha reservado para uno de los valores más importantes a la hora de conducir un vehículo, que es la velocidad. El régimen de revoluciones del motor se muestra en un elemento gráfico algo más pequeño pero con un color más llamativo para indicar el progreso. El máximo de RPM posible se puede configurar desde el menú general.

Se ha añadido además un indicador de la tensión de la batería del vehículo que va acompañado de un icono diseñado para representar dicho parámetro. Además aparece también un texto que indica la orientación del dispositivo y un texto que indica el módulo de la aceleración a la que está sometido el vehículo.

Para el diseño de la pantalla principal, se ha elegido un diseño de tipo ConstraintLayout, que utiliza las restricciones de los límites de los elementos gráficos para posicionar cada elemento dentro de la interfaz, haciendo que ésta sea adaptable a varios tipos de pantalla sin necesidad de editar el diseño de la pantalla principal de forma específica para cada elemento gráfico.

Para algunas funcionalidades se han empleado fragmentos de código de la aplicación que el desarrollador de la librería de comandos OBD publicó como ejemplo de uso. Las clases mencionadas se encuentran en un paquete separado dentro del código de la aplicación, de manera que se diferencia claramente qué código pertenece al desarrollo de este TFG y qué código se ha tomado de la aplicación de ejemplo de la librería.

Para describir el funcionamiento de la aplicación, se hace uso de una serie

de diagramas de casos de uso, que mostrarán, desde el punto de vista de un observador externo, cuál es el manejo y funcionamiento de la aplicación. En esta aplicación no se diferencia entre varios tipos de usuario ni entre grupos de usuarios.

4.4. Casos de uso

4.4.1. Inicio de sesión de recogida de datos en vivo

Caso de uso	Iniciar sesión de recogida de datos en vivo
Resumen	Un usuario quiere iniciar la recogida de datos en vivo de su vehículo
Actores	Usuario
Precondición	El dispositivo Bluetooth conectado debe ser un adaptador OBD
Postcondición	Se comienzan a recoger datos en vivo desde el adaptador OBD
Curso Normal	1. El usuario inicia la aplicación 2. El usuario sigue el caso de uso de la tabla 4.10 para conectarse a un adaptador OBD 3. El usuario elige desde el menú principal el modo solo texto o el modo cuadro de mandos 4. La aplicación actualiza el estado OBD con el comando enviado al adaptador OBD 5. La aplicación comienza a recoger los datos
Curso Alternativo: El dispositivo no es un adaptador OBD	1. La aplicación se conecta pero no permite iniciar ningún modo que implique el uso de la conexión con el adaptador OBD 2. La aplicación actualiza el indicador de adaptador OBD a negro

Tabla 4.5

Siempre que se inician los modos solo texto o cuadro de mandos, se empieza a guardar un log si éste está activo como se muestra en el caso de uso de la tabla 4.26, en el directorio que se puede configurar siguiendo el caso de uso de la tabla 4.28. La aplicación actualizará los elementos gráficos o el texto y recogerá datos hasta que el usuario detenga la recogida de datos en vivo siguiendo el caso de uso de la tabla 4.6.

4.4.2. Detención de la recogida de datos en vivo

Caso de uso	Detener sesión de recogida de datos
Resumen	Un usuario quiere detener la recogida de datos en vivo de su vehículo
Actores	Usuario
Precondición	Debe haberse iniciado una sesión de datos en vivo como se indica en el caso de uso de la tabla 4.5
Postcondición	Se detiene la sesión de recogida de datos y se guarda el log en el directorio indicado en el caso de uso de la tabla 4.28
Curso Normal	1. El usuario utiliza el control del sistema operativo para volver a la actividad anterior 2. La aplicación guarda el log de la sesión y para de mostrar datos

Tabla 4.6

Al detener la recogida de datos en vivo, si el usuario ha configurado la recogida del log de datos siguiendo el caso de uso de la tabla 4.26, la aplicación guardará el log en el directorio que se puede configurar siguiendo el caso de uso de la tabla 4.28. La aplicación volverá al menú principal y estará lista para iniciar un análisis o una nueva sesión de recogida de datos en vivo.

4.4.3. Configuración de las barras de progreso

Caso de uso	Configurar las barras de progreso
Resumen	Un usuario quiere cambiar el parámetro que muestra una barra de progreso
Actores	Usuario
Precondición	La aplicación debe estar iniciada y en el modo cuadro de mandos
Postcondición	Se modifica el parámetro correspondiente a la barra de progreso
Curso Normal	1. El usuario hace una pulsación larga sobre la barra de progreso que desea modificar 2. La aplicación muestra un menú contextual con todos los parámetros disponibles 3. El usuario selecciona el parámetro que desea que muestre la barra de progreso que está configurando 4. Se actualizan el icono, el color, el texto correspondiente a las unidades y el parámetro del que muestra datos la barra de progreso
Curso Alternativo: El parámetro ya se muestra en otra barra de progreso	1. La aplicación muestra un mensaje indicando al usuario que el parámetro seleccionado ya se está mostrando 2. La aplicación no actualiza ni el icono, ni el color, ni el texto correspondiente a las unidades, ni el parámetro del que muestra datos la barra de progreso

Tabla 4.7

Esta configuración permite al usuario elegir qué parámetros devueltos por el adaptador OBD de los disponibles quiere que se muestren en la pantalla principal. En caso de que el usuario haya desactivado la recogida de algún parámetro siguiendo el caso de uso de la tabla 4.25, el parámetro desactivado no actualizará la barra de progreso, pero ésta sí que quedará configurada para mostrar dicho parámetro. El usuario puede, posteriormente, seguir el caso de uso de la tabla 4.25 para reactivar la recogida del parámetro en cuestión.

4.4.4. Obtención de DTC

Se muestra la lista de DTC almacenados en la memoria desde el último borrado de DTC. En la vista de DTC se muestran los DTC con su código y una descripción del mismo.

Caso de uso	Obtención de DTC
Resumen	Un usuario quiere iniciar el proceso de obtención de DTC
Actores	Usuario
Precondición	La aplicación debe estar iniciada y en el menú principal habiéndose conectado a un adaptador OBD siguiendo los pasos del caso de uso 4.10
Postcondición	Se pasa a la vista de DTC
Curso Normal	1. El usuario pulsa sobre el icono del modo de DTC 2. El usuario pulsa sobre el icono de obtención de DTC 3. La aplicación muestra la vista de DTC 4. Se muestran los DTC recogidos

Tabla 4.8

4.4.5. Configurar el Identificador del vehículo

Caso de uso	Configurar el Identificador del vehículo para utilizar en el log de datos recogidos en tiempo real
Resumen	Un usuario quiere configurar el Identificador del vehículo
Actores	Usuario
Precondición	Iniciar la aplicación y conectar a un adaptador OBD o elegir el modo sin conexión
Postcondición	El identificador del vehículo queda guardado
Curso Normal	1. El usuario elige la opción de ajustes del menú principal de la aplicación 2. La aplicación inicia la vista ajustes 3. El usuario pulsa sobre la opción ID del vehículo 4. El usuario introduce el ID del vehículo que quiere utilizar para el log de datos 5. El usuario confirma la acción pulsando el botón Aceptar 6. El ID del vehículo queda configurado en la aplicación

Tabla 4.9

Esta opción permite asignar un Identificador único a cada vehículo. El usuario es el responsable de asegurarse de que el Identificador del vehículo es único.

