Universidad Zaragoza
1542

Trabajo Fin de Grado

Implementación de un enchufe inteligente para vehículos eléctricos basado en Arduino

Autor

Cristian Villarroya Sánchez

Director

Francisco José Martínez Domínguez

Escuela Universitaria Politécnica de Teruel

2019

Agradecimientos

En primer lugar, quiero expresar mi agradecimiento a mis padres por permitir que haya estudiado esta carrera tanto económicamente como anímicamente ya que son los que me impulsaron a ello.

A toda mi familia en general, por todo el apoyo que me han dado siempre, en especial a mi hermana que pesar de las peleas de hermanos, siempre estamos ahí para ayudarnos.

Agradecer también a todas las personas involucradas en este TFG, mención especial para mi tutor Francisco, que ha sido el que me ha dado esta oportunidad.

Finalmente agradecer también a mi compañero y amigo Alberto por toda la ayuda que nos hemos prestado a lo largo de la carrera y, los buenos momentos vividos.

"Implementación de un enchufe inteligente para vehículos eléctricos basado en Arduino"

RESUMEN

En este Trabajo de Fin de Grado se ha realizado el diseño e implementación de un enchufe inteligente basado en Arduino que puede ser controlado por el usuario mediante una aplicación móvil.

El enchufe posee 3 modos diferentes: (i) el modo manual, donde el enchufe se encuentra activado al conectar el vehículo, y por tanto, comienza el proceso de carga, (ii) el modo emergencia, donde el enchufe se encuentra activo hasta que se alcanza un determinado porcentaje de batería y después se espera a realizar la recarga en un periodo donde ésta sea más económica, y (iii) el modo ahorro, donde el enchufe no se activa hasta que un servidor externo determina cuándo es más barato iniciar el proceso de recarga del vehículo.

La utilización de este enchufe inteligente otorga al usuario un mecanismo de control a la hora de recargar su vehículo eléctrico y, además, le permite ahorrar dinero en el proceso. En concreto, según nuestras estimaciones, la diferencia de usar el modo de emergencia frente al modo manual puede suponer un 7,5% de ahorro, mientras que usar el modo ahorro permite ahorrar hasta un 21,8% del precio de la energía. Estos datos demuestran que el uso de nuestro enchufe va a ser interesante para los usuarios, además de que puede animarles definitivamente a la hora de adquirir un vehículo eléctrico.

Palabras clave: enchufe inteligente, vehículo eléctrico, tarifa eléctrica, Arduino, Android.

"Implementation of a smart plug for electric vehicles based on Arduino"

ABSTRACT

On this end-of-degree Project, the design and implementation of a smart plug that can be controlled by a user via a mobile application has been realized.

The smart plug has 3 different modes, (i) the manual mode, which turns the plug on, and so the battery recharge process immediately starts, (ii) the emergency mode, where the plug is activated until a certain percentage of battery is reached and then it waits until the price of the electricity is cheaper, and (iii) the energy saving mode through which, with the help of a server, the plug will wait to start the recharge process until the cost of the recharge will be the lowest.

The use of this smart plug, gives the user a way of controlling the recharge of the electric vehicle and, allows the user to save money in the process. Specifically, according to our estimations, the difference between using the emergency mode and the manual mode, can suppose a 7,5% of saving, while using the saving mode can save until a 21,8% of the recharge cost. These data prove that the use of our plug is going to be interesting to the users, even encouraging them to acquire an electric vehicle.

Keywords: smart plug, electric vehicle, electric rate, Arduino, Android.

Tabla de Contenido

Índice de figuras

1. Introducción

Uno de los grandes problemas de hoy en día es el aumento del efecto invernadero que está provocando un cambio climático, haciendo que aumente la temperatura del planeta. Este aumento está provocado fundamentalmente por la emisión de ciertos gases contaminantes como son el dióxido de carbono, metano y óxidos nitrosos.

Los principales generadores de metano y óxido nitroso son los vehículos de combustión interna. Esto hace que, sumado a la concienciación de la gente sobre el cuidado del medio ambiente, se busque otro tipo de combustible para los vehículos, como es la electricidad.

Además de ser menos contaminante, la electricidad tiene un coste menor al de la gasolina o el diésel [1]. Sin olvidar que, al ser el petróleo una materia prima limitada, usar la electricidad como combustible en los vehículos contribuye a la conservación de esta materia.

El número de ventas de vehículos eléctricos crece año tras año, mostrando un crecimiento prácticamente exponencial, como se ve reflejado en la Figura 1. Si las ventas continúan con esta tendencia, es muy probable que, en un futuro no muy lejano, los vehículos eléctricos superen en número a los de combustión interna.

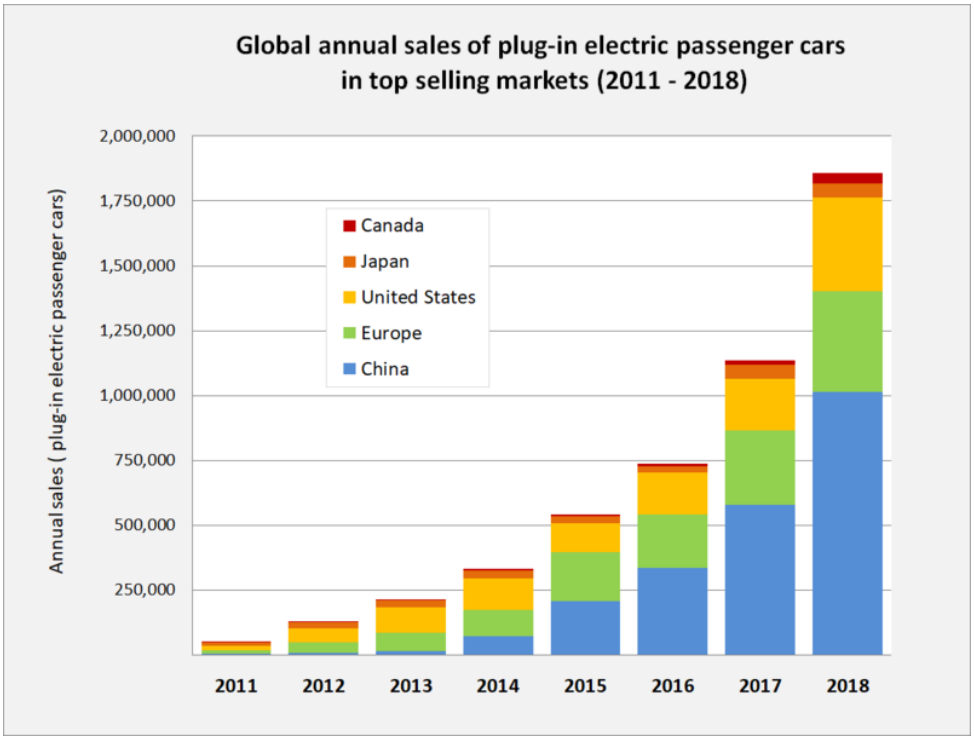

Figura 1. Ventas anuales de vehículos eléctricos por países [2]

1.1 Motivación

En España, el precio de la electricidad es uno de los más elevados de todo el mundo. No obstante, existen diferentes tarifas con discriminaciones horarias y el precio varía en función del día y la hora [3].

Teniendo en cuenta lo anterior, en el presente documento se va a diseñar e implementar un enchufe inteligente para optimizar las recargas de los vehículos eléctricos. Haciendo uso de este enchufe, se consigue optimizar el proceso de recarga de un vehículo eléctrico de manera que se pueda ahorrar dinero en el proceso de recarga. Además, para adaptarse a las necesidades del usuario, se contempla la situación en la que se necesite recargar el vehículo inmediatamente sin tener en cuenta el coste de la electricidad en ese momento.

Como motivación personal, la realización de este trabajo me permite ampliar mis conocimientos en el desarrollo de aplicaciones Android y la programación de placas Arduino. Por otro lado, también adquiriré conocimiento sobre vehículos eléctricos y las diferentes tarifas eléctricas en España.

1.2 Objetivos

El objetivo principal de este proyecto es la creación de un enchufe inteligente que permita ahorrar dinero en el proceso de recarga de un vehículo eléctrico.

Para la consecución de este objetivo, se ha desarrollado tanto una aplicación Android como la programación de una placa Arduino. Los detalles de ambas partes serán explicados más adelante en el presente documento.

Los objetivos secundarios de este trabajo son:

- Conocimiento de la historia y el estado actual de los vehículos eléctricos
- Explicar los diferentes modos existentes en el proceso de recarga de vehículos eléctricos
- Diferenciar los diferentes tipos de tarifas eléctricas en España
- Expandir conocimientos sobre el desarrollo de aplicaciones Android y programación de placas Arduino

.

1.3 Organización de la memoria

El presente documento consta de los siguientes apartados:

- Marco teórico: en esta sección se hará una introducción al sistema Android, realizando una breve explicación de su funcionamiento y arquitectura. También se explicará el funcionamiento de las placas Arduino.
- Estado del arte: en esta sección se realizará una breve presentación de la historia de los vehículos eléctricos y cuál es su estado actual. Además, se expondrán los diferentes tipos de tarifas eléctricas existentes en España.
- Análisis: en esta sección aparece la planificación del proyecto, la especificación de los requisitos a cumplir por el enchufe inteligente, una estimación económica y temporal de la realización del enchufe, una definición de interfaces de la aplicación móvil, los casos de uso que definirán las funcionalidades de las que dispondrá el usuario, una definición de prototipos y la especificación del plan de pruebas.
- Diseño: en esta sección se presentan los diagramas de clases y de despliegue utilizados posteriormente en la implementación.
- Implementación: en esta sección se explican los diferentes procesos llevados a cabo en la realización del enchufe, como son el montaje de las diferentes piezas que formarán el enchufe inteligente, la creación de la aplicación móvil y la programación de la placa Arduino.
- Resultados: en esta sección se desarrolla un ejemplo práctico de cómo resultaría cargar un vehículo eléctrico en cada uno de los modos disponibles, viendo así las horas de inicio y final del proceso de recarga, el coste de la recarga y el ahorro obtenido con el uso de este enchufe.
- Conclusiones: en esta sección se detallan las conclusiones obtenidas con la realización de este Trabajo de Fin de Grado.
- Bibliografía: en esta sección se incluyen las referencias bibliográficas utilizadas a lo largo de la realización del presente documento.

2. Marco teórico

Antes de la realización de este Trabajo de Fin de Grado, se ha realizado una investigación acerca del sistema operativo Android y las placas Arduino, ya que será necesario aprender ciertos conceptos para el correcto desarrollo de este trabajo.

2.1 Android

Android es un sistema operativo, inicialmente diseñado para teléfonos móviles. Sin embargo, hoy en día se utiliza en otros dispositivos como puede ser una Tablet, una televisión o incluso pulseras de actividad o relojes inteligentes.

La característica que diferencia a Android de otros sistemas operativos para teléfonos móviles es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma [4].

Hoy en día, Android es, con mucha diferencia, el sistema operativo más usado en dispositivos móviles, como se puede ver en la Figura 2.

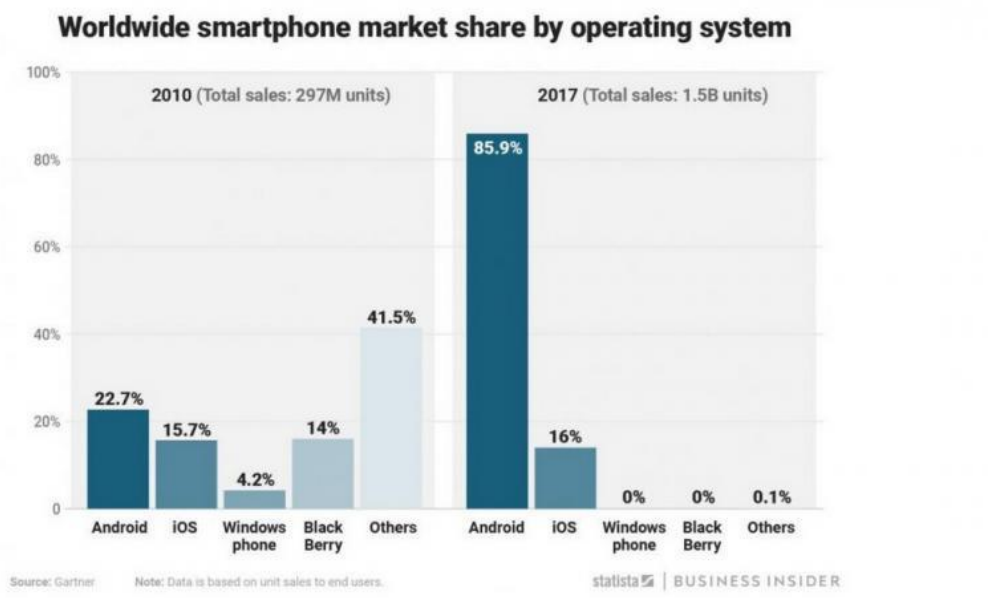

Figura 2. Uso de smartphones por sistema operativo [5]

En cuanto a arquitectura, Android es un sistema monolítico, es decir, únicamente dispone de un núcleo grande que engloba todos los servicios del sistema operativo. Esto le permite tener un mayor rendimiento, sin embargo, también implica que cualquier cambio en alguno de los servicios requiere la recopilación del núcleo y reiniciar el sistema para que los cambios sean aplicados.

Android organiza sus componentes en capas donde cada uno de los diferentes componentes de cada capa, necesita elementos de la capa anterior para realizar sus funciones. Los diferentes componentes de Android se ilustran en la siguiente figura:

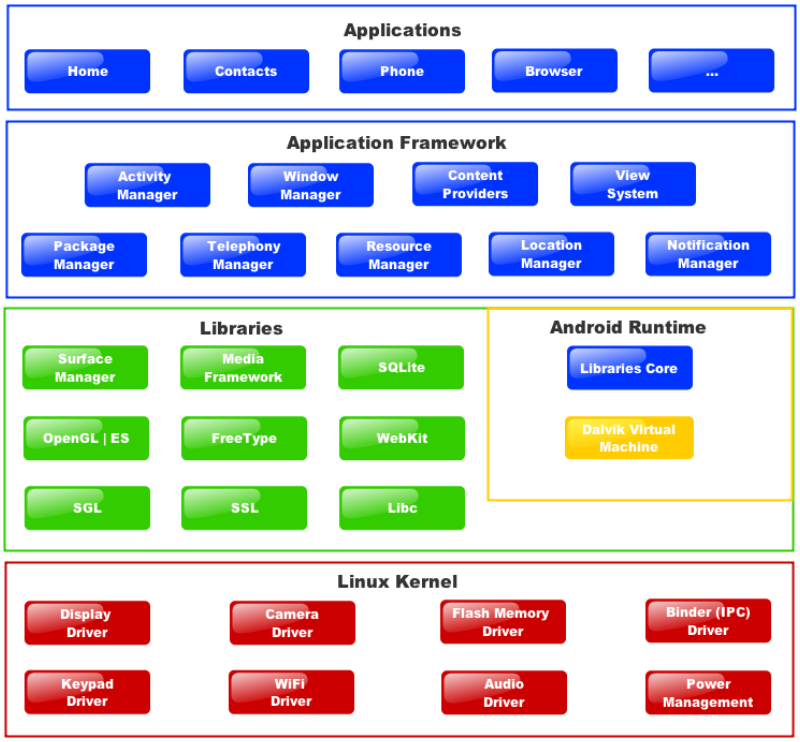

Figura 3. Componentes de Android separados por capas [6]

El kernel de Linux se encarga de proporcionar una capa de abstracción para los elementos hardware a los que acceden las aplicaciones. Este kernel varía en función de la versión de Android.

La capa de librerías se sitúa encima del kernel y es donde se encuentran las librerías nativas de Android. La función de esta capa es dotar de funcionalidad a las aplicaciones de forma eficiente.

La capa de entorno de ejecución se sitúa al mismo nivel que la capa de librerías. El principal componente de esta capa es la máquina virtual Dalvik, que se encarga de ejecutar todas las librerías no nativas de Android.

La capa de marco de aplicación está formada por todos los servicios que son utilizados directamente por las aplicaciones. La mayoría de estos servicios son librerías Java que acceden a los recursos de capas anteriores mediante la máquina virtual Dalvik,

Finalmente, la capa de aplicaciones, que se encuentra en el nivel más alto, es donde se encuentran todas las aplicaciones instaladas en el dispositivo. Las aplicaciones normalmente están programadas en lenguaje Java y, además, utilizan el metalenguaje XML para el desarrollo de las interfaces.

En este trabajo, se va a utilizar Android para la programación de la aplicación que permita gestionar los distinto modos de recarga desde el móvil del usuario.

2.2 Arduino

Arduino es una plataforma de creación de electrónica de código abierto basada en hardware y software libre. Las placas Arduino permiten conectar diferentes periféricos a las entradas y salidas de un microcontrolador, que puede ser controlado tanto en Windows, como macOS y Linux. [7]

Al ser una plataforma de software y hardware libre, existe una comunidad que proporciona apoyo a la hora de desarrollar aplicaciones. Esto se realiza compartiendo librerías y publicando proyectos enteros para que puedan ser replicados, mejorados o servir de apoyo en nuevos proyectos.

Arduino dispone de diferentes modelos de placas y shields. Un shield es una placa compatible que se coloca en la parte superior de una placa Arduino y permite extender las capacidades de la placa. Unos ejemplos de modelos de placas Arduino son Uno, Due, Mega, Ethernet o Leonardo, mientras que algunos de los shields son Ethernet Shield, Módulo Bluetooth HC-06, Arduino Motor Shield o Arduino WiFi Shield 101.

Arduino posee su propio entorno de desarrollo el cual implementa el lenguaje de programación de Arduino y el gestor de arranque. Una de las principales características del software de programación y del lenguaje de programación es su sencillez y facilidad de uso. El entorno de desarrollo proporciona, además, las herramientas para cargar el programa ya compilado en la memoria flash del hardware. Los ficheros de programación de placas Arduino reciben el nombre de sketches.

El lenguaje de programación de Arduino está basado en C++ y, por tanto, es posible usar comandos estándar de dicho lenguaje.

El microcontrolador de Arduino posee una interfaz de entrada, que es una conexión que permite conectar diferentes tipos de periféricos, como pueden ser sensores de humedad o de movimiento, un teclado, etc. La información de estos periféricos se traslada al microcontrolador y éste se encargará de procesar los datos recibidos.

También dispone de una interfaz de salida que es la encargada de llevar la información procesada a otros periféricos. Estos periféricos pueden ser una pantalla LCD, un altavoz o incluso otra placa Arduino.

En este trabajo, se implementará el prototipo del enchufe inteligente utilizando una placa Arduino UNO. De esta forma, podemos tener un enchufe “inteligente” que adapte las recargas a los periodos cuando la electricidad sea más barata. Este dispositivo es de bajo coste, pero nos proporciona toda la potencia que necesitamos para nuestra propuesta.

3. Estado del arte

En esta sección se repasará brevemente la historia de los vehículos eléctricos y su estado actual, explicando los diferentes modos de recarga existentes y cómo han cambiado el material de sus baterías a lo largo del tiempo. También se mostrarán las diferentes tarifas eléctricas que se disponen actualmente en España.

3.1 Vehículos eléctricos

Los primeros vehículos eléctricos surgieron a mediados del siglo XIX y rápidamente emergió un mercado alrededor de ellos en gran parte de Europa y Estados Unidos. No obstante, no fue hasta 1880 cuando aparecieron los primeros vehículos eléctricos prácticos.

Estos vehículos sufrieron un declive debido a los avances en el motor de combustión interna y un decremento en el precio de la gasolina, así como la aparición de un nuevo sistema de arranque eléctrico que sustituyo al manual [8].

Durante la crisis del petróleo en los años 70, los vehículos eléctricos volvieron a ganar popularidad, pero finalmente tampoco consiguieron triunfar.

Actualmente los vehículos eléctricos están en pleno auge debido a diferentes factores como son el descenso en su precio de venta, la concienciación de la gente sobre el cuidado del medio ambiente, que la electricidad es más barata que el diésel o la gasolina, o que además son extremadamente silenciosos. Por esto, varios gobiernos están incentivando su compra y uso estableciendo ayudas para construir más puntos de recarga e incluso eximiéndolos de ciertos impuestos.

Los vehículos eléctricos cuentan con una amplia lista de ventajas sobre los vehículos de combustión interna, por ejemplo, no producen ningún tipo de contaminación durante su circulación y el mantenimiento es mucho menor al de un vehículo convencional.

Sin embargo, también poseen ciertas desventajas, las cuales están relacionadas con la batería, como son el tiempo de recarga y la autonomía de ésta. También hay que tener en cuenta que, aunque no se genere contaminación en su conducción, sí se genera en el proceso de recarga, aunque es mucho menor que la que generan los motores de combustión. Finalmente, su precio sigue siendo elevado, aunque se espera que siga disminuyendo con el tiempo.

Además, hoy en día no hay un gran número de redes públicas de recarga o electrolineras, no obstante, se están empezando a implantar en diferentes ciudades a lo largo de todo el mundo. Dichas electrolineras se dividen en dos, las que permiten recargar las baterías y las que cambian las baterías [9].

Actualmente, en España se dispone de 4545 de puntos de recarga públicos, los cuales se encuentran principalmente en Madrid (552) y Barcelona (722). No obstante, se espera que, en 2020, España cuente con unas 11.000 electrolineras en funcionamiento [10].

Uno de los cargadores más rápidos es el super cargador de Tesla, aunque esté cargador solo funciona para los vehículos de dicha marca. En España, únicamente se dispone de 26 super

cargadores, sin embargo, se espera que más super cargadores sean implantados en un futuro no muy lejano [11]. Este súper cargador permite cargar un 80% del porcentaje de la batería en tan solo media hora.

El tipo de las baterías ha ido variando con el tiempo en busca de un mayor rendimiento. Inicialmente fueron de plomo-ácido, les sucedieron las de níquel y actualmente las más utilizadas son de ion-litio. Se está investigando sobre nuevos tipos de batería que solucionen los problemas de autonomía y tiempo de recarga. En concreto, las baterías de aluminio-aire o de zinc-aire aumentarían la autonomía, requiriendo un menor tiempo de recarga [12].

En cuanto al proceso de recarga, el estándar IEC 62196 ofrece los siguientes modos [13]:

- Modo 1 Modo estándar de carga de uso doméstico. Implica la recarga de baterías de 230 V y un máximo de corriente de 16 A. El proceso de recarga puede durar hasta 8 horas, por lo que es considerado un modo de recarga lento. Se usa fundamentalmente en bicicletas y patinetes eléctricos. Este modo carece de comunicación entre ambos puntos y, por tanto, de seguridad.
- Modo 2 Este modo de carga soporta corrientes de hasta 32 A. En función del vehículo, el proceso de recarga puede durar entre 2 y 4 horas. Este modo es considerado semirrápido. En este caso, el vehículo también se conecta a la red doméstica, pero es necesario un transformador, ya que utiliza corriente alterna. Este modo cuenta con un método de protección mediante el cual se puede verificar si se está conectado correctamente a la red, activar o desactivar la recarga y elegir la velocidad de ésta.
- Modo 3 Es considerado el modo de carga rápida. Soporta corrientes de entre 32 A y 250 A. Dependiendo del vehículo, el proceso de recarga puede durar menos de 1 hora. Este modo es más seguro que el anterior ya que el transformador se encuentra en los denominados wallbox.
- Modo 4 Denominado modo de carga super rápida. Necesita de conectores especiales como son el CHADEMO en Europa y el CSS Combo en Estados Unidos y Japón soportando voltajes de hasta 400 A. Dependiendo del vehículo, el proceso de recarga puede durar menos de 30 minutos.