4.4.6. Activar el Bluetooth

Caso de uso	Activar el uso del Bluetooth en la aplicación
Resumen	Un usuario quiere utilizar el Bluetooth en la aplicación
Actores	Usuario
Precondición	1. Iniciar la aplicación 2. Aceptar los permisos de ubicación de Android
Postcondición	Se activa la conexión Bluetooth
Curso Normal	1. El usuario elige la opción Buscar dispositivos 2. La aplicación inicia la vista de búsqueda de dispositivos 3. El usuario elige un dispositivo al que conectarse 4. La aplicación pasa a la vista de conexión, mostrando información sobre el dispositivo seleccionado 5. El usuario pulsa sobre el icono de Bluetooth 6. La aplicación se conecta con el dispositivo seleccionado y, si es un adaptador OBD, actualiza el icono a color azul

Curso Alternativo	1. El usuario elige la opción Buscar dispositivos 2. La aplicación inicia la vista de búsqueda de dispositivos 3. El usuario elige un dispositivo al que conectarse 4. La aplicación pasa a la vista de conexión, mostrando información sobre el dispositivo seleccionado 5. El usuario pulsa sobre el icono de Bluetooth 6. La aplicación se conecta con el dispositivo seleccionado y, si no es un adaptador OBD, actualiza el icono a color negro y bloquea los iconos de los modos sujetos a conexión

Tabla 4.10

Esta opción permite al usuario activar el uso del Bluetooth del dispositivo Android en la aplicación y conectar con un adaptador OBD.

4.4.7. Desactivar el Bluetooth

Esta opción permite al usuario desactivar el uso del Bluetooth del dispositivo Android en la aplicación.

Caso de uso	Desactivar el uso del Bluetooth en la aplicación
Resumen	Un usuario quiere desactivar el uso del Bluetooth en la aplicación

Actores	Usuario
Precondición	1. Iniciar la aplicación
Postcondición	Se desactiva el uso del Bluetooth en la aplicación
Curso Normal	1. El usuario accede a los ajuste del sistema Android 2. El usuario desactiva el módulo Bluetooth, dentro de la categoría Bluetooth del menú de ajustes del sistema operativo 3. Queda desactivado el uso del Bluetooth en la aplicación y ésta muestra un botón para volver a habilitarlo

Tabla 4.11

4.4.8. Activar el GPS

Esta opción permite al usuario activar el uso del GPS del dispositivo Android en la aplicación.

Caso de uso	Activar el uso del GPS en la aplicación
Resumen	Un usuario quiere activar el uso del GPS en la aplicación
Actores	Usuario

Precondición	1. Iniciar la aplicación hasta el menú principal 2. Tener la ubicación del teléfono activa
Postcondición	Se activa el uso del GPS en la aplicación
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario activa el checkbox de la opción de Permitir GPS, dentro de la categoría GPS del menú general 4. Queda activado el uso del GPS en la aplicación

Tabla 4.12

4.4.9. Desactivar el GPS

Esta opción permite al usuario desactivar el uso del GPS del dispositivo Android en la aplicación.

Caso de uso	Desactivar el uso del GPS en la aplicación
Resumen	Un usuario quiere desactivar el uso del GPS en la aplicación
Actores	Usuario

Precondición	1. Iniciar la aplicación hasta el menú principal 2. Tener la ubicación del teléfono activa
Postcondición	Se desactiva el uso del GPS en la aplicación
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario desactiva el checkbox de la opción de Permitir GPS, dentro de la categoría GPS del menú de ajustes 4. Queda desactivado el uso del GPS en la aplicación.

Tabla 4.13

4.4.10. Configurar el periodo de actualización del GPS en segundos

Esta opción permite al usuario configurar el periodo, en segundos, con el que la aplicación actualizará la posición GPS utilizando el módulo GPS del dispositivo Android. Por defecto el periodo es 1 s. Si el usuario introduce un valor no válido, se usa el valor por defecto.

Caso de uso	Configurar el periodo de actualización del GPS en segundos
Resumen	Un usuario quiere configurar el periodo de actualización del GPS en segundos
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurado el periodo de actualización del GPS en segundos
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Actualizar periodo en segundos, dentro de la categoría GPS del menú general 4. El usuario introduce, en segundos, el número de segundos 5. Queda configurado el periodo de actualización del GPS en segundos

Tabla 4.14

4.4.11. Configurar el periodo de actualización del GPS en metros

Esta opción permite al usuario configurar el periodo, en metros, con el que la aplicación actualizará la posición GPS utilizando el módulo GPS del disposi-

tivo Android. Por defecto el periodo es 5 m. Si el usuario introduce un valor no válido, se usa el valor por defecto.

Caso de uso	Configurar el periodo de actualización del GPS en metros
Resumen	Un usuario quiere desactivar el uso del GPS en la aplicación
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurado el periodo de actualización del GPS en metros
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Actualizar periodo en metros, dentro de la categoría GPS del menú general 4. El usuario introduce, en metros, el número de metros 5. Queda configurado el periodo de actualización del GPS en metros

Tabla 4.15

4.4.12. Configurar el protocolo OBD a utilizar

Esta opción permite al usuario configurar el protocolo a utilizar por el Adaptador OBD. Las opciones se muestran en la tabla 4.4.

Caso de uso	Configurar el protocolo a utilizar por el Adaptador OBD
Resumen	Un usuario quiere configurar el protocolo a utilizar por el Adaptador OBD
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurado el protocolo a utilizar por el Adaptador OBD
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Protocolo OBD, dentro de la categoría Preferencias OBD del menú general 4. La aplicación muestra la lista de protocolos a utilizar 5. El usuario selecciona el protocolo deseado 6. Queda configurado el protocolo a utilizar por el Adaptador OBD

Tabla 4.16

4.4.13. Activar el sistema de unidades Imperial

Esta opción permite al usuario activar el uso del del sistema imperial de unidades en todos los parámetros recogidos por la aplicación.

Caso de uso	Activar el sistema de unidades imperial
Resumen	Un usuario quiere activar el uso del sistema de unidades imperial
Actores	Usuario
Precondición	1. Iniciar la aplicación hasta el menú principal 2. Tener la opción de Unidades imperiales desactivada
Postcondición	Se activa el uso del sistema de unidades imperial en la aplicación
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario activa el checkbox de la opción de Unidades imperiales, dentro de la categoría Preferencias OBD del menú general 4. Queda activado el uso del sistema de unidades imperial en la aplicación.

Tabla 4.17

4.4.14. Desctivar el sistema de unidades Imperial

Esta opción permite al usuario desactivar el uso del del sistema imperial de unidades para utilizar el sistema de unidades internacional en todos los pará-

metros recogidos por la aplicación.

Caso de uso	Desctivar el sistema de unidades imperial
Resumen	Un usuario quiere desactivar el uso del sistema de unidades imperial
Actores	Usuario
Precondición	1. Iniciar la aplicación hasta el menú principal 2. Tener la opción de Unidades imperiales activada
Postcondición	Se desactiva el uso del sistema de unidades imperial en la aplicación. Se usará el sistema de unidades internacional
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario desactiva el checkbox de la opción de Unidades imperiales, dentro de la categoría Preferencias OBD del menú general 4. Queda desactivado el uso del sistema de unidades imperial en la aplicación. Se usará el sistema de unidades internacional

Tabla 4.18

4.4.15. Configurar el periodo de actualización de los comandos en milisegundos

Caso de uso	Configurar el periodo de actualización de los comandos en milisegundos
Resumen	Un usuario quiere configurar el periodo de actualización de los comandos en milisegundos
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurado el periodo de actualización de los comandos en milisegundos
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Actualizar periodo en milisegundos, dentro de la categoría Preferencias OBD del menú general 4. El usuario introduce, en milisegundos, el número de milisegundos 5. Queda configurado el periodo de actualización de los comandos en milisegundos

Tabla 4.19

Esta opción permite al usuario configurar el periodo, en milisegundos, con el que la aplicación actualiza los comandos. Es decir, el periodo con el que se hacen consultas al adaptador OBD. Por defecto, tras varias pruebas en distintos vehículos, se ha configurado esta opción en 260 ms, ya que, según [Elm] permite la máxima velocidad de actualización bajo el protocolo CAN Bus al ser 50 ms el tiempo de respuesta de la red CAN al adaptador OBD.