3.2 Tarifas eléctricas

El precio de la luz varía dependiendo de la tarifa contratada, por eso, es un factor muy importante en el proceso de recarga.

En España existen diferentes tipos de tarifas en función de la tensión que ofrezcan, la potencia contratada, o la discriminación horaria. Dependiendo de la discriminación horaria, las tarifas pueden ser: (i)la tarifa normal, donde no hay discriminación horaria y se establece un precio en €/KWH fijo para las 24 horas del día los 365 días del año; (ii) la tarifa con discriminación horaria de dos periodos, que tiene un precio más caro para las horas punta y un precio más bajo para las horas valle y (iii) la tarifa con discriminación horaria de tres periodos, que ofrece 3 precios distintos según el día y la hora de la semana distinguiendo así precio punta, precio llano y precio valle, de mayor a menor importe.

A continuación, se ilustran los diferentes tipos de tarifas eléctricas:

Tarifa	Tensión	Potencia	Nº Horas			Tipo tarifa
			Punta	Llano	Valle	
2.0A	<=1KV	<=10kW	24	0	0	Baja tensión
2.0DHA			10	0	14	
2.0DHS			10	8	6	
2.1A		10kW > P <15kW	24	0	0	
2.1DHA			10	0	14	
2.1DHS			10	8	6	
3.0A		>=15kW	4	12	8	
3.1A	> 1KV	15kW>=P<=450kW	6	10	8	Alta tensión

Figura 4. Tipos de tarifas eléctricas [14]

Una posible forma de ahorro en el proceso de recarga del vehículo eléctrico es aquélla en la que, en función de la tarifa contrada y del porcentaje de batería, se empezase el proceso de recarga en las horas valle y finalizara cuando el nivel de carga de la batería esté al máximo posible. Este método es el que se ha implementado en el modo ahorro del enchufe inteligente siendo un servidor externo, que no ha sido desarrollado por el autor de este Trabajo de Fin de Grado, el que se encarga de calcular dicho periodo de tiempo.

4. Análisis

En esta sección se detalla la fase de análisis de este trabajo de fin de grado. Aparecen la planificación del proyecto, una descripción conceptual del trabajo, la especificación de requisitos, la definición de interfaces, la definición de prototipos, el diagrama de casos de uso, la especificación del plan de pruebas y una estimación económica de la realización del enchufe inteligente.

4.1 Planificación

En primer lugar, se ha realizado una planificación para tener una idea de la duración de este proyecto. Al ser tareas que ya he realizado previamente, se puede hacer una estimación del tiempo que puede llevar su realización. Para ello se ha utilizado un diagrama de Gantt haciendo uso de la herramienta GanttProject.

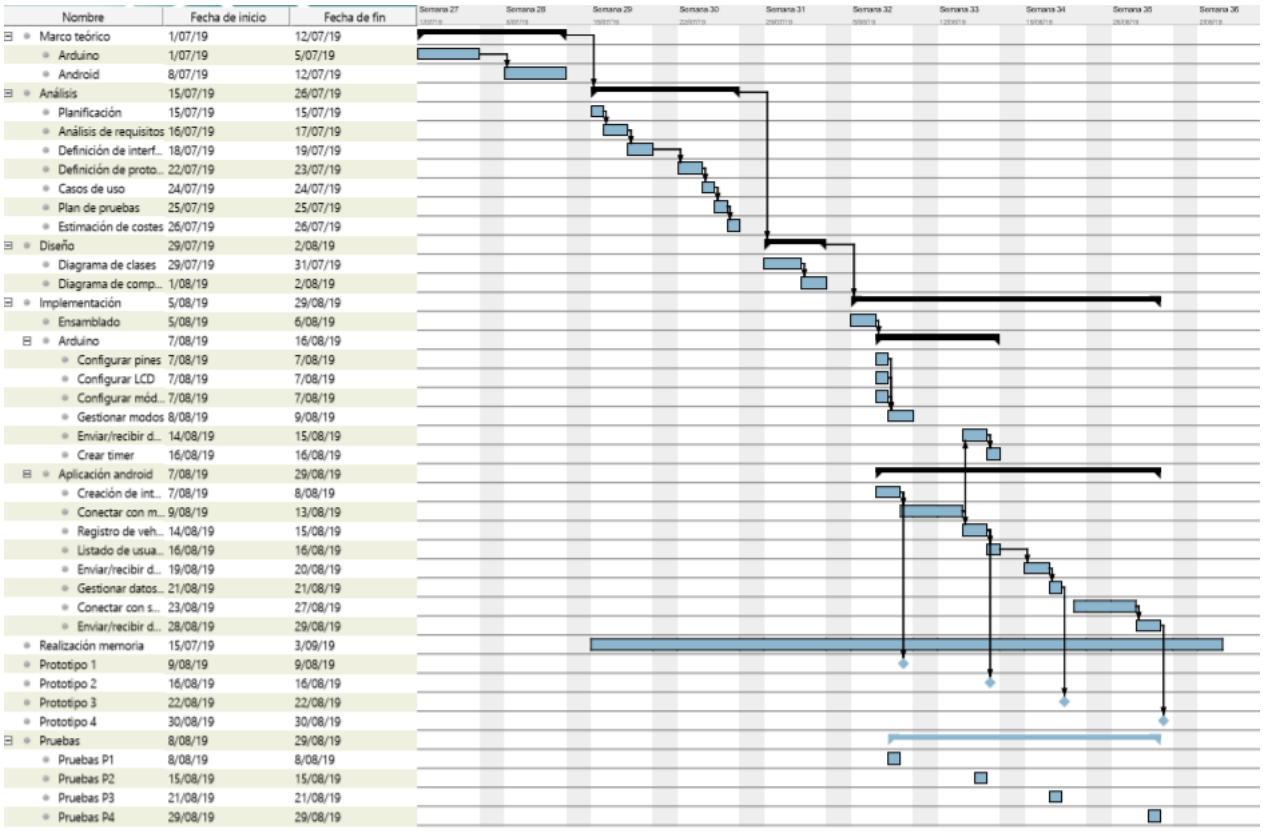

Figura 5. Planificación inicial del proyecto

Realizar una planificación de este modo permite ver si hay variaciones reales a la hora de realizar las tareas y sirve de indicativo para realizar futuras planificaciones, ya que puede servir de referencia.

La fecha de inicio de este proyecto ha sido el 1 de julio de 2019 y la fecha de finalización el 3 de septiembre de 2019.

Este trabajo se puede dividir en 6 grandes tareas:

- Marco teórico
- Análisis
- Diseño
- Implementación
- Pruebas
- Realización de la memoria

Dentro del marco teórico, aparece tanto la investigación acerca de Android y programación de aplicaciones Android, como la investigación de Arduino y la programación de sus placas. Esto se debe a que son temas que no se tratan muy a fondo a lo largo de la carrera y es necesario conocer ciertos detalles antes de empezar con la programación.

En el apartado de análisis, aparece en primer lugar la planificación estimada, que hace referencia al diagrama de Gantt de la Figura 5, la definición de interfaces de la aplicación Android, la definición

de los prototipos a desarrollar, la especificación de los requisitos que debe cumplir el enchufe inteligente, tanto funcionales como no funcionales, los casos de uso que representarán las diferentes funcionalidades que el usuario dispone con este enchufe inteligente y una estimación del coste de realizar este trabajo, tanto de los materiales como de la mano de obra.

En cuanto al diseño, se definen todos aquellos documentos necesarios antes de la implementación. Para este trabajo, se han desarrollado tanto un diagrama de clases como un diagrama de componentes.

En la implementación se pueden diferenciar 3 partes: (i) el ensamblado del prototipo, (ii) la programación de la placa Arduino y (iii) la programación de la aplicación Android. Cada una de ellas se subdivide en tareas más concretas que permiten una implementación por niveles y restan complejidad a la misma. Más adelante, en el apartado de implementación, serán explicadas con mayor nivel de detalle.

Por otra parte, aparece la realización de pruebas, para verificar el correcto funcionamiento de cada uno de los prototipos definidos previamente.

Finalmente, la realización de la memoria consiste en la redacción del presente documento que se ha ido realizando prácticamente a lo largo de todo el proyecto.

Además de estas 6 tareas, aparecen unos hitos que corresponden a las fechas en las que tendrán que estar acabados los diferentes prototipos.

4.2 Descripción conceptual

Para el desarrollo este enchufe inteligente, se ha utilizado una placa Arduino UNO, una placa de pruebas para el cableado, un módulo bluetooth HC-06 y una pantalla LCD donde se muestra el modo actual del enchufe. Se disponen de 3 modos diferentes:

- Modo manual Este modo se subdivide en dos estados, on y off.
  - Estado Off El enchufe no se encuentra activo.
  - Estado On El enchufe se encuentra activo y, por tanto, el vehículo está cargando su batería en caso de no estar cargada completamente.
- Modo de emergencia El enchufe se encuentra activo durante un periodo de tiempo hasta que se alcance un determinado porcentaje de batería (en este caso el 60%). La recarga del porcentaje de batería restante se realizará cuando la electricidad sea más barata.
- Modo ahorro El enchufe se encuentra desactivado hasta que el coste de la recarga sea lo más económico posible. Justo en ese momento, se activa hasta finalizar la recarga.

Para permitir al usuario conectarse con el enchufe y cambiar de modo, se ha desarrollado una aplicación Android. Esta aplicación solicita al usuario, mediante un formulario, su nombre, modelo de vehículo, capacidad de la batería, porcentaje de batería restante, hora de salida y días hasta que se efectúe la salida. Para agilizar este proceso, la aplicación permite al usuario registrar vehículos, con un nombre de usuario (dueño del vehículo), modelo de vehículo y capacidad de batería del vehículo, para poder seleccionar posteriormente dicho vehículo en la lista de vehículos registrados y permitir que la aplicación pueda introducir estos datos en el formulario automáticamente.

Además, por defecto la aplicación introducirá un 1 en el campo de los días hasta la salida, ya que lo más usual es cargar el vehículo para utilizarlo al día siguiente.

Una vez rellenado el formulario, los datos son enviados a un servidor externo, el cual calcula el número de horas que han de transcurrir hasta que comience la recarga para ésta sea lo más económica posible en función del contexto.

4.3 Requisitos

En este apartado se definen los requisitos que se deben de cumplir. Los requisitos están divididos en dos grupos, funcionales y no funcionales:

4.3.1 Funcionales

RQ F 1. Se deberá desarrollar una aplicación móvil para el control del enchufe.

RQ F 2. El usuario se podrá conectar con la placa Arduino haciendo uso de la aplicación móvil que dispondrá de un botón para ello.

RQ F 3. La aplicación deberá permitir al usuario cambiar de modo solicitándole los siguientes datos haciendo uso de un formulario:

a. Nombre de usuario
b. Modelo de vehículo
c. Porcentaje actual de batería
d. Hora de salida
e. Número de días hasta la salida
f. Capacidad de la batería

RQ F 4. La aplicación deberá permitir al usuario registrar un vehículo con un nombre de usuario, modelo de vehículo y capacidad de la batería, así mismo, también podrá seleccionar un vehículo de una lista de vehículos registrados cuyos datos serán usados para agilizar el proceso de inserción de datos en el formulario de cambio de modo.

RQ F 5. El enchufe constará de 3 modos:

a. Modo manual El usuario podrá encender y apagar el enchufe cuando lo necesite.
b. Modo de emergencia El enchufe estará activo durante un periodo determinado de tiempo, pasado dicho tiempo, se desactivará. Finalmente, volverá a activarse cuando el coste de la electricidad sea más bajo para acabar la recarga completa de la batería.
c. Modo ahorro El enchufe empezará a cargar pasado un determinado periodo de tiempo. Dicho tiempo, corresponde al tiempo de espera necesario para que la recarga del vehículo eléctrico sea más barata. La recarga finalizará cuando el vehículo haya cargado completamente su batería.