4.4.16. Configurar el máximo de RPM en el vehículo

Caso de uso	Configurar el máximo de RPM en el vehículo
Resumen	Un usuario quiere configurar el máximo de RPM en el vehículo
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurado el máximo de RPM en el vehículo
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción máximo de RPM en el vehículo, dentro de la categoría Preferencias OBD del menú general 4. El usuario introduce, en RPM, el número de RPM máximas que quiere que se representen

Tabla 4.20

Esta opción permite al usuario configurar el máximo de RPM en el vehículo. Gracias a este parámetro, el elemento gráfico de la pantalla principal que muestra las RPM del vehículo en tiempo real, puede adaptar su máximo valor al vehículo concreto que se está utilizando, mostrando una proporción adecuada en el porcentaje de progreso.

4.4.17. Configurar el valor máximo de economía de combustible

Caso de uso	Configurar el valor máximo de economía de combustible
Resumen	Un usuario quiere configurar el valor máximo de economía de combustible
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurado el valor máximo de economía de combustible
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Máximo de economía de combustible, dentro de la categoría Preferencias OBD del menú general 4. El usuario introduce, en tantos por ciento, el porcentaje máximo de economía del combustible 5. Queda configurado el valor máximo de economía de combustible

Tabla 4.21

Esta opción permite al usuario configurar el valor máximo de economía de combustible. Este valor se utiliza para el cálculo de consumo de combustible en

vehículos sin sensor MAF.

4.4.18. Configurar el valor de eficiencia volumétrica del vehículo

Caso de uso	Configurar el valor de eficiencia volumétrica del vehículo
Resumen	Un usuario quiere configurar el valor de eficiencia volumétrica del vehículo
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurado el valor de eficiencia volumétrica del vehículo
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Eficiencia volumétrica, dentro de la categoría Preferencias OBD del menú general 4. El usuario introduce, en proporciones, el valor de eficiencia volumétrica del vehículo 5. Queda configurado el valor de eficiencia volumétrica del vehículo

Tabla 4.22

Esta opción permite al usuario configurar el valor de eficiencia volumétrica del vehículo. Este valor se utiliza para el cálculo de consumo de combustible en vehículos sin sensor MAF. Este valor es la efectividad que puede alcanzar un motor en el proceso de admisión al llenar el cilindro. Los vehículo con sensor MAF son capaces de calcular el flujo de aire que entra al motor en $\mathrm{g/s}$.

4.4.19. Configurar la cilindrada en litros

Esta opción permite al usuario configurar la cilindrada del motor del vehículo en litros. Este valor se utilizar para el cálculo de consumo de combustible en vehículos con sensor MAF.

Caso de uso	Configurar la cilindrada del vehículo
Resumen	Un usuario quiere configurar la cilindrada del motor del vehículo en litros
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurada la cilindrada del motor del vehículo en litros
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Cilindrada del motor, dentro de la categoría Preferencias OBD del menú general 4. El usuario introduce, en litros, el valor de la cilindrada del motor del vehículo 5. Queda configurada la cilindrada del motor del vehículo en litros

Tabla 4.23

4.4.20. Configurar los comandos utilizados para configurar la conexión del adaptador OBD

Caso de uso	Configurar los comandos utilizados para configurar la conexión del adaptador OBD
Resumen	Un usuario quiere configurar los comandos utilizados para configurar la conexión del adaptador OBD
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Quedan configurados los comandos utilizados para configurar la conexión del adaptador OBD
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Comandos de configuración del adaptador OBD, dentro de la categoría Preferencias OBD del menú general 4. El usuario introduce los comandos utilizando el conjunto de comandos Hayes 5. Quedan configurados los comandos utilizados para configurar la conexión del adaptador OBD

Tabla 4.24

Esta opción permite al usuario configurar los comandos utilizados para configurar la conexión del adaptador OBD. Se pueden utilizar comandos del conjunto de comandos Hayes [Hay].

4.4.21. Configurar los comandos OBD a enviar al adaptador OBD

Esta opción permite al usuario activar y desactivar cada uno de los comandos OBD disponibles en la aplicación.

Caso de uso	Configurar los comandos OBD a enviar al adaptador OBD
Resumen	Un usuario quiere configurar los comandos OBD a enviar al adaptador OBD
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Quedan configurados los comandos OBD a enviar al adaptador OBD
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Comandos OBD, dentro de la categoría Preferencias OBD del menú general 4. El usuario marca los comandos OBD que desea que se envíen al adaptador OBD 5. Quedan configurados los comandos OBD a enviar al adaptador OBD

Tabla 4.25

4.4.22. Activar el log completo de datos recogidos en vivo

Caso de uso	Activar el log completo de datos recogidos en vivo
Resumen	Un usuario quiere activar el log completo de datos recogidos en vivo
Actores	Usuario
Precondición	1. Iniciar la aplicación hasta el menú principal 2. La opción Permitir log completo debe estar desactivada, como se indica en el caso de uso de la tabla 4.27 o por defecto
Postcondición	Queda activado el log completo de datos recogidos en vivo
Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario activa el checkbox de la opción de Permitir log completo, dentro de la categoría Configuración del log del menú general 4. Queda activado el log completo de datos recogidos en vivo

Tabla 4.26

Esta opción permite al usuario activar el log completo de datos recogidos en vivo. El log se irá escribiendo a medida que se reciben las respuestas de los comandos. Los comandos que no obtengan respuesta se informarán con el valor `null` dentro del log. Una vez se detiene la recogida de datos en vivo, tal como se indica en el caso de uso de la tabla 4.6, la aplicación guardará el log en el directorio proporcionado por el usuario siguiendo el caso de uso de la tabla 4.28. Los logs de sesiones de recogida de datos en vivo quedan separados por ficheros, de forma que por cada sesión que se inicie se generará un nuevo fichero. Los ficheros pueden exportarse para realizar análisis de datos avanzados.

4.4.23. Desactivar el log completo de datos recogidos en vivo

Esta opción permite al usuario desactivar el log completo de datos recogidos en vivo. Una vez desactivado, no será posible generar gráficas para el análisis posterior de los parámetros recogidos durante la sesión de recogida de datos en vivo. Las sesiones anteriores seguirán disponibles para el análisis posterior en el directirio configurado por el usuario siguiendo el caso de uso de la tabla 4.28.

Caso de uso	Desactivar el log completo de datos recogidos en vivo
Resumen	Un usuario quiere desactivar el log completo de datos recogidos en vivo
Actores	Usuario
Precondición	1. Iniciar la aplicación hasta el menú principal 2. La opción Permitir log completo debe estar activada, como se indica en el caso de uso de la tabla 4.26
Postcondición	Queda desactivado el log completo de datos recogidos en vivo

Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario desactiva el checkbox de la opción de Permitir log completo, dentro de la categoría Configuración del log del menú general 4. Queda desactivado el log completo de datos recogidos en vivo

Tabla 4.27

4.4.24. Configurar el nombre del directorio para el log de datos

Esta opción permite al usuario configurar el nombre del directorio para el log de datos recogidos en tiempo real. Si el directorio existe, el log se guardará en dicho directorio. Si no existe, la aplicación creará el directorio automáticamente.

Caso de uso	Configurar el nombre del directorio para el log de datos
Resumen	Un usuario quiere configurar el nombre del directorio para el log de datos
Actores	Usuario
Precondición	Iniciar la aplicación hasta el menú principal
Postcondición	Queda configurado el nombre del directorio para el log de datos

Curso Normal	1. El usuario entra en los ajustes de la aplicación 2. La aplicación inicia la vista del menú de ajustes 3. El usuario pulsa sobre la opción Nombre del directorio, dentro de la categoría Configuración del log del menú general 4. El usuario introduce un nombre para el directorio 5. Queda configurado el nombre del directorio para el log de datos

Tabla 4.28

4.4.25. Salir de la aplicación

Caso de uso	Salir de la aplicación
Resumen	Un usuario quiere salir de la aplicación
Actores	Usuario
Precondición	Iniciar la aplicación
Postcondición	Se sale de la aplicación
Curso Normal	1. El usuario utiliza las funciones del sistema operativo Android para salir de la aplicación (botón inicio) 2. Se sale de la aplicación

Curso Alternativo: La aplicación estaba iniciada y la recogida de datos estaba iniciada	1. El usuario detiene la recogida de datos en vivo, como se indica en el caso de uso de la tabla 4.6 2. El usuario utiliza las funciones del sistema operativo Android para salir de la aplicación (botón inicio) 3. Se sale de la aplicación

Tabla 4.29

Salir correctamente de la aplicación permite que, en caso de estar recogiendo datos y tener el log de datos configurado, los datos se guarden en un fichero csv dentro del directorio indicado por el usuario.