RQ F 6. El enchufe deberá de contar con una placa LCD en la cual se indiquen el estado y el modo actual. El estado podrá ser on/off y el modo uno de los 3 modos del RQ F 5.

RQ F 7. Para determinar la cantidad de horas que debe esperar el enchufe a activarse en el modo ahorro, la aplicación móvil se conectará con un servidor externo que será el que determine dichas horas de espera para que la recarga sea lo más económica posible en función de los datos introducidos en el formulario de cambio de modo. Cuando el servidor determine la cantidad de horas que ha de estar desactivado, la aplicación móvil las transmitirá a la placa Arduino que, haciendo uso de un temporizador, esperará dicho tiempo, activándose cuando finalice dicha espera.

4.3.2 No funcionales

RQ NF 1. La aplicación móvil deberá ser desarrollada en Android

RQ NF 2. La comunicación entre la aplicación móvil y la placa Arduino deberá de ser vía bluetooth.

RQ NF 3. La aplicación móvil y el servidor se conectarán haciendo uso de un servicio web con el método POST.

RQ NF 4. La conexión bluetooth entre aplicación y placa Arduino se realiza automáticamente al abrir la aplicación y, si ésta falla, el botón para enviar datos al servidor estará desactivado hasta que la conexión se realice correctamente.

RQ NF 5. La aplicación deberá de controlar los datos introducidos por el usuario en el formulario de cambio de modo haciendo que se cumplan las siguientes condiciones:

a. El usuario debe introducir todos los campos.
b. El usuario debe seleccionar obligatoriamente 1 modo.
c. El porcentaje de batería debe ser un número comprendido entre 0 y 100 estando ambos incluidos.
d. La capacidad de la batería no puede ser un número negativo
e. La hora tiene que ser una hora válida y ser introducida con el formato HH:MM.
f. Los días hasta la salida no pueden ser un número negativo.

RQ NF 6. La aplicación deberá de controlar los datos introducidos por el usuario en el formulario de registro de vehículos haciendo que se cumplan las siguientes condiciones:

a. El usuario debe introducir todos los campos
b. La capacidad de la batería no puede ser un número negativo

RQ NF 7. Para la persistencia de los vehículos, se deberá de utilizar el sistema gestor de bases de datos SQLite.

4.4 Definición de interfaces

Antes de empezar con la programación, se definen unos bocetos iniciales sobre los que trabajar con respecto a la parte visual de la aplicación. Las interfaces definidas son las siguientes:

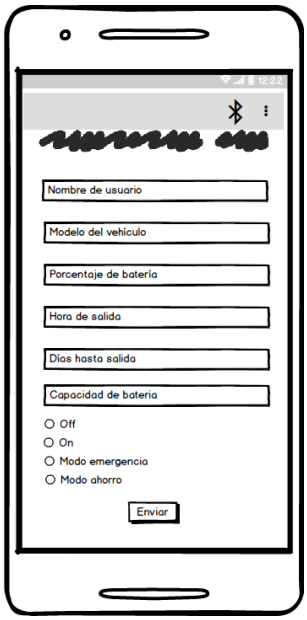

Figura 6. Pantalla inicial, formulario de cambio de modo

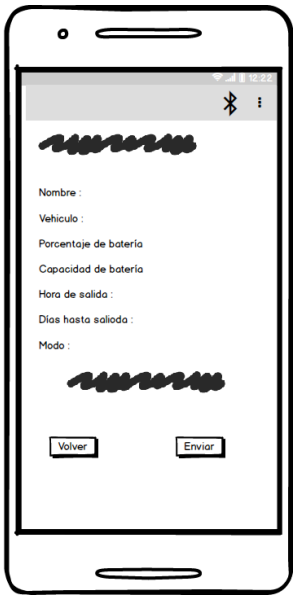

Figura 7. Pantalla de confirmación de cambio de modo

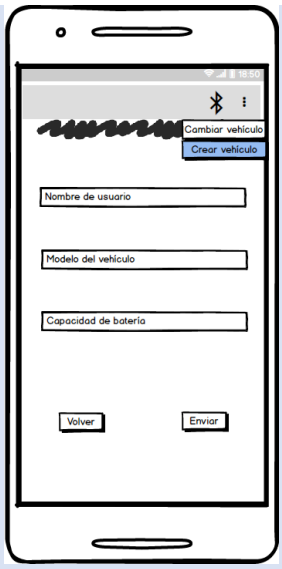

Figura 8. Pantalla de formulario de registro de vehículos

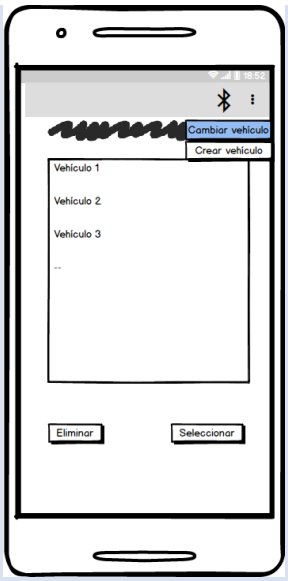

Figura 9. Pantalla de lista de vehículos registrados

Al abrir la aplicación aparece el formulario de cambio de modo ilustrado en la Figura 6. Si no hay ningún vehículo registrado, todos los campos estarán vacíos, pero si hay algún vehículo, la aplicación rellenará los campos con la información de dicho vehículo, agilizando el proceso de inserción de

datos. Además, el campo *Días hasta la salida,* tendrá el valor de 1 por defecto, aunque el usuario puede cambiar el valor de cualquier campo si lo necesita.

Si la conexión con el módulo bluetooth ha fallado, el botón *Enviar* estará desactivado hasta que se conecte correctamente.

Si se pulsa el botón de *Enviar*, entonces aparece la pantalla de confirmación ilustrada en la Figura 7. En esta pantalla, se muestran los datos introducidos en el formulario y dos botones, uno de *Enviar* y otro de *Atrás*. Si se considera que los datos introducidos son correctos, se pulsará el botón de enviar y la petición será procesada, cambiando el modo del Arduino y enviando los datos al servidor si es necesario, pero si se ha introducido mal algún dato o simplemente no se quiere realizar la petición, el botón de atrás hará que se vuelva a la pantalla inicial.

Para volver a realizar la conexión en caso de error, existe un menú en el que aparecen dos botones, uno con el símbolo de bluetooth, el cual es el encargado de volver a intentar la conexión, y otro con el símbolo de tres puntos que, al pulsarlo, aparecen dos opciones, *Crear vehículo* y *Cambiar vehículo*.

Los botones de *Crear vehículo* y *Cambiar usuario* abren otras pantallas diferentes. El primero de ellos abre la pantalla de la Figura 8 en la que aparece otro breve formulario de registro de vehículo. En este formulario, se ha de introducir el nombre de usuario, el modelo de vehículo y la capacidad de la batería. Para realizar el registro correctamente, se deberán de introducir todos los campos ya que serán utilizados posteriormente por la aplicación. El nombre de usuario únicamente será requerido la primera vez que se instale la aplicación. El resto de las veces aparecerá dicho nombre por defecto, ya que se presupone que se dispone de un usuario por cada dispositivo móvil en la que se instale en la aplicación, no obstante, la posibilidad de registrar varios usuarios está presente.

Finalmente, el botón de *Cambiar vehículo* lleva a la pantalla de la Figura 9 en la que simplemente aparece una lista con todos los vehículos registrados en ese momento. Al pulsar sobre alguno de ellos, se dispondrán de dos opciones, una de eliminar dicho vehículo con el botón de *Eliminar*, donde al pulsar dicho botón, aparece un mensaje de confirmación para asegurar que se desea eliminar dicho vehículo, y la otra opción es la de seleccionar dicho vehículo, donde al pulsar el botón *Seleccionar*, se usarán los datos de ese vehículo seleccionado en el formulario y se volverá a la pantalla inicial del formulario de cambio de modo.

4.5 Casos de uso

Los casos de uso permiten modelar el comportamiento de un sistema, por tanto, para este trabajo se ha definido el siguiente diagrama:

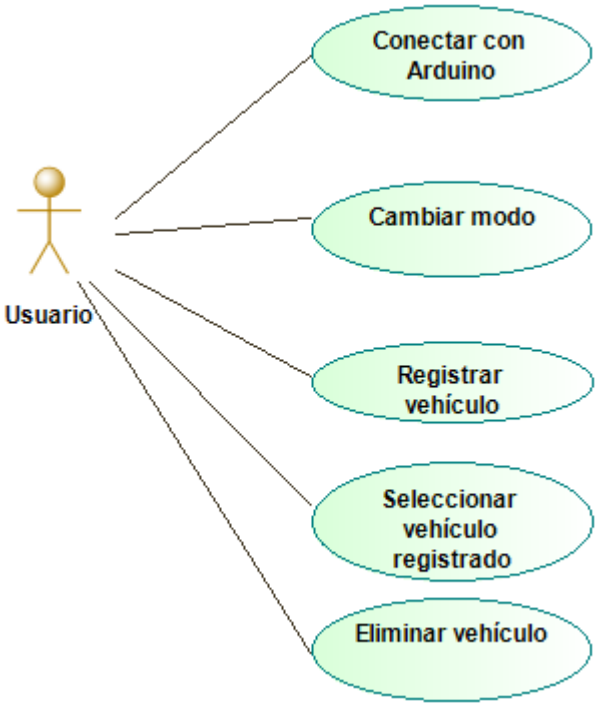

Figura 10. Diagrama de casos de uso

Haciendo uso de la aplicación móvil, el usuario podrá conectar con la placa Arduino, pulsando el botón habilitado para ello en el menú, cambiar de modo el enchufe, rellenando el formulario correspondiente, registrar vehículos haciendo uso de otro formulario para ello, seleccionar un vehículo registrado de una lista, o eliminar un vehículo de dicha lista.

4.6 Definición de prototipos

Para simplificar la realización de este trabajo, se ha dividido la implementación de la aplicación en 4 prototipos. Cada prototipo va adquiriendo funcionalidades de manera incremental. Los prototipos definidos son:

- Prototipo 1 Implementación de las interfaces y navegación entre ellas, sin dotarlas de mayor funcionalidad. Deberá de estar acabado el día 9/08/2019.
- Prototipo 2 Añadir la funcionalidad de conexión con el módulo bluetooth. Además, la activación/desactivación del botón de la pantalla inicial será añadida. También deberá ser posible registrar vehículos. Deberá de estar acabado el día 16/08/2019.
- Prototipo 3 Añadir la comunicación con el módulo bluetooth, así como el procesado de los datos recibidos. También se implementará un mecanismo de control en los formularios siguiendo las condiciones especificadas en los requisitos. Por último, se mostrará la lista de vehículos registrados pudiendo seleccionarse uno de ellos o eliminarlo. Deberá de estar acabado el día 22/08/2019.

- Prototipo 4 Añadir la conexión con el servidor y la comunicación con el mismo. Se realizará también un procesamiento de los datos recibidos por el servidor. Deberá de estar acabado el día 30/08/2019.

4.7 Especificación del plan de pruebas

Cada vez que un prototipo esté terminado, se deberá de realizar una serie de pruebas que verifiquen su correcto funcionamiento. Sin embargo, debido a la simplicidad del prototipo 1, estas pruebas únicamente se realizarán para los prototipos 2, 3 y 4.

Las pruebas contemplarán aspectos tanto de funcionalidad como de usabilidad. Por tanto, se realizarán dos tipos de pruebas:

- Pruebas unitarias: Una prueba unitaria permite verificar el correcto funcionamiento de un módulo de código. Para realizar estas pruebas, se utilizará el propio Android Studio, ya que consta de la funcionalidad necesaria para realizar estas pruebas. Deberán de realizarse pruebas unitarias para cada funcionalidad de la que disponga cada prototipo.