Capítulo 5

Conclusiones

5.1. Conclusiones

Los productos de este TFG han sido la aplicación OBDII Dashboard y esta misma memoria.

Gran parte del trabajo realizado en este TFG ha sido de investigación. Se ha llevado a cabo un estudio del estado del arte, profundizando en los motivos que han llevado a los fabricantes a incluir redes de conexión de los sistemas electrónicos que hay dentro de sus vehículos y enumerando algunas de las ventajas que esto conlleva. Más en concreto, se ha repasado el funcionamiento del bus CAN que utilizan los vehículos de hoy en día, con el objetivo de comprender cómo se comunican las distintas unidades electrónicas que van instaladas en cada vehículo y se ha estudiado una clasificación de estos sistemas, separándolos en dominios que atienden al área del vehículo en el que operan.

Se han enumerado aplicaciones existentes en el mercado, capaces de extraer información del sistema OBD con ayuda de un adaptador OBD y se ha dado la motivación para desarrollar una aplicación capaz de ofrecer esa información de forma que sea comprensible de forma rápida y cuyo manejo sea lo más sencillo posible, puesto que el objetivo es el de servir de cuadro de mandos avanzado para la conducción y herramienta de análisis de datos.

Una vez visto el estado del arte y dada la motivación para el desarrollo de este TFG, se ha estudiado en profundidad el funcionamiento del bus CAN, revisando cómo se hace la comunicación a nivel físico y de enlace de datos. Inmediatamente después se ha estudiado una forma estándar de acceder a algunos datos del conjunto de sistemas electrónicos que controlan el motor, las emisiones y el

diagnóstico de averías, el sistema OBD. Se han repasado las versiones de OBD existentes y se ha analizado la forma en que este sistema puede devolver la información a un dispositivo que sepa interpretar las respuestas del sistema OBD. Para ello se ha estudiado el lenguaje de comandos utilizado y se han descrito los comandos más utilizados y más comunes en los vehículos.

Posteriormente se han estudiado los módulos Bluetooth y GPS y se ha realizado un análisis de la arquitectura del sistema operativo utilizado, Android, con el objetivo de establecer un punto de partida para la descripción del desarrollo de la aplicación.

En el transcurso de este TFG se ha pretendido, bajo la premisa de obtener una aplicación que sea realmente útil durante la conducción, desarrollar una aplicación que permita obtener datos relevantes del estado del vehículo en tiempo real. Durante el desarrollo de la aplicación, en la etapa de desarrollo de la función de conectividad Bluetooth e investigación de las API disponibles para realizar la conexión con el adaptador OBD, se encontró una librería de código libre y con licencia Apache 2.0, que permite el uso de la misma, siempre mencionando al autor, para la creación de proyectos relacionados con los comandos OBD. En este punto, se ha desarrollado una aplicación que se encarga de hacer peticiones al sistema de diagnóstico y de recoger los datos, añadiendo una interfaz gráfica que aporta una buena experiencia de usuario durante la conducción y permite además realizar distintas tareas de análisis de datos y telemetría.

Se ha explicado qué librerías han sido utilizadas en el desarrollo de esta aplicación y cuál es su funcionamiento. Los elementos gráficos incluidos en la interfaz gráfica muestran la información de una forma que permite conocer el estado de cada parámetro sin suponer una gran distracción, tal y como hace un cuadro de mandos de un vehículo normal.

El desarrollo de la aplicación en sí, en lo que a código se refiere, se ha llevado más tiempo en investigación y análisis que en desarrollo propiamente dicho. El resultado de esta investigación es la documentación aportada en cuanto a los sistemas y protocolos implicados. Otro punto que ha llevado tiempo ha sido el de desarrollar un diseño estético sencillo y actual junto con un conjunto de iconos descriptivos para cada parámetro. Estos iconos se han creado utilizando el programa Illustrator de la suite de Adobe CC.

El resultado de este TFG permite a cualquier usuario tener un medio con el que realizar telemetría sobre su vehículo, análisis de averías y además permite aprovechar dispositivos Android para obtener un cuadro de mandos avanzado

y fácil de usar. La aplicación cobra más sentido cuando se observa que el mercado de smartphones cada vez se orienta más a dispositivos con grandes pantallas. Y no solo eso, también el mercado de dispositivos multimedia en automóviles está orientándose al uso de sistemas de doble DIN con pantallas multimedia de gran tamaño equipados, normalmente, con el sistema operativo Android en versiones adaptadas por cada fabricante de estos sistemas.

5.2. Lineas futuras

Quedan abiertas múltiples lineas de trabajo posibles con este TFG, pudiendo ampliarse la funcionalidad de la aplicación añadiendo opciones que utilicen los distintos parámetros de una forma inteligente para brindar datos útiles ya tratados al usuario de la aplicación, como podrían ser estadísticas de conducción, avisos de exceso de velocidad, predicción de averías utilizando datos de un gran número de vehículos, consejos de mantenimiento de los sistemas del vehículo, mejoras en el apartado de análisis de la aplicación, etc.

Referencias

[Ake15] Akexorcist. *Android-RoundCornerProgressBar*. 2015. URL: https://github.com/akexorcist/Android-RoundCornerProgressBar (visitado 16-09-2019).

[Anda] *"Arquitectura de la plataforma", en Android Developers*. URL: https://developer.android.com/guide/platform (visitado 16-09-2019).

[Andb] *"Arquitectura de la plataforma", en Android Developers*. URL: https://developer.android.com/guide/platform (visitado 16-09-2019).

[Andc] *"Meet Android Studio", en Android Developers*. URL: https://developer.android.com/studio/intro?hl=en (visitado 16-09-2019).

[Aos] *"Debugging ART Garbage Collection", en Android Open Source Project*. URL: https://source.android.com/devices/tech/dalvik/gc-debug (visitado 11-11-2019).

[BJ19] Khac-Hoai Nam Bui y Jason J. Jung. "Computational negotiation-based edge analytics for smart objects". En: *Information Sciences* 480 (2019), págs. 222 -236. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2018.12.046. URL: http://www.sciencedirect.com/science/article/pii/S002002551830985X.

[Blu] *"Bluetooth", en Wikipedia*. URL: https://en.wikipedia.org/wiki/Bluetooth (visitado 16-09-2019).

[Bus] *"Bus_CAN", en Wikipedia*. URL: https://es.wikipedia.org/wiki/Bus_CAN (visitado 10-09-2019).

[Elm] *ELM327 OBD to RS232 Interpreter Integrated Circuit, data sheet for firmware version 1.4b*. Elm Electronics Inc., 2012. URL: https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf.

[Esd] *"CAN Remote Frames on Recall", en esd electronics*. URL: http://www.esd-electronics-usa.com/CAN-Remote-Frames.html (visitado 11-09-2019).

[Gps] *"Global Positioning System", en Wikipedia*. URL: https://en.wikipedia.org/wiki/Global_Positioning_System (visitado 16-09-2019).

[Gre] *"Mazda unveiling KAI Concept with SKYACTIV-X SPCCI engine; more details on the technology", en Green Car Congress*. URL: https://www.greencarcongress.com/2017/10/20171025-spcci.html (visitado 10-09-2019).

[Hay] *Comandos Hayes*. URL: https://en.wikipedia.org/wiki/Hayes_command_set (visitado 18-09-2019).

[KFB14] Sylvain Kubler, Kary Främling y Andrea Buda. "A standardized approach to deal with firewall and mobility policies in the IoT". En: *Pervasive and Mobile Computing* 20 (sep. de 2014). DOI: 10.1016/j.pmcj.2014.09.005.

[LH02] G. Leen y D. Heffernan. "Expanding automotive electronic systems". En: *Computer* 35.1 (2002), págs. 88-93. DOI: 10.1109/2.976923.

[Lec] *HelloCharts for Android*. 2014. URL: https://github.com/lecho/hellocharts-android (visitado 13-10-2019).

[NSL17] N. Navet y F. Simonot-Lion. *Automotive Embedded Systems Handbook*. Industrial Information Technology. CRC Press, 2017. ISBN: 9780849380273. URL: https://books.google.com.ar/books?id=vB700Gb4RtkC.

[Obd] *"On-board diagnostics", en Wikipedia*. URL: https://en.wikipedia.org/wiki/On-board_diagnostics (visitado 11-09-2019).

[Ora] Oracle. *Documentación de Java SE*. URL: https://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/intro.html#wp9502 (visitado 16-09-2019).

[Pid] *"PIDs", en Wikipedia*. URL: https://en.wikipedia.org/wiki/OBD-II_PIDs (visitado 12-09-2019).

[Pir13] Pires. *andoid-obd-reader*. 2013. URL: https://github.com/pires/android-obd-reader (visitado 16-09-2019).

[ZLQ14] Shu-en Zhao, Yuling Li y Xian Qu. "Vehicle Chassis Integrated Control Based on Multimodel and Multilevel Hierarchical Control". En: *Mathematical Problems in Engineering* 2014 (abr. de 2014), págs. 1-13. DOI: 10.1155/2014/248676.

[Zür18] Alejandro Zürcher. *plain-pie*. 2018. URL: https://github.com/zurche/plain-pie (visitado 16-09-2019).

Apéndice A

Manual de usuario

A.1. Introducción

Este desarrollo está basado en el uso de la librería, que está publicada en GitHub bajo una licencia Apache 2.0. La librería, que ha sido desarrollada por un desarrollador que se hace llamar Pires, dejó de tener soporte a mediados de 2017.

OBDII Dashboard es una aplicación que ofrece un cuadro de mandos configurable para utilizar con cualquier vehículo que utilice el protocolo CAN bus como red de comunicación entre sistemas. Además ofrece modos de obtención de códigos de error de la ECU y de análisis de datos en tiempo real con un modo solo texto o mediante gráficas en el modo análisis utilizando registros de las sesiones de análisis en tiempo real o del modo cuadro de mandos.

Para el funcionamiento del cuadro de mandos y el modo solo texto se recogen datos en vivo a través del adaptador OBD con el objetivo de actualizar los indicadores del cuadro de mandos o los registros del modo solo texto en tiempo real. En cada sesión de recogida de datos en vivo se puede generar un log con los datos recogidos en cada instante de tiempo que se guarda en un fichero en formato csv. El modo de análisis se aprovecha de estos ficheros csv para brindar al usuario una interfaz de análisis sencillo. Se pueden utilizar estos ficheros csv para exportarlos a otros sistemas que permitan un análisis más avanzado de los datos.

Se utilizan sensores del dispositivo Android como el GPS, el sensor de orientación o el de aceleración si están presentes en el dispositivo que ejecuta la aplicación para la recogida de la posición geográfica, la orientación del vehículo y

las aceleraciones soportadas por el mismo. La conexión con el vehículo se hace a través de una conexión Bluetooth con el adaptador OBD que deberá estar equipado con un módulo Bluetooth.

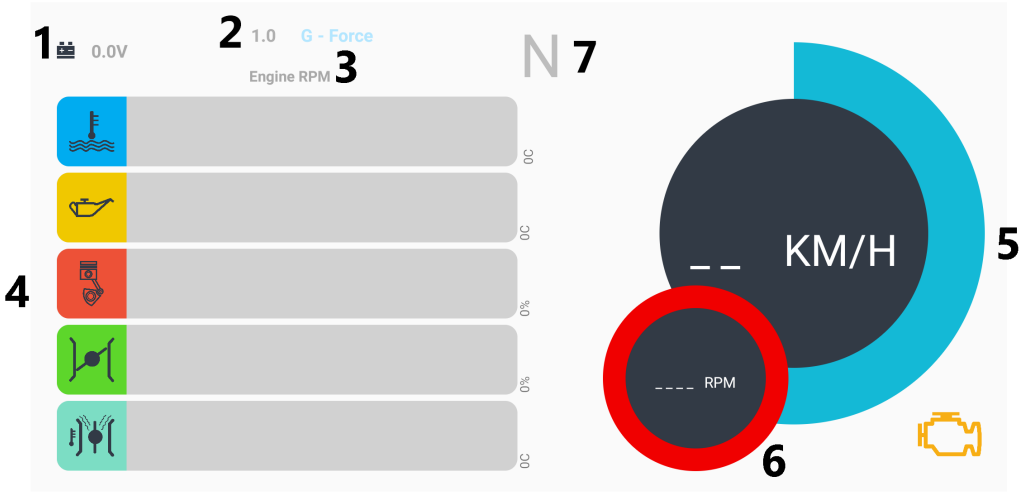
El siguiente manual pretende dar todas las indicaciones necesarias para el manejo y configuración de todas las funciones disponibles en la aplicación.

A.2. Requisitos

Los requisitos para poder ejecutar la aplicación y utilizar todas las funcionalidades son los siguientes:

- Versión de Android mayor o igual a 5.0 en el dispositivo Android.
- Adaptador OBD con módulo Bluetooth que implemente la interfaz OBDII.
- Módulo Bluetooth en el dispositivo Android.
- Módulo GPS en el dispositivo Android.
- Sensor de orientación en el dispositivo Android.
- Acelerómetro en el dispositivo Android.
- Vehículo con puerto de diagnóstico OBD que utilice el protocolo CAN bus.

A.3. Leyenda del cuadro de mandos



1- Indicador de voltaje de la batería
2- Indicador de acelreación
3- Indicador de estado OBD
4- Indicadores de parámetros OBD
5- Indicador de velocidad
6- Indicador de RPM
7- Indicador de orientación

Figura A.1: Cuadro de mandos.

A.4. Inicio de la aplicación

Al iniciar la aplicación, se muestra una pantalla con un botón para activar el módulo Bluetooth del teléfono, un botón para iniciar el modo sin conexión y un texto que muestra el estado del módulo Bluetooth del teléfono. Si se pulsa el botón para activar el módulo Bluetooth del teléfono, el sistema operativo preguntará al usuario si desea permitir la activación del módulo Bluetooth por parte de la aplicación. Si el usuario permite la activación, el estado se actualizará para mostrar el estado activado del módulo Bluetooth del teléfono y, además, desaparecerá el botón de activación del módulo Bluetooth para dar lugar a la aparición del botón para iniciar la búsqueda de dispositivos.

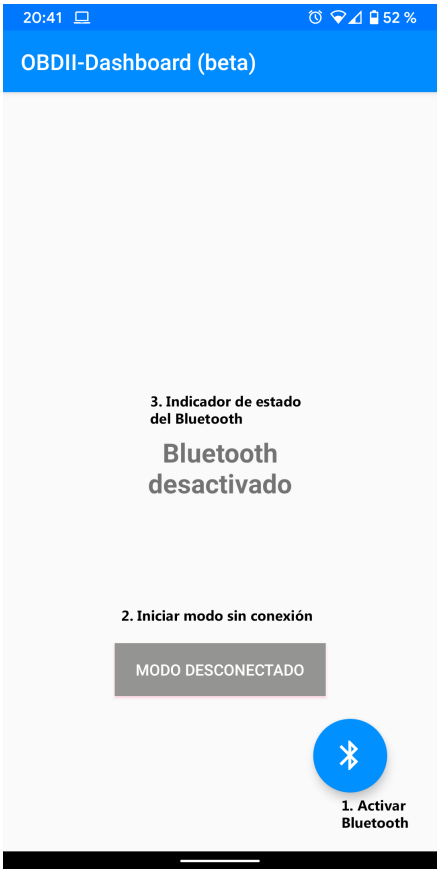


Figura A.2: Pantalla de inicio de la aplicación cuando el Bluetooth está desactivado.