- Pruebas de usabilidad: Este tipo de pruebas determinan si el usuario es capaz de utilizar correctamente la aplicación y permiten conocer si le ha resultado más o menos complejo. Estas pruebas se deberán realizar en dos tipos de usuarios diferentes, uno que posea conocimientos de informática y otro que no los posea, o al menos no tenga grandes conocimientos de ella. Con estas pruebas se busca:
    - Determinar si un usuario es capaz de completar satisfactoriamente todas las funcionalidades de la aplicación.
    - Determinar si las interfaces son lo suficientemente intuitivas.
    - Determinar si la aplicación requiere modificaciones para que se cumplan los objetivos anteriores.

4.8 Estimación económica

Para realizar este proyecto ha sido necesario la siguiente lista de elementos con su coste:

- Enchufe macho y hembra 3€
- Cableado 1€
- Arduino Starter Kit 29€
- Módulo bluetooth HC-06 8€
- Caja de registro estanca 3€

Podría abaratarse el coste suprimiendo el Arduino Starter Kit y comprando únicamente la placa y los componentes necesarios, pero por comodidad, se ha preferido comprar dicho kit.

Por tanto, el coste estimado de los materiales necesarios para este proyecto es de unos 44€.

Además de los materiales, hay que tener en cuenta el coste de desarrollar la aplicación y la programación de la placa Arduino. Para ello, se va a utilizar la técnica de los puntos función (PF) para calcular el esfuerzo que necesita la realización del trabajo.

Los tipos de puntos función para tener en cuenta son:

- Internal Logical Files (ILFs): Conjunto de datos lógicamente relacionados, identificables por el usuario y mantenidos a través de procesos dentro de los límites de la aplicación.
- External Interface Files (EIFs): Conjunto de datos lógicamente relacionados, identificables por el usuario y mantenidos a través de procesos dentro de los límites de la aplicación.
- External Inputs (EIs): Proceso elemental que introduce información en el sistema.
- External Outputs (EOs): Proceso elemental que introduce una salida de información en el sistema.
- External Querys (EQs): Proceso elemental compuesto por una combinación de entrada-salida que obtiene una recuperación de datos internos.

A su vez, los ILFs y los EIFs, se dividen en:

- Data Element Type (DET): Corresponden a campos o atributos de los ficheros, incluyendo claves ajenas.
- Record Element Type (RET): Corresponden a campos del fichero que referencian a otros ficheros.

En primer lugar, los ILFs en este trabajo, serían conexión a internet, conexión bluetooth y la tabla Vehículos de la base de datos.

Los EI encontrados en este trabajo son registrar usuario, cambiar de modo y eliminar vehículo.

En cuanto a EO, aparecen listar vehículos y listar datos de cambio de modo.

Por último, las EQ son seleccionar vehículo y obtener vehículos.

Para cada tipo de punto función, hay que elegir un valor multiplicativo dependiendo del nivel de calificación que se tenga. Se han definido los siguientes valores:

- EI baja x3, media x4, alta x6
- EO baja x4, media x5, alta x7
- EQ baja x3, media x4, alta x6
- ILF baja x7, media x10, alta x15
- EIF baja x5, media x7, alta x10

A continuación, se muestran en una tabla todos los puntos función encontrados en la realización del trabajo, a los que se ha aplicado su correspondiente valor multiplicativo, y el total de puntos función.

Nombre	Tipo	Calificación	PF
Conexión Internet	ILF	Media	10
Conexión Bluetooth	ILF	Media	10
Tabla vehículos	ILF	Baja	7
Registrar vehículo	EI	Baja	3
Cambiar modo	EI	Alta	6
Eliminar vehículo	EI	Baja	3
Listar vehículo	EO	Baja	4
Listar cambio modo	EO	Baja	4
Seleccionar vehículo	EQ	Baja	3
Obtener vehículos	EQ	Baja	3
		PF Totales	**53**

Figura 11. Tabla de PF por tipo y calificación

El siguiente paso es calcular el factor de ajuste. El factor de ajuste es un coeficiente que sirve para adaptar el resultado de los puntos función. Con este factor, se obtienen los puntos función ajustados, para ello, se realiza una valoración de las distintas características del proyecto y se define el valor de ajuste.

El cálculo del factor de ajuste se ha realizado en la siguiente tabla:

Características	Grado de Influencia (0…5)
Comunicaciones de datos	5
Funciones distribuidas	0
Rendimiento	3
Configuraciones fuertemente utilizadas	4
Frecuencia de transacciones	3
Entrada de datos on-line	3
Eficiencia del usuario final	2
Actualizaciones on-line	3
Procesos complejos	4
Reutilización	4
Facilidad de instalación	5
Facilidad de operación	3
Instalación en distintos lugares	0
Facilidad de cambios	4
Grado total de influencia (TDI)	**43**

Figura 12. Valores de ajuste para los puntos función

El valor del factor de ajuste es un coeficiente que sirve para adaptar el resultado de los puntos función obtenidos previamente a las características del sistema diseñado en base a la importancia de éstas. La fórmula es:

VAF = (TDI * 0,01) + 0,65 = (43* 0,01) + 0,65 = 1,08

A partir del factor de ajuste, se puede hacer el calculo de los puntos función ajustados:

AFP = PFTotales * VAF = 53*1,08 = 57,24 = 56

Para poder calcular cuánto tiempo estimado se necesita para calcular esta aplicación, es necesario conocer la productividad. La productividad estándar, utilizando lenguajes de alto nivel, es de 2PF por persona-día [15]. Por tanto, el coste temporal sería

Coste Temporal = PFA * Productividad = 57 * 2 = 114 PF/persona-día.

Suponiendo que el sueldo medio de un ingeniero sea de 65€ por día, el coste total de desarrollar este proyecto sería de:

65 * 114 = 7.410€

A esta cantidad, hay que sumarle los costes de los materiales calculados anteriormente (44€), el importe total de desarrollar este trabajo sería de 7.454€.

5. Diseño

En esta sección se definen los diagramas que serán utilizados posteriormente en el desarrollo de la aplicación. En este caso, se ha creado un diagrama de clases y un diagrama de componentes.

5.1 Diagrama de clases

En la realización de este diagrama, se han omitido las partes que hacen referencia a la interfaz de usuario, únicamente se representan aquellas clases que dan funcionalidad a la aplicación. En la siguiente figura aparece el diagrama de clases:

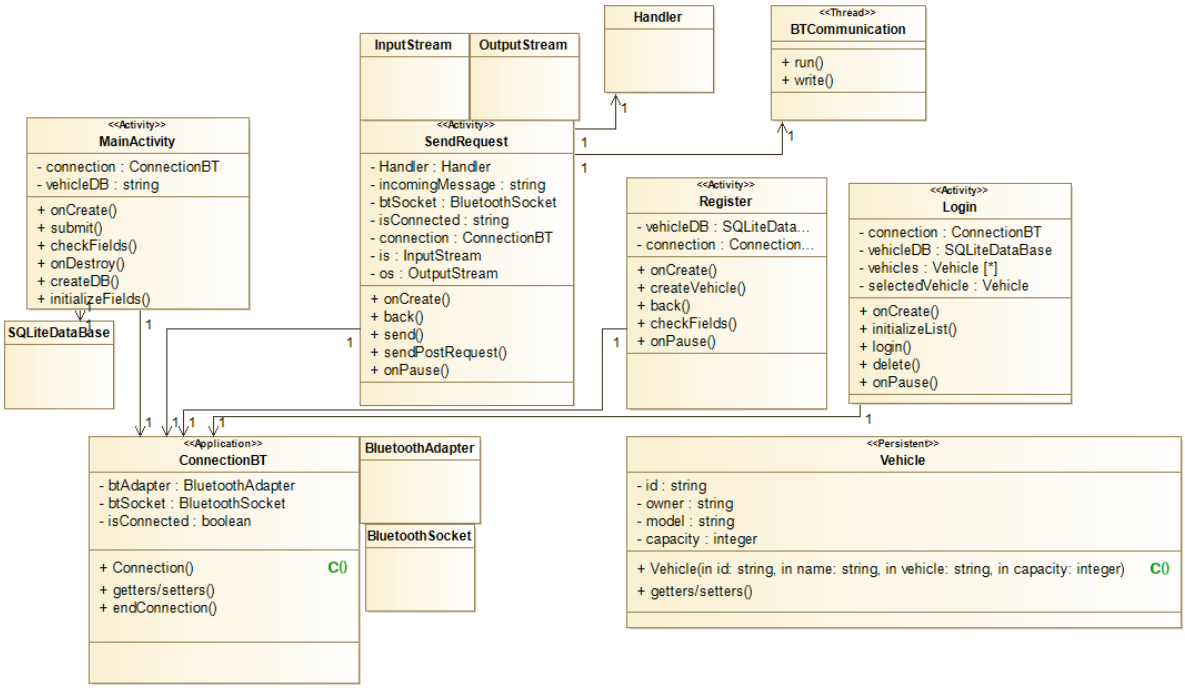

Figura 13. Diagrama de clases

Se define una clase para cada una de las diferentes actividades que tendrá la aplicación. Cada una de ellas, dispone del método onCreate() donde se inicializan las variables y se realizan las acciones necesarias para crear dicha actividad.

La clase MainActivity hace referencia a la actividad principal que aparece al abrir la aplicación ilustrada en la Figura 6. Esta clase controla que todos los datos hayan sido introducidos correctamente en el formulario con el método checkFields(), crea o abre la base de datos con createDB(), inicializa los datos del formulario con los datos del primer vehículo almacenado en la base de datos, si es que lo hay, con initializeFields() y con el método submit(), si los datos han sido introducidos correctamente, se envían a la siguiente actividad, que es SendRequest. Además se dispone del método onDestroy() que se encarga de cerrar la base de datos y la conexión con el modulo bluetooth al salir de la aplicación.

La actividad SendRequest, ilustrada en la Figura 7, simplemente se encarga de mostrar los datos introducidos en el formulario, a modo de confirmación. Tiene los métodos back(), que sirve para

volver a la actividad anterior, send() que se encarga de mandar los datos al módulo bluetooth y, si se ha seleccionado el modo ahorro, realiza una llamada al método sendPostRequest(), que realiza una petición HTTP al servidor web. El servidor le responde con un número de horas, y el método se las envía al módulo bluetooth. Para realizar la comunicación con el módulo bluetooth, esta clase crea un thread de la clase BTCommunication que posee dos métodos, uno para escribir en el módulo, write(), y el método run() que está a la espera de que el módulo le envíe la confirmación del cambio de modo. Cuando reciba la confirmación, haciendo uso de un Handler, muestra un mensaje en la pantalla de confirmación de cambio de modo. Finalmente está el método onPause() que únicamente finaliza la actividad.

La actividad Register, cuya apariencia se muestra en la Figura 8, es lanzada cuando se pulsa el botón *Registrar Vehículo* en el menú de aplicación. En esta actividad se disponen de los métodos back(), que finaliza la actividad volviendo a la anterior, checkFields(), que comprueba que los datos han sido introducidos correctamente en el formulario, y createVehicle() que recoge los datos del formulario y crea un nuevo vehículo en la base de datos. El método onPause() finaliza la actividad.

La clase Vehicle representa la forma en la que los vehículos son guardados en la base de datos. Únicamente posee un constructor, los métodos getter y setter para cada atributo y sobrescribe el método toString().

La actividad Login, ilustrada en la Figura 9, posee los métodos initializeList(), que se encarga de rellenar la lista con los vehículos existentes en la base de datos, el método login(), que permite usar los datos del vehículo seleccionado en el formulario de cambio de modo, y el método delete(), que permite eliminar el vehículo seleccionado. Como anteriormente, aparece el método onPause que finaliza la actividad.

Por último, la clase ConnectionBT, es una clase que extiende de la clase predefinida Application. Esto significa que será creada antes que cualquier actividad y que podrá ser usada en cualquiera de ellas realizando una llamada al método getApplication(). Esta clase se encarga de iniciar la conexión entre la aplicación y el módulo bluetooth en el método onCreate(). También posee un método endConnection() que se encarga de finalizar la conexión entre ambas partes y, por último, los métodos getter y setter.

5.2 Diagrama de componentes

Para desarrollar este diagrama se ha utilizado la herramienta online https://online.visual-paradigm.com/es/

Con el diagrama de componentes se muestran los diferentes componentes que forman el sistema y sus dependencias. Se diferencian 5 componentes: (i) la interfaz gráfica de la aplicación móvil, (ii) el hardware, (iii) la base de datos del sistema operativo Android, (iv) el servidor y (v) el módulo bluetooth del Arduino Uno.

La siguiente figura ilustra los diferentes componentes:

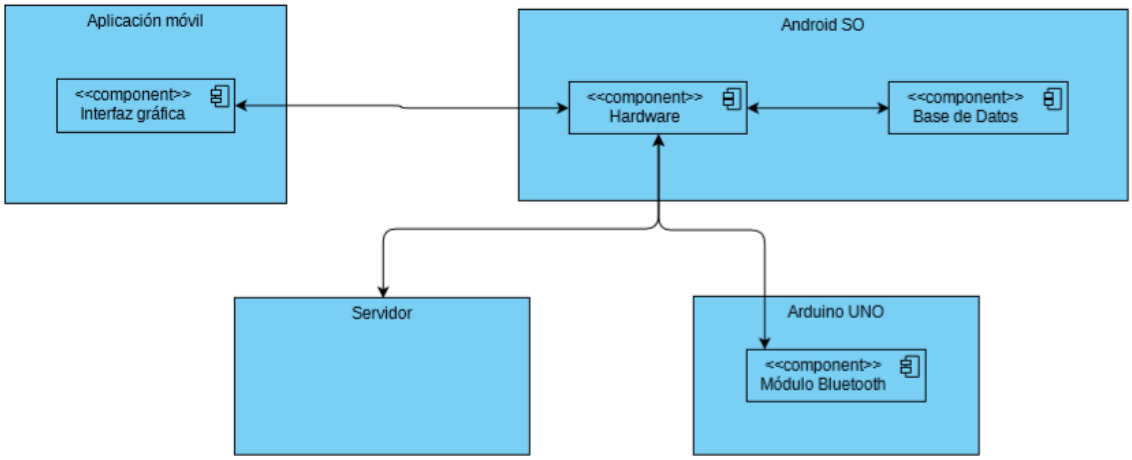

Figura 14. Diagrama de componentes

La interfaz gráfica se comunica con el hardware. Por ejemplo, cuando el usuario pulsa un botón, le indica al hardware del dispositivo Android que se tiene que realizar cierta acción. Además, el hardware también se comunica con la interfaz, ya que puede indicarle que muestre o deje de mostrar ciertos elementos.

A su vez, el hardware y la base de datos se comunican cuando el usuario quiere registrarse o cuando se muestra la lista de usuarios registrados.

Por otro lado, el hardware realiza una petición al servidor y éste le responde con cuántas horas tiene que estar inactivo el enchufe, activándose cuando transcurra ese periodo de tiempo. De esta manera, se produce un ahorro económico, ya que el enchufe se activará cuando el servidor estime que el coste de la recarga va a ser menor.

Además, el hardware del dispositivo Android también se comunica con la placa Arduino. En concreto, le comunica el nuevo modo al que tiene que cambiar y la placa le responde con un OK en caso de que el cambio de modo se haya producido.

6. Implementación

En esta sección se explicará con mayor detalle las partes de la implementación del enchufe. Dichas partes corresponden al ensamblado de los componentes que forman el enchufe, la programación de la aplicación Android y la programación de la placa Arduino.

6.1 Ensamblado

Para la realización del enchufe será necesario:

- 1 Arduino UNO [7]
- 1 modulo Bluetooth HC-06 [16]
- 1 placa de pruebas
- 1 cable de manguera con toma de tierra
- 1 hilo de cable sin toma de tierra.
- 1 enchufe hembra y 1 macho
- 1 potenciómetro de 10K ohmios
- 1 resistencia de 200 ohmios
- 1 relé

Para realizar el boceto del circuito se ha utilizado la herramienta Fritzing [17]. Este programa permite crear esquemas eléctricos y conectarlos con una placa Arduino. Sin embargo, no tiene definidos los componentes de los enchufes, por lo que al circuito que se muestra a continuación, habría que añadirle dichos elementos. Más adelante se realizará una explicación de las conexiones.

A continuación, en la Figura 15, se muestra un esquema de las conexiones:

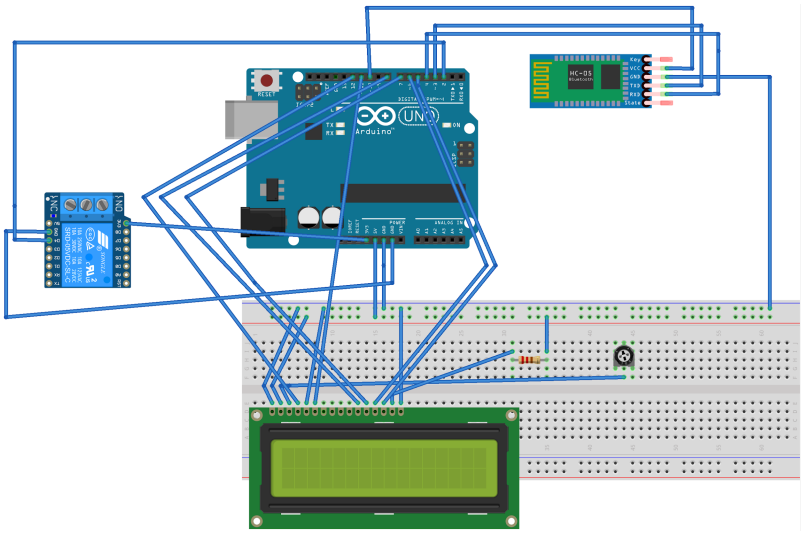

Figura 15. Esquema del montaje del enchufe

La resistencia y el potenciómetro no son estrictamente necesarios, sin embargo, la resistencia permite regular el voltaje de entrada de la alimentación de la pantalla, por lo que tendrá más o menos brillo. Por otro lado, el potenciómetro regula el contraste de la pantalla, por lo que es posible que, sin el potenciómetro, no aparezcan las letras en la pantalla debido al contraste de ésta.

Para el módulo bluetooth, los pines TX (transmisión) y RX (recepción) se han conectado a los pines 3 y 4 del Arduino respectivamente. No se ha utilizado los pines RX y TX del Arduino ya que, como la conexión entre Arduino y el ordenador se realiza por dichos pines, si conectásemos en ellos el módulo, se perdería la conexión con el ordenador. Por tanto, hay que configurar dichos pines en el sketch, el pin 3 como RX y el pin 4 como TX. Tienen que ir cruzados, ya que cuando uno envía, el otro tiene que recibir y viceversa.

El siguiente paso es realizar las conexiones de los enchufes. En primer lugar, se conectan al enchufe macho las 3 partes del cable con toma de tierra en sus respectivas posiciones siendo el de color marrón fase, el azul neutro y el verde y amarillo tierra.

Por el otro lado, la fase se conecta al puerto común del relé y el neutro y la tierra se conectan con el enchufe hembra. A su vez, un nuevo cable que actuará como fase del enchufe hembra, va conectado al puerto normalmente cerrado del relé.

Esta parte, sumada a la anterior, formarían el enchufe inteligente. El pin VCC (corriente) del relé se conecta al pin 3.3V del Arduino, el pin GND (tierra) del relé con la tierra de la placa de pruebas y en este caso, el pin S (señal) del relé con el pin 2 del Arduino.

Podría alimentarse la placa de pruebas con 3.3V y el relé con 5V, pero en ese caso, no se dispone de suficiente voltaje para iluminar la pantalla LCD, es por eso por lo que se ha decidido esta conexión. Finalmente, todas las partes han sido introducidas en una caja de luces. El resultado es el siguiente:

Figura 16. Resultado final del enchufe

6.2 Aplicación Android

A continuación, se explicarán las diferentes partes de la programación de la aplicación Android.

6.2.1 Entorno de trabajo

Para el desarrollo de la aplicación se ha elegido el IDE Android Studio ya que es el IDE oficial para desarrollo de aplicaciones Android y posee múltiples funcionalidades que serán de utilidad en el desarrollo de la aplicación. Por ejemplo, la existencia de simuladores de dispositivos Android, un

plugin que permite realizar pruebas unitarias de manera simple y un componente gráfico que permite crear fácilmente las interfaces de la aplicación.

Para la depuración de la aplicación se ha utilizado un móvil Xiaomi Pocophone F1. No obstante, está aplicación es compatible con cualquier dispositivo que posea un sistema operativo Android 4.0.3 o superior.

6.2.2 Permisos

Antes de empezar con la programación, hay que modificar el fichero AndroidManifest.xml. Esto se debe a que vamos a utilizar una comunicación bluetooth y solicitar un servicio web y ambas necesitan permisos especiales para ello.

El fichero AndroidManifest.xml es el archivo de configuración del proyecto donde aparece información esencial sobre la aplicación que necesita el sistema Android. Entre otras cosas, contiene el nombre del paquete Java de la aplicación, describe los componentes de la aplicación, declara los permisos que debe tener la aplicación, declara el nivel mínimo de Android API que requiere la aplicación, etc.

Para que la aplicación tenga permisos para utilizar el bluetooth, es necesario añadir al fichero las líneas de código de la Figura 17:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Figura 17. Permisos para usar bluetooth

Por otro lado, para poder utilizar el servicio web, es necesario realizar 2 acciones. La primera, hay que modificar el fichero AndroidManifest.xml añadiendo permisos de internet y además hay que crear un fichero XML en el que se declarará la configuración de seguridad. Las modificaciones en AndroidManifest.xml se muestran en la Figura 18:

```
<uses-permission android:name="android.permission.INTERNET"/>

<application
    android:networkSecurityConfig="@xml/network_security_config"
    android:usesCleartextTraffic="true"
```

Figura 18. Permisos de internet y declaración de fichero de seguridad

El resultado final del fichero XML se ilustra en la Figura 19:

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

Figura 19. Fichero de configuración de seguridad en Internet

Con estas modificaciones ya se pueden realizar tanto comunicaciones bluetooth como peticiones HTTP al servidor web.

6.2.3 Comunicación bluetooth con Arduino

Para establecer la conexión entre ambas partes, se ha creado la clase ComunicacionBT, la cual extiende de la clase Application predefinida en Android.

La razón por la que extiende de Application es que esta clase va a ser usada en múltiples actividades de la aplicación. Para hacer uso de esta clase y que pueda ser llamada desde cualquier parte de la aplicación, es necesario volver a modificar el fichero AndroidManifest.xml. En este caso únicamente hay que definir la aplicación con el nombre de la clase. Para ello, será necesario añadir la línea de código mostrada en la Figura 20:

```
<application
    android:name=".ConnectionBT"
```

Figura 20. Definición de la aplicación en el manifest

Dicha línea de código tiene que ir dentro de la etiqueta <application>.

Dentro de la clase ConnectionBT, en primer lugar, se necesita conocer la dirección MAC y el UUID del módulo bluetooth HC-06. Para conocer la MAC basta con realizar una búsqueda bluetooth en un teléfono móvil y aparecerá la MAC del módulo, que en este caso es *00:14:03:05:5A:56*. Por otro lado, el UUID siempre es el mismo para los módulos HC-06, que es *00001101-0000-1000-8000-00805F9B34FB*.

Una vez conocidos esos datos, se procede a la conexión de ambas partes, para ello, se han utilizado las clases BluetoothAdapter, para obtener el adaptador bluetooth del smartphone, y BluetoothSocket para establecer la comunicación entre smartphone y el módulo.