Figura A.3: Pantalla de inicio de la aplicación cuando el Bluetooth está activado.

Si se elige la opción de búsqueda de dispositivos, se pasará a una pantalla en la que aparecerá una lista con los dispositivos Bluetooth que se vayan encontrando dentro del rango de conexión del dispositivo. Cuando el usuario seleccione un dispositivo de la lista, se pasará a una pantalla que mostrará el nombre y la dirección MAC del dispositivo elegido y un botón cuya pulsación iniciará el proceso de conexión con dicho dispositivo.

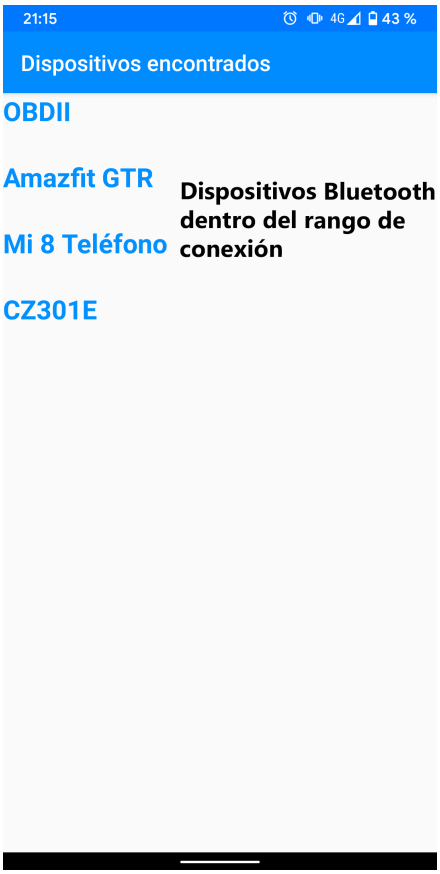


Figura A.4: Pantalla dispositivos encontrados dentro del rango de conexión.

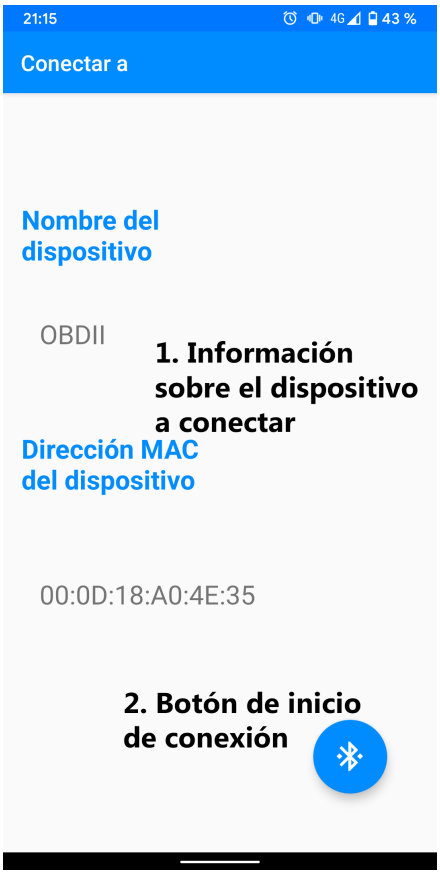


Figura A.5: Pantalla de conexión con el dispositivo seleccionado.

Si el dispositivo al que se intenta conectar no se trata de un adaptador OBD, entonces se mostrará un error.

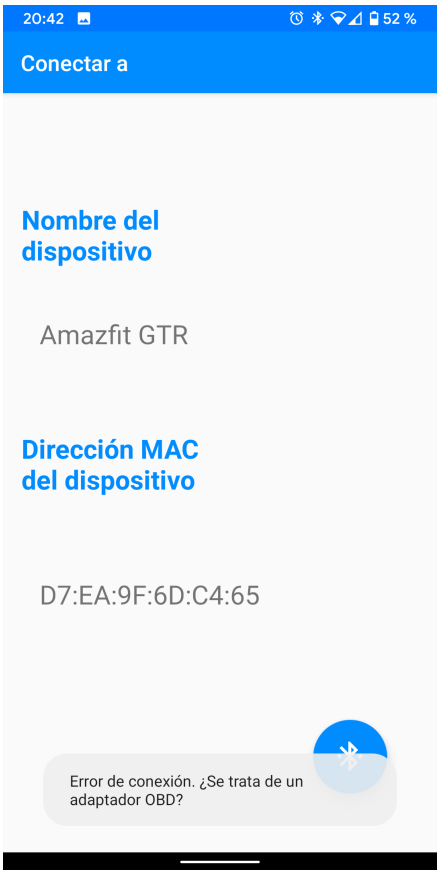


Figura A.6: Pantalla de conexión con el dispositivo seleccionado.

Por el contrario, si el dispositivo sí que se trata de un adaptador OBD, entonces se pasará al menú principal y se coloreará de azul el indicador de adaptador OBD, situado en la parte superior izquierda. Además, se desbloquearán los modos solo texto, cuadro de mandos y de DTC.

Figura A.7: Menú principal de la aplicación con el indicador de adaptador OBD en gris, sin conexión OBD. Modo sin conexión.

A partir del menú principal de la aplicación, se pueden iniciar distintos modos que se comentan a continuación.

A.5. Instrucciones de uso del modo solo texto

El modo solo texto es un modo en el que simplemente se muestran los datos tal como se reciben del adaptador OBD, sin utilizar gráficos para mostrar los valores. No existe ningún tipo de configuración dentro de este modo.

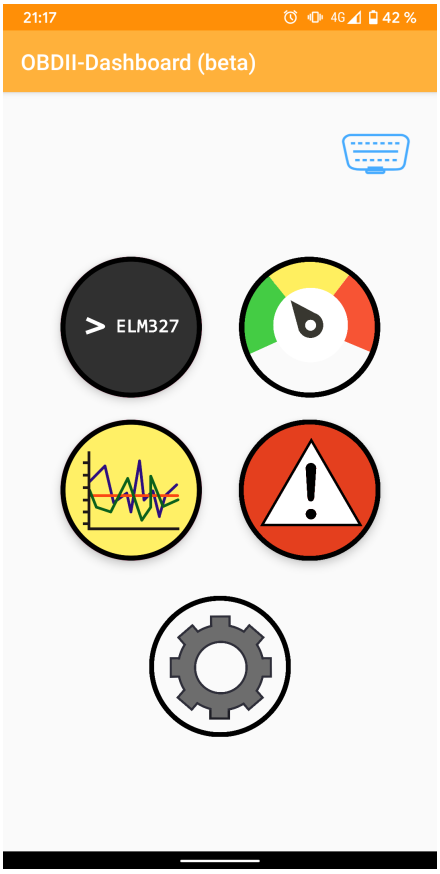


Figura A.8: Menú principal de la aplicación con el indicador de adaptador OBD en azul, con conexión OBD. Modo con conexión.

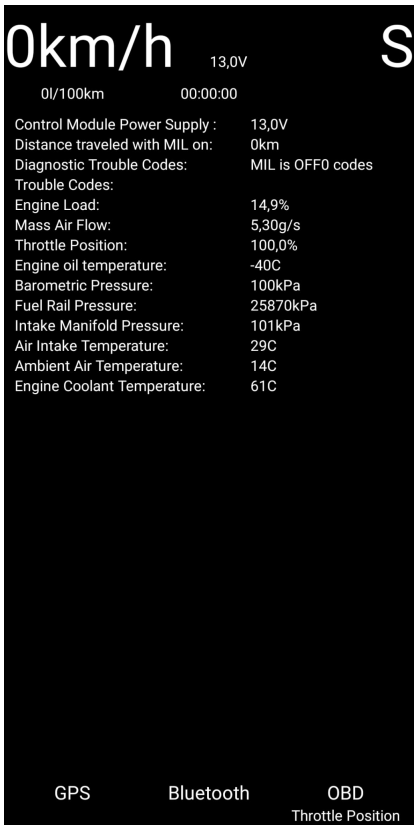


Figura A.9: Modo solo texto.