Además de esas dos clases, será necesario también el uso de BluetoothDevice. Esta clase hará la función del módulo bluetooth. Al crear un objeto de esta clase, es necesario proporcionarle una dirección MAC, que en este caso será la del módulo bluetooth.

Cuando se tenga el objeto de la clase BluetoothDevice, ya se puede crear una comunicación entre la aplicación y dicho objeto. Para ello, hay que inicializar el objeto de la clase BluetoothSocket realizando una llamada al método que aparece en la Figura 21, el cual se encarga de crear una comunicación no segura pero que es suficiente para el uso que se le va a dar en este trabajo:

```
btSocket = device.createInsecureRfcommSocketToServiceRecord(MY_UUID);
```

Figura 21. Creación de la comunicación entre aplicación y módulo

Finalmente, hay que llamar al método connect() del objeto BluetoothSocket y habrá sido creada la conexión.

Para la transmisión de datos basta con utilizar un InputStream y un OutputStream para la entrada y salida de datos respectivamente.

6.2.4 Comunicación con el servidor

Para realizar la petición al servidor, se ha utilizado el método POST. Android cuenta con una librería para realizar peticiones HTTP, esta es la librería *Volley*.

Antes de utilizar esta librería, hay que añadir una dependencia en el fichero build.gradle (Module:app).

Esta dependencia se encargará de importar la librería, para ello, hay que añadir al fichero comentado la línea de código mostrada en la Figura 22:

```
dependencies {
    implementation 'com.android.volley:volley:1.1.0'
}
```

Figura 22. Importación de la librería Volley

Después de realizar la modificación, será necesario reconstruir el proyecto y una vez finalice, la librería habrá sido importada.

El funcionamiento de Volley comienza creando un objeto de la clase StringRequest.

El constructor de este método necesita de 4 parámetros. En primer lugar, (i) el tipo de solicitud, que en este caso será POST, (ii) la URL del servicio web, y dos nuevos objetos, el primero de ellos (iii) Response.Listener donde se recogerá la respuesta del servidor si no ha ocurrido ningún error, y el segundo de ellos (iv) Response.ErrorListener, donde se recoge el error que ha ocurrido en la petición, si es que ha habido alguno.

Además de esto, es necesario sobre escribir el método getParams() de la clase StringRequest. Este método devuelve un mapa con los parámetros que se desean enviar al servidor. El mapa contiene un par de strings que serán los parámetros que se envíen al servidor. El mapa se muestra en la Figura 23:

Figura 23. Parámetros solicitud POST

Finalmente, hay que realizar una llamada al método newRequestQueue() de la clase Volley y añadir el objeto creado de la clase StringRequest para realizar la petición.

6.3 Arduino

Finalmente, se explican los diferentes pasos para la programación de la placa Arduino.

6.3.1 Entorno de trabajo

Para la programación de la placa se ha utilizado el IDE de Arduino que puede ser descargado de la página oficial https://www.arduino.cc/en/main/software.

Para la alimentación de la placa se ha utilizado el puerto USB de la misma y se ha conectado a una batería.

6.3.2 Librerías utilizadas

Se han utilizado dos librerías, por un lado, la librería *LiquidCrystal.h* para el control de la pantalla LCD y la librería *SoftwareSerial.h* para el control del módulo bluetooth.

La librería utilizada para el módulo no sería necesaria si se hubieran utilizado los puertos TX y RX de la placa, pero como se ha explicado anteriormente, no han sido utilizados. Esta librería lo que permite es definir 2 pines de entrada como pines de transmisión y recepción. Se ha utilizado el pin 3, como pin TX del módulo bluetooth y el pin 4 como pin RX.

En cuanto a la librería LiquidCrystal, se encarga de definir los pines necesarios para la comunicación con la pantalla LCD. Necesita definir un pin RW (escritura y lectura), un pin Enable y unos pines de datos, que en este caso son los pines D1, D2, D3 y D4.

Los pines de la placa utilizados para el control de la pantalla LCD son 12, 11, 8, 7, 6 y 5 respectivamente.

6.3.3 Configuración y programa

Lo primero que hay que hacer es configurar los pines e iniciar la pantalla LCD, el módulo bluetooth y el Serial, que es la conexión entre placa y ordenador. Esto se hace en el método setup(). Este método se ejecuta una única vez al inicio del programa y su contenido se muestra en la Figura 24:

```
void setup() {
  pinMode(RelayPin, OUTPUT); //exit mode
  pinMode(BTPin, OUTPUT);

  digitalWrite(BTPin, HIGH);
  lcd.begin(COLS, ROWS);
  Serial.begin(9600);
  BTS.begin(9600);
}
```

Figura 24. Setup del sketch de la placa Arduino

En primer lugar, define el pin del relé, que es el 2, y el pin del módulo bluetooth, que es el 10, como pines de salida.

A continuación, manda una señal *HIGH* al pin del módulo para su activación. Sin esta señal, no sería posible conectar con la aplicación.

Finalmente, inicializa la pantalla LCD indicando que se dispone de 2 filas y 16 columnas, haciendo uso de las variables *COLS* y *ROWS*, e inicializa tanto el Serial como el módulo bluetooth a 9600 baudios.

Una vez finalizada la configuración, se procede a la ejecución del programa principal, que tiene lugar en el método loop(). Este método está siempre en ejecución.

Debido a que este método está ejecutándose en bucle siempre, se han definido una serie de variables booleanas para simular una programación por eventos.

Estas variables son:

- *modeChanged* → Su valor es falso si el modo no ha cambiado y verdadero si ha habido un cambio de modo. Si el modo ha cambiado, esta variable hace que se muestre el nuevo modo en la pantalla LCD.

```
if(modeChanged == true){
    changeMode();
    modeChanged = false;
  }
```

Figura 25. Acción para la variable modeChanged

- *flag* → Su valor es falso si no se ha recibido ningún mensaje por el módulo bluetooth y verdadero si se ha recibido algún dato y ha de ser procesado.

```
if(flag){
    parseCommand();
    flag = false;
    index = 0; //Clear
  }
```

Figura 26. Acción para la variable flag

- *waiting* Usada para el control del modo de emergencia (modo 2) y el modo ahorro (modo 3). Su valor es falso hasta que se entra en uno de dichos modos, momento en el que su valor pasa a ser verdadero. Como se muestra en la siguiente figura, dependiendo del modo en el que se encuentre realiza una función u otra:

```
if(waiting){
  //Mode 2
  if(mode == 2){
    //Acciones del modo de emergencia
    }
  //Mode 3
  else if (mode == 3){
    //Acciones del modo ahorro
  }
}
```

Figura 27. Acción para la variable waiting

Cada modo realizará una acción u otra. En el modo manual, únicamente se apaga el relé si se selecciona el estado off o se enciende el relé si se selecciona el estado on. Para encender el relé hay que mandar una señal *HIGH* al pin correspondiente y una señal *LOW* para apagarlo.

Por otro lado, si se está en el modo de emergencia (modo 2), se activa el relé por un determinado periodo de tiempo, que corresponderá a la carga del 60% de la batería. Después, se realizará 40% restante cuando el coste de la electricidad sea más bajo. Para ello, esperará hasta que el precio de la electricidad sea más barato y, entonces, continuará el proceso de recarga.

El control de esta acción realiza con la variable *charging*, cuyo valor inicial es verdadero y pasará a tener el valor falso cuando se reinicie el temporizador. El temporizador se reiniciará una única vez al expirar el tiempo que tiene que estar activado el relé.

Una vez reiniciado el temporizador, da comienzo la espera hasta que la electricidad sea más barata y, en ese instante, continua la recarga.

La siguiente figura muestra las acciones que se realizan en el modo de emergencia:

```
//Emergency Mode
    if(mode == EMERGENCY){
      //First wait, charging
      if(charging){
        currentMillis = millis();
        //Disconnect relay when time is up.
        if(currentMillis - initialMillis > waitTime){
          disconnectRelay();
          state = EM_MODE_OFF;
          showState();
          charging = false;
          waitingToCharge = true;
        }
      }
      else{
        if(!timerRestarted){
          initialMillis = millis();
          timerRestarted = true;
        }
        //Wait to charge, save mode simulation
        if(waitingToCharge){
          currentMillis = millis();
          if(currentMillis - initialMillis > waitTime){
            connectRelay();
```

Figura 28. Acciones del modo de emergencia

En cuanto al modo ahorro (modo 3), el enchufe permanece desactivado hasta que haya pasado el tiempo necesario para que la recarga del vehículo eléctrico sea lo más económica posible. Esta espera se controla con la variable *waitingToCharge*, cuyo valor es verdadero hasta que haya transcurrido el tiempo, instante en el cual su valor pasa a ser falso. En este instante, el relé se activa para que de comienzo la recarga del vehículo.

El proceso de recarga finalizaría cuando el vehículo haya completado la carga de su batería. El control de esta acción realiza con la variable *charging*, cuyo valor inicial es falso y pasará a tener el valor verdadero cuando se reinicie el temporizador. Esta acción se realizará siempre una única vez al expirar el tiempo que tiene que estar desactivado el relé. Una vez reiniciado el temporizador, da comienzo la carga.

A continuación, en la Figura 29, se muestra el código del modo ahorro:

```
//Save Mode
else if (mode == SAVE){
  unsigned long currentMillis;
  if(waitingToCharge){
    currentMillis = millis();
    if(currentMillis - initialMillis > waitTime){
      connectRelay();
      state = SAVE_MODE_ON;
      showState();
      waitingToCharge = false;
      waiting = false;
    }
  }
}
```

Figura 29. Acciones del modo ahorro

6.3.4 Comunicación bluetooth con la aplicación Android

El primer paso es, en el método loop(), comprobar si el módulo bluetooth (*BTS*) tiene un número de bytes para leer. Esto se realiza con la función available().

En caso de que haya bytes por leer, entonces, se entra en un bucle mediante el cual, mientras haya bytes por leer, se leen dichos bytes y se almacenan en un array de caracteres. Cuando se haya acabado de leer, entonces se cambia el valor de la variable *flag* a verdadero para avisar de que hay comandos por procesar. El código se muestra en la Figura 30:

```
if(BTS.available()){
  while(BTS.available() > 0){
    c = BTS.read();
    delay(10); // requeired
    data[index] = c;
    index++;
  }
  flag = true;
}
```

Figura 30. Código lectura del módulo bluetooth

En cuanto a la escritura, únicamente es necesario llamar a la función print().

6.3.5 Temporizador

Para controlar las esperas necesarias en los modos de emergencia y de ahorro, se ha creado un temporizador haciendo uso de la función millis(). Esta función devuelve el tiempo en milisegundos desde que la placa se ha reiniciado.

En primer lugar, hay que realizar una primera llamada dicha función, almacenando el valor devuelto en la variable *initialMilis* definida al principio del programa. Esta variable solamente tiene que actualizarse una única vez, que es al inicio de la espera. Por otro lado, cada iteración del loop() se realiza una nueva llamada y se almacena en otra variable denominada *currentMilis*. Estas variables se comparan y cuando la resta de ambas sea mayor que el tiempo que hay que esperar, entonces es que ha pasado el tiempo y hay que realizar una acción dependiendo del modo en el que se encuentre. En la Figura 31 se ilustra el código del temporizador:

```
unsigned long currentMillis = millis();
  if(currentMillis - initialMillis > waitTime){
    // Realizar accion
  }
```

Figura 31. Código del temporizador

7. Resultados

En este apartado, se realizará una simulación de una recarga de un vehículo eléctrico para cada uno de los modos existentes.

Los datos para la simulación son los siguientes:

- TC = Tiempo necesario para recargar completamente la batería
- P(h) = Precio del kilovatio (kW) en una hora h
- HB = Hora donde el precio de kW es más barato
- S = Hora de salida
- I = Hora de inicio de carga
- F = Hora de fin de carga

A continuación, se muestra una gráfica con el precio del kilovatio por horas para el 9 de septiembre de 2019, según la tarifa PVPC (Precio Voluntario al Pequeño Consumidor) en la modalidad de vehículos eléctricos.