A.6. Instrucciones de uso del modo de análisis

En este modo, se puede navegar dentro del directorio de ficheros de log configurado desde el menú de ajustes de la aplicación. Simplemente es una lista desplazable en la que aparecen todos los ficheros disponibles en el directorio indicado. El usuario puede seleccionar cualquier fichero para abrir la vista de análisis mediante gráficas. En primer lugar se carga una gráfica vacía para, posteriormente, mediante un gesto de swipe, mostrar los posibles parámetros a representar en el gráfico de líneas. Pueden representarse tantos parámetros como se quiera. En el momento de leer el fichero de log, se generan los botones que permiten elegir qué parametro debe ser mostrado y con qué color es representado.

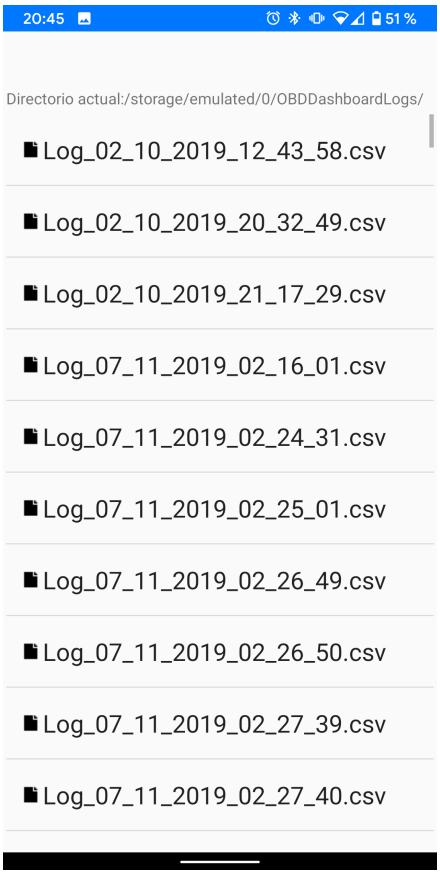


Figura A.10: Explorador de archivos dentro de la aplicación.

Figura A.11: Parámetros que el usuario puede elegir para el análisis.

En la figura A.12 se puede observar cómo se superponen varias gráficas pertenecientes a distintos parámetros con el objetivo de hacer un análisis de la rela-

ción entre dichos parámetros.

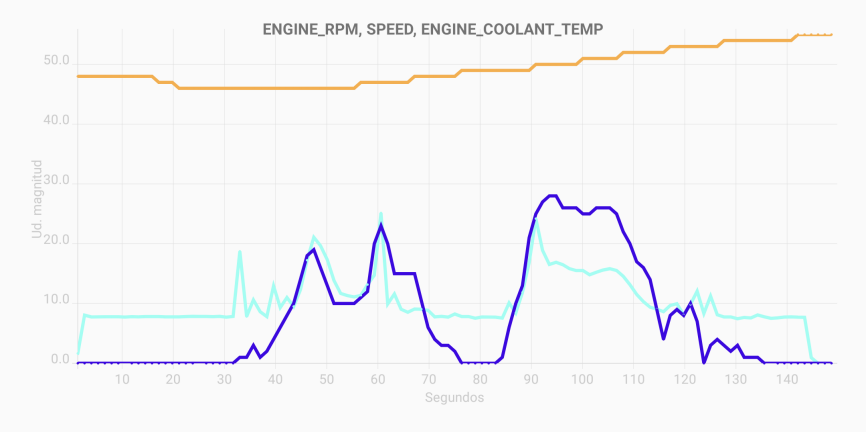


Figura A.12: Pantalla de análisis. Muestra una representación gráfica de los parámetros elegidos.

A.7. Instrucciones de uso del cuadro de mandos

El cuadro de mandos consta de cinco indicadores configurables que muestran el valor del parámetro para el que están configurados. Por defecto, los parámetros que se muestran son:

- Engine coolant temperature.
- Engine oil temperature.
- Engine load.
- Throttle position.
- Air intake temperature.

Cada parámetro tiene su correspondiente color e icono. Para configurar cualquiera de los cinco indicadores, se debe hacer una pulsación larga sobre el indicador que se desea configurar. Aparecerá una lista desplazable con 13 parámetros disponibles.

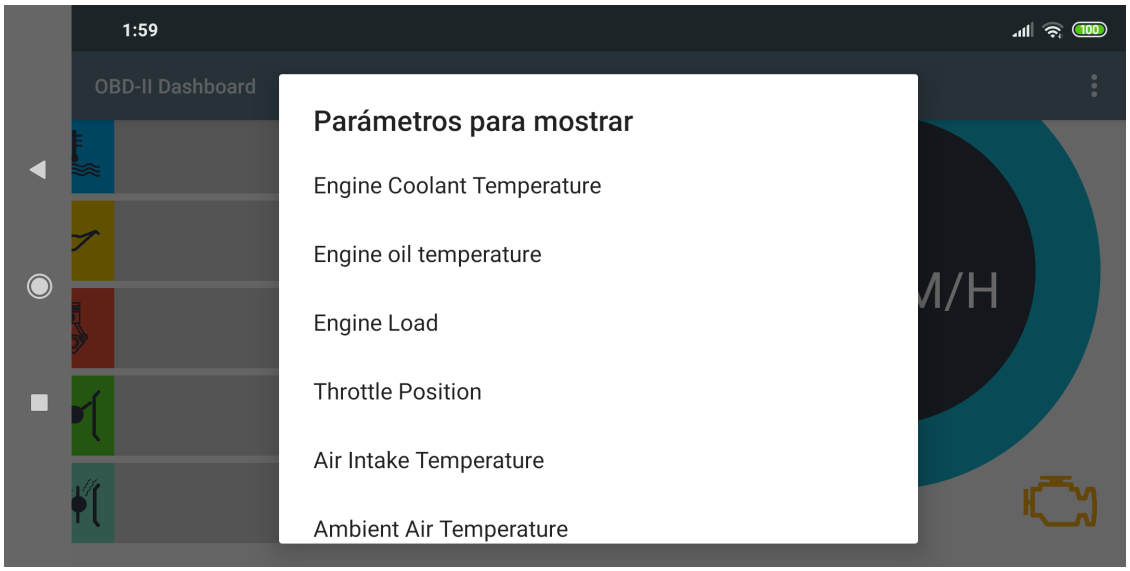


Figura A.13: Lista desplazable de parámetros OBD.

La lista completa de parámetros OBD que se pueden visualizar es la siguiente:

- **Engine coolant temperature:** 
- **Engine oil temperature:** 
- **Engine load:** 
- **Throttle position:** 
- **Air intake temperature:** 
- **Ambient air temperature:** 
- **Intake manifold pressure:** 
- **Fuel rail pressure:** 
- **Fuel pressure:** 
- **Barometric pressure:** 
- **Fuel level:** 
- **Fuel consumption rate:** 
- **Mass air flow:** 

A la derecha de cada uno de los indicadores aparecen los valores de los parámetros mostrados en las unidades correspondientes.

Además de los cinco indicadores horizontales, el cuadro de mandos dispone de un indicador del voltaje de la batería del vehículo expresado en voltios, un indicador de orientación que sirve de brújula, el indicador de aceleraciones, y dos indicadores circulares, uno para indicar la velocidad del vehículo y otro para indicar el régimen de revoluciones del motor en RPM (revoluciones por minuto).

El indicador del régimen de revoluciones del motor permite configurar el número de RPM máximo para ajustar la proporción del gráfico circular. A este ajuste se accede a partir del menú de ajustes de la aplicación.

El proceso que funciona en fondo se encarga de actualizar el indicador de estado OBD con los comandos que se envían al adaptador OBD.