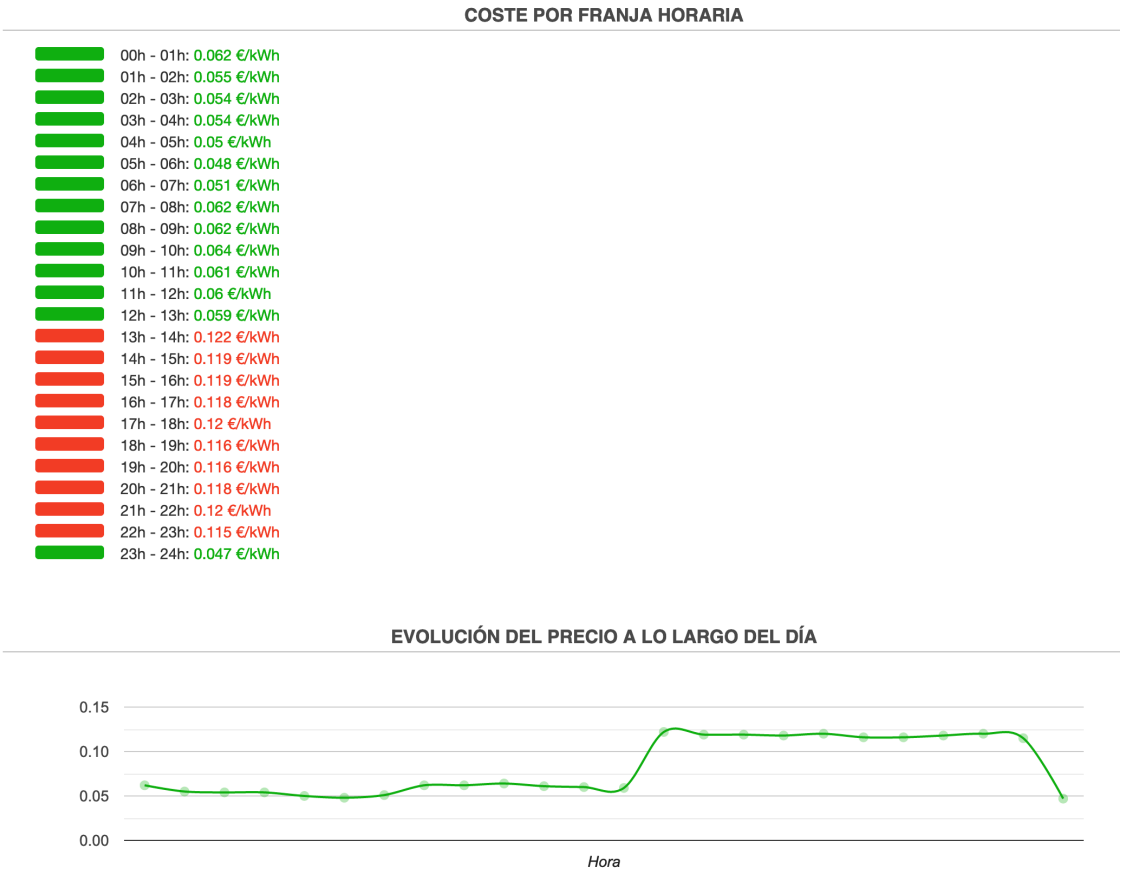

Figura 32. Precio del kWh el 9 de septiembre 2019 (PVPC vehículos eléctricos) [18]

Teniendo en cuenta la tarifa anteriormente presentada, vamos a analizar el coste de una recarga utilizando los distintos modos que permite nuestro enchufe, es decir, modo manual, ahorro y emergencia. Para el modo de ahorro, el servidor utiliza una fórmula mediante la cual empieza a cargar en el momento en el que la hora donde el kilovatio es más barata. En el modo manual, la recarga comienza inmediatamente sin tener en cuenta el coste de la electricidad. Mientras que, en el modo de emergencia, el enchufe esta activo durante un periodo de tiempo, donde la batería alcance el 60% y después se desactiva a la espera de que llegue el periodo donde la electricidad sea más barata.

El escenario planteado es el siguiente:

- TC = 10 horas
- HB = 23:00
- S = 12:00
- I = 18:00
- F = 2:00
- Vehículo Tesla Model S con batería de 60 kW con un porcentaje de carga del 39% [19]

Además, suponemos que realizamos la carga en nuestro domicilio y por tanto, un enchufe de casa convencional, permite cargar 3,7kw de potencia por hora.

En el modo manual, se iniciaría la recarga a las 18:00 (justo al conectar el vehículo al enchufe, cuando el precio es uno de los más elevados durante el día), y se finalizaría a las 04:00.

Si se utilizara el modo de emergencia, la recarga también empezaría a las 18:00. Sin embargo, el proceso de recarga pararía cuando la batería estuviera al 60%, es decir, aproximadamente 3,5 horas después de comenzar, a las 21:30. Posteriormente, la recarga continuaría a partir de las 23 horas y finalizaría 6,30 horas después, es decir, a las 5:30.

Por otro lado, si se utilizara el modo ahorro, la recarga comenzaría a las 23:00 y finalizaría a las 11:00 del día siguiente.

Teniendo en cuenta todo lo anterior, el coste que tendría cada recarga, de acuerdo con los precios de la Figura 31, sería de 3,17€ para el modo manual, 2,93€ para el modo de emergencia y de 2,48€ para el modo ahorro.

Por tanto, la diferencia de usar el modo de emergencia frente al modo manual supone un 7,5% de ahorro y usar el modo ahorro supone un ahorro del 21,8%.

8. Conclusiones

Como se puede constatar en muchos de países, el número de ventas de vehículos eléctricos está creciendo año tras año, prácticamente con un ritmo exponencial y se espera que siga con esta tendencia, por lo que podría llegar a superar en número de vehículos en circulación a los vehículos de combustión interna a medio plazo. También hemos presentado las diferentes tarifas eléctricas que se ofertan y el precio de la electricidad en España, que es uno de los más elevados del mundo.

Por otro lado, en la actualidad, el denominado Internet Of Things o Internet de las cosas, es un tema que está de moda. Este término hace referencia a que objetos de la vida cotidiana, como puede ser un enchufe, sean capaces de conectarse con Internet permitiendo así, que tengan usos para los que no habían sido diseñados en un principio.

En el presente Trabajo de Fin de Grado se ha realizado el diseño e implementación de un enchufe inteligente basado en Arduino que permite reducir el coste de la electricidad necesaria para cargar los vehículos eléctricos. Más en detalle, nuestro enchufe posee 3 modos diferentes: (i) el modo manual, donde el enchufe se encuentra activado al conectar el vehículo, y por tanto, comienza el proceso de carga, (ii) el modo emergencia, donde el enchufe se encuentra activo hasta que se alcanza un determinado porcentaje de batería y después se espera a realizar la recarga en un periodo donde ésta sea más económica, y (iii) el modo ahorro, donde el enchufe no se activa hasta que un servidor externo determina cuándo es más barato iniciar el proceso de recarga del vehículo.

Una vez implementado el prototipo, y realizadas las pruebas, también he analizado con un ejemplo, cómo funcionaría mi propuesta y cuál sería el posible ahorro obtenido. En concreto, según nuestras estimaciones, la diferencia de usar el modo de emergencia frente al modo manual puede suponer un 7,5% de ahorro, mientras que usar el modo ahorro permite ahorrar hasta un 21,8% del precio de la energía. Estos datos demuestran que el uso de nuestro enchufe va a ser interesante para los usuarios, además de que puede animarles definitivamente a la hora de adquirir un vehículo eléctrico.

En definitiva, el objetivo de este proyecto, que es la realización de un enchufe que permita un ahorro en el proceso de recarga de un vehículo eléctrico se ha cumplido. Además, se han adquirido conceptos teóricos, como son la historia, funcionamiento y estado actual de los vehículos eléctricos, los diferentes modos de recarga existentes y los diferentes tipos de tarifas eléctricas en España, y conceptos prácticos, como el desarrollo de aplicaciones Android y la programación de placas Arduino.

9. Bibliografía

[1] Herrera, Cristina. (2017). *Así influyen los coches en el cambio climático*. Noticias eltiempo.es.

Recuperado de https://noticias.eltiempo.es/asi-influyen-los-coches-en-el-cambio-climatico/

[2] *Electric car use by country*. (s.f). *Wikipedia*.

Recuperado de https://en.wikipedia.org/wiki/Electric_car_use_by_country

[3] Ocu (24 de mayo de 2019). *La electricidad más cara que nunca*. Expansión. Recuperado de https://www.ocu.org/vivienda-y-energia/gas-luz/noticias/precio-electricidad-espana-europa

[4] Nakamura, M y Gargenta, M. (2014). *Learning Android, 2nd Edition.* Estados Unidos: O'Reilly Media.

[5] Pascual, Juan Antonio. (2018). *Android vs iPhone: La guerra de los smartphones en cifras*. ComputerHoy.

Recuperado de https://computerhoy.com/reportajes/industria/android-vs-iphone-guerra-smartphones-cifras-271447

[6] *¿Qué es Android?* (8 de febrero de 2011). *Xataka*.

Recuperado de https://www.xatakandroid.com/sistema-operativo/que-es-android

[7] Qué es Arduino, cómo funciona y qué puedes hacer con uno. (21 de julio de 2018). Xataka. Recuperado de https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

[8] *Historia del vehículo eléctrico*. (s.f). *Wikipedia*. Recuperado de https://es.wikipedia.org/wiki/Historia_del_veh%C3%ADculo_el%C3%A9ctrico

[9] Ruiz, Débora. (2019). *Electrolineras: El consumo energético sostenible*. Revista digital. Recuperado de https://revistadigital.inesem.es/gestion-integrada/electrolineras/

[10] *ForocochesElectricos* (2019). *¿Cuántos puntos de recarga de coches eléctricos hay en España?*

Recuperado de https://forococheselectricos.com/2019/06/cuantos-puntos-de-recarga-de-coches-electricos-hay-en-espana.html

[11] Martínez García, Gonzalo. (2019). *Actualización de los supercargadores de Tesla en España. MovilidadElectrica*.

Recuperado de https://movilidadelectrica.com/supercargadores-de-tesla-espana/

[12] *ElectroMovilidad* (s.f). *Tipos de batería para coche eléctrico*.

Recuperado de http://electromovilidad.net/tipos-de-bateria-para-coche-electrico/

[13] V. Torres-Sanz, J. A. Sanguesa, F. J. Martinez, P. Garrido and J. M. Marquez-Barja, *"Enhancing the Charging Process of Electric Vehicles at Residential Homes,"* in IEEE Access, vol. 6, pp. 22875-

22888, 2018. doi: 10.1109/ACCESS.2018.2829158. Recuperado de https://ieeexplore.ieee.org/document/8344801

[14] *Bioeduca Málaga* (s.f). *Reduce tu factura eléctrica*.

Recuperado de http://bioeduca.malaga.eu/es/detalle-contenido/Reduce-tu-factura-electrica#!tab3

[15] *IFPUG*. (14 de Julio de 2017). *Software economics and function points metrics.*

Recuperado de https://www.ifpug.org/wp-content/uploads/2017/04/IYSM.-Thirty-years-of-IFPUG.-Software-Economics-and-Function-Point-Metrics-Capers-Jones.pdf

[16] *PromeTec* (s.f)*. Módulo Bluetooth HC-06.*

Recuperado de https://www.prometec.net/bt-hc06/

[17] *Fritzing.* (s.f). *Wikipedia.* Recuperado de https://es.wikipedia.org/wiki/Fritzing

[18] *TarifaLuzHora* (s.f). *Precio de la luz por horas*.

Recuperado de https://tarifaluzhora.es/?tarifa=coche_electrico&fecha=2019-09-09

[19] *Tesla Model S*. (s.f). *Wikipedia.*

Recuperado de https://es.wikipedia.org/wiki/Tesla_Model_S