A.8. Instrucciones del modo DTC

El modo DTC consta de dos opciones, que se dan a elegir al usuario al entrar en dicho modo. La primera de ellas es la de mostrar los posibles DTC almacenados en la memoria del sistema OBD del vehículo. La segunda se trata de una opción para eliminar de la memoria del sistema OBD del vehículo los posibles DTC que se pueden encontrar mediante la opción anterior.

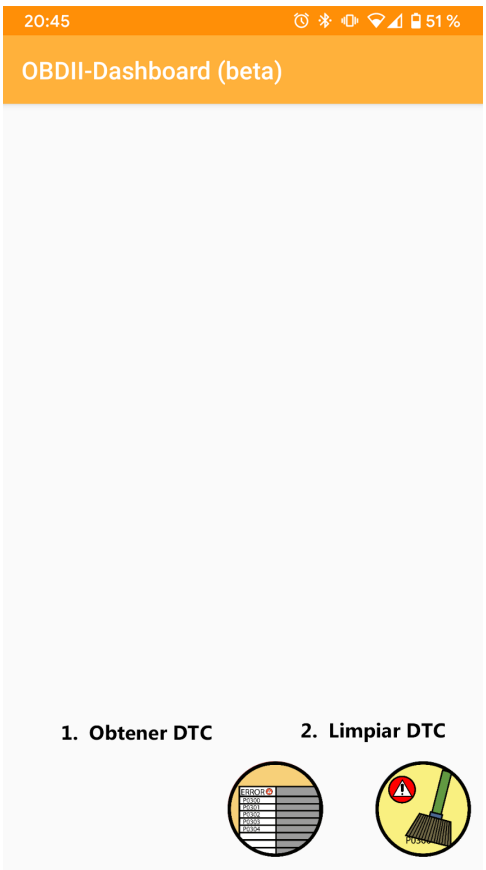


Figura A.14: Menú DTC.

Cuando el usuario elige la primera opción, la aplicación muestra una pantalla que se va rellenando con los distintos DTC que puedan aparecer.

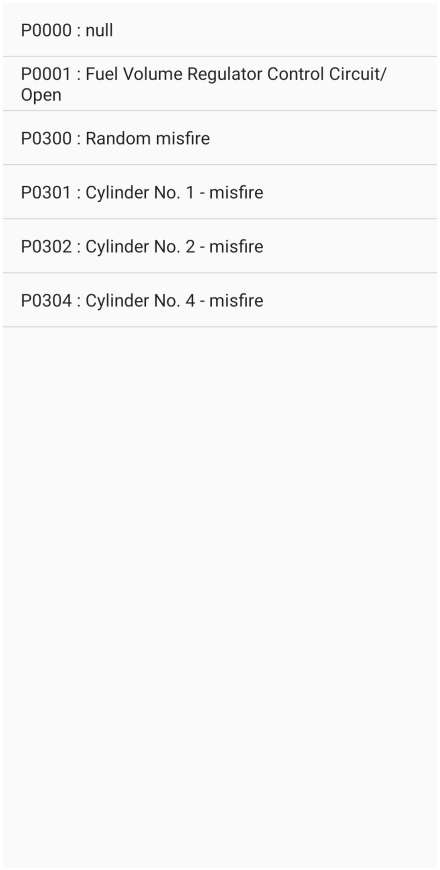


Figura A.15: Lista de DTC obtenida del sistema OBD.

Al elegir la segunda opción, simplemente se muestra un mensaje informando de que los DTC han sido eliminados.

A.9. Instrucciones del menú de ajustes de la aplicación

El menú de ajustes de la aplicación, que aparece ilustrado en la figura A.16, está estructurado en cinco categorías y dentro de cada una de ellas hay diferentes ajustes relacionados que se detallan en la siguiente lista:

- **Data Upload:** Ajustes relacionados con la subida de los datos en fondo.
    - **Vehicle ID:** Identificador para discernir entre los datos de distintos vehículos.

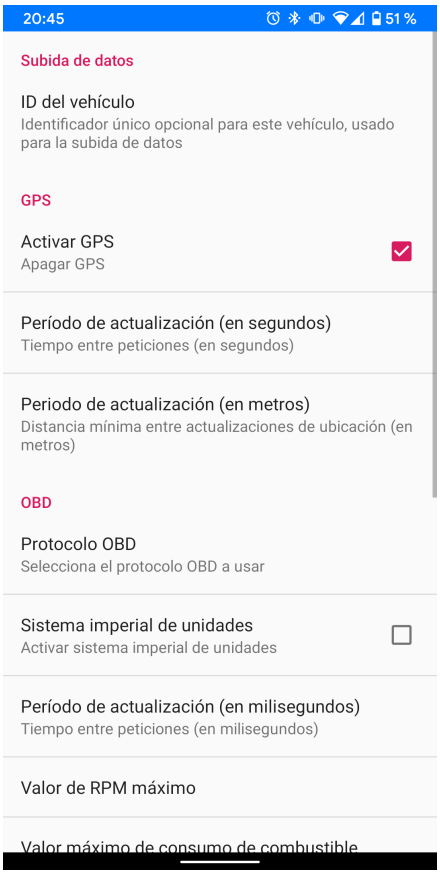


Figura A.16: Menú de ajustes de la aplicación.

- **GPS:** Ajustes relacionados con el GPS.
  - **Enable GPS:** Activar o desactivar el GPS.
  - **Update Period in Seconds:** Permite establecer cada cuántos segundos se consultará la posición GPS.
  - **Update Period in Meters:** Permite establecer la distancia mínima en metros en la que se consultará la posición GPS.
- **OBD Preferences:** Ajustes relacionados con el Adaptador OBD.
  - **OBD Protocol:** Permite seleccionar el protocolo OBD a utilizar en la conexión. Por defecto selecciona el protocolo automáticamente y es la configuración recomendada.
  - **Imperial Units:** Permite establecer las unidades del sistema imperial.

- **Update Period in Miliseconds:** Permite establecer cada cuántos milisegundos se enviarán comandos al adaptador OBD. La configuración por defecto es 260 milisegundos y es la recomendada. Si no se reciben datos, es conveniente aumentar el valor.
- **Max RPM value:** Permite establecer el máximo de revoluciones del vehículo. Este ajuste se utiliza para obtener una proporción correcta en el indicador del régimen de revoluciones del motor. Por defecto su valor es 7500 RPM.
- **Maximum Fuel Economy Value:** Permite establecer el máximo valor de economía del combustible.
- **Volumetric Efficiency:** Permite establecer la eficiencia volumétrica para el cálculo del consumo de combustible en vehículos sin sensor de masa del flujo de aire.
- **Engine Displacement (liters):** Permite establecer la cilindrada del motor para usarla en el cálculo del consumo de combustible en vehículos sin sensor de masa del flujo de aire.
- **Reader Config Commands:** Permite escribir los comandos utilizados para configurar la conexión con el adaptador OBD. Por defecto se utilizan los comandos AT SP 0 para seleccionar automáticamente el protocolo de conexión del adaptador OBD con la ECU del vehículo y AT Z para reiniciar el adaptador OBD y comenzar una conexión limpia.

- **OBD Commands:** Ajustes relacionados con los comandos a utilizar en cada consulta del adaptador OBD a las ECUs del vehículo.
  - **OBD Commands:** Permite marcar qué comandos se quiere enviar al adaptador OBD en cada intervalo de consulta (referente a la configuración Update Period in Miliseconds de la categoría OBD Preferences). Cuanto menor sea el número de comandos seleccionados, mejor frecuencia de actualización tendrá la aplicación.
- **Full logging configuration:** Ajustes relacionados con el log de la aplicación.
  - **Enable full logging:** Permite activar el log que escribe en un fichero los datos recogidos en vivo durante una sesión de recogida de datos en vivo.

- **Directory name:** Permite establecer el directorio donde se guardarán los ficheros de log.

A.10. Diagrama de flujo de la aplicación

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga