Trabajo Fin de Grado

Sistema de Información/Entretenimiento para vehículo con soporte de voz y diagnóstico

Autor

Juan Antonio Cepero Chicote

Director

Darío Suárez Gracia

Escuela de Ingeniería y Arquitectura

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

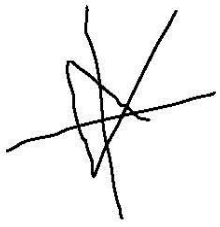
D./Dª. Juan Antonio Cepero Chicote,

con nº de DNI 72898976-Q en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Sistema de Información/Entretenimiento para vehículo con soporte de voz y diagnóstico,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 19 de Septiembre de 2016



Fdo: Juan Antonio Cepero Chicote

Resumen ejecutivo

El presente proyecto presenta el diseño un sistema de información/entretenimiento instalable en cualquier vehículo. Está formado por un computador de bajo coste y una pantalla táctil fácilmente acoplables al salpicadero de un coche y cumple un doble objetivo: permite conocer los parámetros propios de la conducción (aceleración, RPM…) en tiempo real, así como mantener en todo momento la seguridad de los ocupantes del vehículo evitando que el conductor deba soltar las manos del volante para interactuar con el sistema ya que permite su control mediante la voz. Por otra parte, el sistema también ofrece otra serie de funcionalidades, como la de centro multimedia conectado al sistema de altavoces del automóvil, o permitir el acceso a Internet (siempre que se disponga de conexión a la Red). Además, su diseño modular permite añadir fácilmente extensiones en el futuro por cualquier interesado.

El "cerebro" del sistema es un computador de bajo coste, Raspberry Pi 3, con un Sistema Operativo derivado de Debian Linux, la cual ofrece un enorme potencial y versatilidad al proyecto debido a lo extendido que está su uso entre la Comunidad Maker, y la gran cantidad de documentación disponible en la Red. En lo referente al Sistema Operativo, se encuentra almacenado en una tarjeta microSD para permitir que el proyecto sea mucho más flexible y escalable en funcionalidades que los Sistemas de Infotenimiento propietarios que integran algunos vehículos de serie, cuyos Sistemas Operativos cerrados no permiten la adición de más características.

Todo el software que se ha utilizado es Open Source, y el que se ha desarrollado mantiene esa misma licencia, ya que otro de los objetivos principales de este proyecto es dejarlo a disposición de quien desee utilizarlo en algún repositorio de código de forma pública. La idea es que otros usuarios puedan reproducir el trabajo aquí realizado con relativamente poco esfuerzo. Esta es la dirección del repositorio en el que se encuentra todo el código desarrollado:

https://github.com/cepero/Carputer

Sobre la Raspberry Pi corre una aplicación, compuesta por diversas partes, desarrollada en el proyecto, que centraliza las citadas funcionalidades y simplifica su uso. En el diseño de esta aplicación se han tenido en cuenta varios Principios Generales de Diseño de Interfaces para que sea lo más funcional e intuitiva posible, además de visualmente atractiva. Esta aplicación, escrita en lenguajes PHP y Python, permite el manejo de todas las características disponibles en el sistema, bien mediante periféricos tradicionales como ratón y teclado, bien mediante la pantalla táctil integrada, o bien mediante comandos por voz.

Se han empleado Metodologías Ágiles para el desarrollo de todo el proyecto, el cual se ha compuesto de 5 tareas principales:

1- Comunicación con el vehículo para extracción y procesado de datos provenientes del motor, mediante el uso del puerto OBDII del coche y el protocolo Bluetooth.
2- Presentación de la información del vehículo.
3- Control del sistema mediante comandos por voz.
4- Integración del sistema en el salpicadero de un vehículo
5- Pruebas de integración entre los distintos sistemas.

Dada la naturaleza de este proyecto, podría decirse que está directamente relacionado con una buena parte de las competencias del Grado en Ingeniería Informática, desde la interacción con un sistema embebido, hasta el desarrollo software con tecnologías Web, por lo que cubre buena parte de la titulación en su conjunto.

A mis padres, por haberme traído hasta donde estoy ahora.

A Laura, por apoyarme durante el trayecto.

Agradecimientos

En primer lugar, agradecer a Eduardo Jarabo Baún, de la empresa *Instalaciones EdJar*, su inestimable ayuda a la hora de integrar el sistema en el vehículo. Su compromiso e implicación, además de su infinita paciencia han permitido no solo la finalización de este Trabajo de Fin de Grado, sino también la transmisión de una serie de conocimientos de gran valor al alumno.

También dar las gracias al director de este trabajo, Darío Suárez Gracia, por creer en mí y apoyar el proyecto desde el primer día. Sus consejos y múltiples aportaciones han conseguido mejorar enormemente diferentes aspectos de este proyecto.

Agradecer a mi padre, Juan A. Cepero Andrés, el gran apoyo que me ha prestado. Su interés constante en el proyecto y su inestimable ayuda en algunos puntos del desarrollo del mismo han sido claves, ya que, de haberlo hecho en solitario, no se hubiesen alcanzado los resultados que se presentan a continuación.

A mi madre, Mª Lorena Chicote García, y a mi hermana, Lorena I. Cepero Chicote, por sus valiosos comentarios y propuestas de mejora relacionadas con el diseño de la interfaz. Sois las dos unas excelentes *betatesters*.

A Laura de Miguel Delgado, por su enorme apoyo y comprensión durante todo el proceso de desarrollo. Gracias por ser mi burbuja de oxígeno y conseguir hacerme reír constantemente.

Índice

Índice de Figuras

1. Introducción

Hoy en día la mayoría de vehículos de alta gama integran en su interior lo que se conoce como Sistema de Infotenimiento (o Infotainment System en Inglés). Estos sistemas suponen una integración vertical de hardware y software que ofrecen a los ocupantes del vehículo acceso tanto a información importante durante la conducción (información del tráfico, GPS, velocidad actual…), como todo tipo de entretenimiento (videos, música…). Bien es cierto, que, aunque la tendencia en la integración de estos sistemas se está extendiendo a otras gamas de vehículos, su uso es aún minoritario. Esto es debido principalmente a la elevada edad media del parque automovilístico español, tal y como se recoge en [1] y [2].

Prácticamente la totalidad de fabricantes de coches han desarrollado sistemas de este tipo, con mayor o menor grado de satisfacción para sus usuarios. Sin embargo, el común denominador de todos estos sistemas es que su software asociado es propietario e imposible de modificar para añadir más características. En secciones sucesivas de esta memoria se repasarán algunos de estos sistemas y se compararán entre sí para comprobar las funcionalidades que ofrece cada uno de ellos.

Es por eso que este Trabajo de Fin de Grado pretende dar solución a algunos de los inconvenientes que presentan este tipo de sistemas, como la utilización de Software cerrado, o el elevado desembolso necesario para su adquisición. No se pretende en ningún momento competir con aquellos, sino simplemente dejar patente que un Ingeniero en Informática puede desarrollar un sistema similar en muchos puntos, e incluso mejorarlo en algunos, todo ello con un coste económico muy reducido.

Para poder alcanzar este objetivo, se han reutilizado y adaptado algunas librerías Open Source que se encontraban disponibles en Internet y de las cuales se hablará más adelante. A continuación, se describe brevemente la estructura de las sucesivas secciones de esta memoria:

Capítulo 2: Motivación - En este apartado se explicarán los motivos por los que el alumno ha decidido realizar este Trabajo de Fin de Grado.

Capítulo 3: Estado del Arte – Se analizarán aquí algunos sistemas o proyectos similares al que se pretende realizar.

Capítulo 4: Requisitos – Se recogerán aquí los Requisitos, tanto Funcionales, como no Funcionales que debe cumplir el proyecto.

Capítulo 5: Metodología – En esta sección se tratará la forma en la que se ha abordado la realización del proyecto, herramientas utilizadas, gestión del tiempo, etc.

Capítulo 6: Arquitectura del Sistema – Se detallarán aquí las partes principales que constituyen este proyecto.

Capítulo 7: Conclusiones – En este apartado se recogerán las conclusiones que el alumno ha obtenido tras la realización del proyecto.

Capítulo 8: Bibliografía – Aparecen aquí todas las referencias que se han empleado para la realización del presente proyecto.

Capítulo 9: Anexos – Se recogen bajo esta sección la explicación pormenorizada de algunas decisiones de diseño que se han ido tomando, así como algunas fotografías del sistema ya montado.

2. Motivación

La motivación principal de este proyecto es dotar a cualquier vehículo de "inteligencia" con un coste lo más reducido posible, permitiendo así que personas no necesariamente relacionados con la Ingeniería Informática, puedan disfrutar de sistemas de información/entretenimiento sin tener que realizar un gran desembolso o tener que cambiar su vehículo.

Es importante indicar que el tema en torno al que gira este Trabajo de Fin de Grado (TFG), ha sido propuesto por el alumno, y que éste tenía la idea de la realización de este proyecto desde hace tiempo, y consideró que podía encajar a la perfección con la realización de su TFG. Debido a la extensión temporal de este tipo de trabajos, este TFG constituye la base e implementa las directrices y los módulos principales de un sistema de Infotenimiento para, en un futuro, añadir otras funcionalidades menos importantes. Algunas de estas posibles funcionalidades futuras son por ejemplo navegación GPS, conexión con los Smartphones de los ocupantes del vehículo, cámara trasera para ayuda en el aparcamiento, etc. Sería perfecto también que incluso otros individuos colaboren en el futuro con este proyecto. Por tanto, la arquitectura del sistema que se va a desarrollar está compuesto principalmente por tres partes bien diferenciadas: Adquisición y procesado de parámetros del motor durante la conducción, control del sistema mediante comandos por voz y finalmente la integración del sistema en un vehículo real. Se desarrollará también una aplicación que centralice los servicios disponibles por el momento y que permita la inclusión de forma sencilla de otros servicios en un futuro.

De esta forma, un usuario sin mucha experiencia técnica puede modernizar su vehículo para aprovechar las ventajas que ofrece un sistema de este tipo, algo que hasta ahora solo está disponible en vehículos nuevos o de gama alta, o cambiando la radio integrada por una más moderna y a un precio mucho más elevado (y probablemente con menos funcionalidades).

Otra importante motivación ha sido la de intentar que el presente proyecto tenga la mayor calidad y robustez posible, empleando para ello técnicas de Ingeniería del Software, y algunas herramientas totalmente desconocidas para el alumno hasta ahora, lo cual ha aportado un pequeño plus motivacional.

3. Estado del Arte

Una de las fases iniciales de este proyecto fue la de recopilación de información de diversas fuentes (principalmente en la Web) en busca de proyectos similares o trabajo que se pudiera aprovechar. La realización de esta tarea en etapas muy tempranas del desarrollo permitió conocer el entorno relacionado con el proyecto y resultó de vital importancia.

La conclusión a la que llegó el alumno después de realizar esta búsqueda fue que actualmente no existen proyectos tan completos como el que se propone aquí. Sin embargo, sí que es verdad que algunas partes de este trabajo (como la comunicación con el motor o una parte del control por voz) se han basado en bibliotecas y trabajo previo ya existente (bibliotecas Py-obd ([3]) y PocketSphinx ([4]), de las cuales se hablará más adelante). El alumno consiguió adaptar dichas bibliotecas para su uso dentro de un mismo proyecto. También se descubrió la existencia de soluciones comerciales (tanto hardware como software de pago, algunas de las cuales se mencionarán más adelante en este apartado) que cubrían parte de los requisitos de este proyecto, aunque su aplicación hubiese impedido la flexibilidad y escalabilidad necesarias en este proyecto, además de ir en contra de la filosofía Open Source del mismo.

Entrando más en detalle, se aprovechó una biblioteca escrita en Python y disponible en repositorios públicos para conectar la Raspberry Pi al puerto OBDII del vehículo (del cual se hablará más adelante en la sección 6.2 de esta memoria). Esta biblioteca se conoce como *Py-obd*. Por otra parte, también se aprovechó otra biblioteca pública, esta vez escrita en C, llamada PocketSphinx, que se encarga del reconocimiento de patrones de voz. No se encontró ningún trabajo o prototipo en el que esta biblioteca se emplee para controlar un sistema de Infotenimiento por medio de la voz, y fue necesario un trabajo importante de adaptación al presente proyecto.

En definitiva, el Estado del Arte en el momento de comenzar con este TFG se reducía a soluciones comerciales, tanto hardware como software, poco flexibles y precios desorbitados, y a algunos pequeños proyectos parecidos, pero sin las funcionales que este TFG presenta.

El hecho de haber aprovechado un par de bibliotecas software extraídas de la Red aceleró considerablemente el desarrollo del proyecto, y carecía de sentido implementar las funcionalidades de ambas desde cero cuando ya existía algo de forma pública y cubría los requisitos necesarios para este proyecto.

A continuación, aparecen una serie de ejemplos de sistemas o proyectos parecidos al que se pretende desarrollar. Se ha intentado que en estos ejemplos aparezcan tanto soluciones "Hágalo usted mismo" (Do It Yourself en inglés), como soluciones comerciales:

OBD-Pi

Este fue uno de los primeros proyectos relacionados que se encontraron. Se trata de un proyecto DIY en el que el usuario "CowFish" integra en el salpicadero una Raspberry Pi (el primer modelo que salió al mercado) y la conecta a la unidad central de su vehículo. El resultado que obtiene es parecido al que se pretende desarrollar, aunque únicamente tiene la funcionalidad de extracción de parámetros del motor. En [5] se puede consultar toda la información relativa a este proyecto:

Figura 1 - Proyecto OBD-Pi

Reproductor multimedia DIY con pantalla táctil

El usuario Eric Kester quería sustituir la radio de su antiguo vehículo y decidió montar todo un centro multimedia con la ayuda de una Raspberry Pi y una pantalla táctil. De nuevo este proyecto únicamente cubre una de las funcionalidades que se pretenden implementar en el presente TFG. En [6] se puede consultar cómo lo hizo:

Figura 2 – Reproductor multimedia integrado en salpicadero

i-Carus

Es la primera de las soluciones comerciales que se han encontrado. Además, es probablemente la más completa en cuanto a funcionalidades. Sin embargo, el problema principal es su elevado precio, además de la poca flexibilidad que ofrece, ya que todo el software (incluido el Sistema Operativo) viene precargado en una tarjeta SD y no es posible modificarlo. Este proyecto incluye no solo el citado software propietario, sino que también incluye el dispositivo hardware como tal en el que se aloja, muy similar a una radio de coche en formato 2DIN con pantalla táctil. En [7] se encuentra la página principal de este proyecto:

Figura 3 – Proyecto i-Carus

Carberry

Representa otra solución comercial, pero con un enfoque muy diferente al de i-Carus. Carberry es fundamentalmente un escudo (o shield en inglés), que se conecta a una Raspberry Pi y le dota de una enorme capacidad de comunicación, integrando conectividad GPS, posibilidad de conexión con el motor para extracción de datos… Sin embargo, no ofrecen ningún tipo de solución software, por lo que debe ser el comprador el que consiga exprimir las posibilidades del hardware que ha adquirido. En [8] aparece la página del proyecto:

Figura 4 – Proyecto Carberry

Seguidamente, se adjuntan otra serie de ejemplos de Sistemas de Infotenimiento, esta vez desarrollados por fabricantes de coches, e integrados de serie en algunos de sus vehículos. El objetivo es que se puedan comparar los sistemas/ejemplos comentados anteriormente con soluciones realizadas por equipos de desarrollo muy numerosos, y con gran cantidad de recursos, tanto temporales como económicos, como pueden ser los integrados por grandes casas comerciales de vehículos:

Tesla Infotaintment:

El sistema integrado en los vehículos eléctricos de la marca americana Tesla está compuesto por una enorme pantalla táctil de 17 pulgadas en formato vertical, capaz de mostrar todo tipo de información, desde GPS hasta llamadas telefónicas, pasando por la posibilidad de controlar todos los elementos del vehículo desde la propia pantalla (ya que no tiene controles físicos, todo se realiza desde esta pantalla táctil). Una gran ventaja de este sistema es que, debido al enorme tamaño de esta pantalla, permite mostrar dos aplicaciones a la vez en formato pantalla partida. Este sistema viene de serie en un par de modelos al alcance de muy pocos bolsillos. En [9] se puede consultar la información relativa a este sistema.

Figura 5 – Tesla Infotaintment

Volkswagen Entertainment System:

Este sistema, integrado en algunos modelos de la marca desde 2012 (aprox.) tiene funcionalidades multimedia (como radio o posibilidad de acceder a los contenidos del Smartphone de los pasajeros) y de GPS. Para el control de este sistema, la marca ha integrado en el volante una serie de botones que permite el desplazamiento por los menús de forma muy simple, lo cual supone un punto a favor para este sistema. También integra un control por voz ligeramente primitivo y con no demasiadas opciones. Todo esto, además de información complementaria se puede consultar en [10].

Figura 6 – Volkswagen Entertainment System

Renault R-Link:

Finalmente se va a hacer un repaso por el sistema ofrecido por la marca francesa Renault. Su gran baza es que no solo está disponible en vehículos de las gamas más altas, sino que su integración puede hacerse en cualquiera de sus vehículos (con el desembolsando correspondiente por supuesto). En cuanto a funcionalidades, permite bastante personalización a nivel estético, navegación GPS, realización de llamadas desde la propia pantalla y a nivel multimedia permite también el acceso al contenido de un Smartphone. Al igual que el sistema de Volkswagen, también incluye un pequeño sistema de reconocimiento de patrones de voz, aunque de nuevo, sin exprimir todas las posibilidades que este tipo de control ofrece. En [11] aparece la página principal de este sistema.

Figura 7 – Renault R-Link

Para finalizar esta sección, aparecen otro tipo de soluciones comerciales, esta vez de fabricantes de equipos de sonido para el automóvil. Al igual que sucedía con las soluciones propuestas por las casas automovilísticas, el precio es, de nuevo aquí un factor muy a tener en cuenta si se quiere adoptar uno de estos sistemas:

Pioneer AVIC-8200NEX:

Este dispositivo hardware representa una alternativa a la hora de sustituir la radio integrada de serie en el vehículo. Está formado por una pantalla multitáctil de 7 pulgadas, ocupando un espacio 2DIN en el frontal del salpicadero del vehículo (por lo tanto, no es fácilmente instalable en todo tipo de vehículos, ya que muchos emplean parte de este espacio para colocar controles de aire acondicionado/calefacción). En cuanto a funcionalidades, se encuentra en sintonía con las soluciones anteriormente comentadas, permite la navegación GPS, acceso al contenido multimedia de los Smartphones de los ocupantes, y control por voz. En [12] se puede encontrar más información acerca de este dispositivo.

Figura 8 – Radio Pioneer AVIC-8200NEX

Kenwood DDX9716BTS

Esta es otra opción, muy similar a la anterior, tanto en formato como en funcionalidades. Como elementos diferenciadores, permite un mayor ajuste durante su instalación para reducir posibles brillos al mirar la pantalla, así como la posibilidad de incorporarle una serie de cámaras (que se adquieren por separado) para ayudar en el aparcamiento. De nuevo permite conexión con los teléfonos inteligentes de los pasajeros, pero en este modelo no se permite la interacción por voz, lo cual resta algo de valor al mismo. En [13] se puede encontrar toda la información relacionada con esta opción.

Figura 9 – Radio Kenwood DDX9716BTS

Muchas de las soluciones comerciales presentadas hasta ahora, tienen en común la compatibilidad con Smartphones de los ocupantes del vehículo, permitiendo acceder a determinadas aplicaciones almacenadas en estos. Esta opción es posible gracias a tres formas principales de conexión: En primer lugar, dotar al sistema/dispositivo de tecnología propietaria de mirroring del Smartphone, aunque esta opción es cada vez más minoritaria. Y en segundo y tercer lugar, aprovechar los ecosistemas creados por Google con Google Auto (para teléfonos Android) y por Apple, con Apple Car (para dispositivos ios). De esta forma, es posible la interacción con los teléfonos inteligentes de los pasajeros (ios, Android, o ambos simultáneamente), lo cual puede ofrecer ventajas muy interesantes frente a dispositivos que no dispongan de esta característica.

A continuación, aparece una tabla comparativa que enfrenta todas las soluciones comentadas anteriormente con el sistema que se pretende desarrollar en este TFG. En esta tabla aparecen las funcionalidades de los diferentes sistemas, así como su precio aproximado, con el fin de comparar (nunca competir) el presente proyecto con soluciones comerciales disponibles actualmente:

Nombre	Parámetros OBD	Control por voz	GPS	Control desde el volante	Cámara	Compatible Android Auto	Compatible Apple Car	Precio (aprox.)
OBD-pi	✓	✗	✗	✗	✗	✗	✗	90 €
Reproductor multimedia DIY	✗	✗	✗	✗	✗	✗	✗	90 €
i-Carus	✓	✗	✓	✗	Opcional	✗	✗	200 €
Carberry	✓	✗	✓	✗	Opcional	✗	✗	90 €
Tesla Infotainment	✓	✓	✓	✓	✓	✓	✓	Integrado de serie (desde 66.000€)
Volkswagen Entertainment System	✓	✓	✓	✓	Opcional	✓	✓	Integrado de serie (desde 11.000€)
Renault R-Link	✓	✓	✓	✓	✓	✓	✓	Integrado de serie (desde 12.000€)
Pioneer AVIC-8200NEX:	✓	✗	✓	✓	Opcional	✓	✓	1.400 €
Kenwood DDX9716BTS	✗	✗	Opcional	Opcional	✗	✓	✓	635 €
Sistema que se propone en este TFG	✓	✓	✗	✗	✗	✗	✗	110 €

Figura 10 – Tabla comparativa entre las opciones analizadas

4. Requisitos Funcionales y No Funcionales

En esta sección se van a recoger los Requisitos, tanto Funcionales, como no Funcionales, ordenados por importancia. Los tres Requisitos Funcionales más importantes a cumplir en la realización de este TFG son, en primer lugar, la conexión con el motor del vehículo para complementar y mejorar la información que se recibe durante la conducción, lo cual permite que ésta sea más eficiente. En segundo lugar el control del sistema mediante comandos de voz, ya que la seguridad al volante es algo primordial y estar manejando el sistema, bien mediante ratón y teclado, bien mediante la pantalla táctil puede suponer un riesgo. Y finalmente la integración real en un vehículo, para que el sistema tenga una utilidad real y práctica, y su realización no se limite a lidiar con problemas derivados del desarrollo de software. A continuación se muestran las tablas con los Requisitos Funcionales y No Funcionales:

Requisitos Funcionales:

Número	Requisito	Descripción
RF1	Conexión con el motor	Permitir la extracción y procesado de datos propios de la conducción como puedan ser velocidad actual o revoluciones por minuto.
RF2	Control por voz	Permitir el manejo del sistema de forma eficiente y sin necesidad de usar la pantalla táctil o un teclado para ello.
RF3	Integración en vehículo	Integración real en el vehículo del alumno.
RF4	Reproductor multimedia	Instalación de un reproductor multimedia (audio, videos…) que aproveche el sistema de altavoces del vehículo.

Figura 11 – Tabla de Requisitos Funcionales

Requisitos No Funcionales:

Número	Requisito	Descripción
RNF1	Coste reducido	Minimizar el presupuesto necesario para desplegar este sistema.
RNF2	La conducción no se verá comprometida	La seguridad del vehículo, así como la de sus ocupantes no se verá comprometida en ningún momento.
RNF3	Instalación simple	Cualquier persona será capaz de reproducir la instalación (tanto hardware como software) de este sistema.
RNF4	Utilización simple	Cualquier persona, sin necesidad de tener conocimientos de Informática, será capaz de utilizar este sistema.
RNF5	Diversos métodos de control	El sistema podrá controlarse tocando en la pantalla táctil, mediante comandos por voz o empleando ratón y teclado.
RNF6	Fácilmente escalable en funcionalidades	Resultará muy sencillo la adición de nuevas funcionalidades, tanto por el alumno, como por cualquier otra persona que así lo desee.
RNF7	Disponible de forma pública	La parte Software de este proyecto se dejará a disposición de quien lo desee en un repositorio de código.
RNF8	Soporte hardware de cara al futuro	Al emplear como "cerebro" del proyecto una Raspberry Pi se garantiza la disponibilidad y funcionamiento de este proyecto de cara al futuro.

Figura 12 – Tabla de Requisitos No Funcionales

5. Metodología

Para el desarrollo de las diferentes fases de este TFG se han empleado las denominadas Metodologías Ágiles, apoyadas sobre Prototipos Rápidos. De esta forma, para cada fase se ha empezado desarrollando un prototipo muy simple que poco a poco ha ido ganando en funcionalidades, hasta obtener el resultado final. Esto ha permitido que el alumno desarrolle el proyecto con gran eficiencia. Los tres bloques principales de los que se compone este trabajo han sido desarrollados secuencialmente, empezando por la parte de adquisición y procesado de parámetros del motor para a continuación, pasar al control por voz, y dejando para el final la integración en el vehículo. Se ha seguido este orden porque se considera el más lógico y eficiente, empezando por el nivel más bajo y cercano al hardware, para avanzar después hacia el software. Cada bloque no se ha empezado hasta que no ha sido completado el anterior, documentación incluida (se ha producido algún ligero solape entre documentación de un bloque y el comienzo del siguiente).

En las tres partes constitutivas de este proyecto se ha tenido en mente el Ciclo de Vida clásico de la Ingeniería del Software, y se ha prestado especial atención a las dos primeras fases de este, al Análisis y al Diseño. La fase de Análisis de cada una de las fases ha consistido en la recopilación de la información necesaria, y la adquisición los conocimientos necesarios para abordar cada una de las partes del proyecto. Durante la fase de Diseño, se han desarrollado diferentes Prototipos Rápidos para las dos primeras partes del proyecto, y un pequeño prototipo “físico” para la tercera parte, que ha permitido simular el lugar donde se iba a integrar la pantalla en el vehículo para poder trabajar en el circuito sin necesidad de estar dentro del vehículo. Algunos de los citados prototipos fueron desechados, pero otros se fueron refinando hasta formar la versión final.

También se realizaron diferentes prototipos en la fase de Diseño de la Interfaz de la aplicación centralizadora de funcionalidades. Al final, y después de consultar entre algunos conocidos, se escogió el que mejor se adaptaba a las necesidades del proyecto, sobre todo por simplicidad y facilidad de uso del mismo. El resto de prototipos serán reaprovechados en un futuro, ya que se desea que la interfaz pueda ser configurable por el usuario del sistema (aunque esto queda fuera de los límites de este TFG).

5.1 Herramientas

Las herramientas empleadas para el desarrollo del código necesario para este proyecto han tratado de agilizar este proceso lo máximo posible. Se ha trabajado mediante un cliente SSH para acceder a la Raspberry Pi, escribiendo el código (principalmente el código escrito en lenguaje PHP) en un computador de sobremesa con Windows y el entorno de desarrollo Adobe Dreamweaver, y una vez que el código estaba listo, se enviaba a la Raspberry Pi por medio de un servidor FTP configurado a tal efecto. De esta forma, el grueso del código pudo generarse de forma muy rápida con un buen entorno de desarrollo, mientras que los cambios/ajustes menores se realizaron de forma local en la Raspberry Pi por medio de SSH.

Para la depuración del código generado se han empleado diversos prototipos, desarrollados a tal efecto. Se ha simulado por ejemplo el comportamiento de la visualización de los datos provenientes del motor en situaciones muy extremas, simulando que el vehículo viajaba a 200km/h (algo que por supuesto no se ha puesto en práctica en la realidad). Para realizar esto, se ha diseñado un test de caja negra que imitaba el comportamiento del motor, por lo que al aplicarlo al sistema, este pensaba que realmente se encontraba conectado al coche, aunque no fuera así. También se ha experimentado con la interacción por voz, realizando diversas secuencias de comandos en diferentes ambientes (silencio, ruido de motor, conversación de fondo…) para comprobar su correcto funcionamiento. Como se comentará en el apartado 6.3 de esta memoria, también se han realizado diferentes pruebas de control mediante comandos en Inglés y en Español, obteniendo mejores resultados en el reconocimiento empleando el Inglés.

Finalmente, las herramientas empleadas en la instalación del sistema en el vehículo han sido las típicas que se puedan encontrar en un taller mecánico (aunque por supuesto sin tanta variedad ni especificidad en las herramientas como la que se pueda encontrar allí), varios tipos de destornilladores para retirar el frontal del salpicadero, pistola termofusible para fijar algunos componentes… Merece la pena comentar aquí, que también se han desarrollado diferentes programas en Python para automatizar la puesta en marcha del sistema en el vehículo.

5.2 Gestión del Proyecto. Diagrama de Gantt

En esta sección va a detallarse la organización temporal que se ha seguido para el desarrollo de este proyecto. Para ello se va a emplear un diagrama de Gantt que se detallará a continuación.

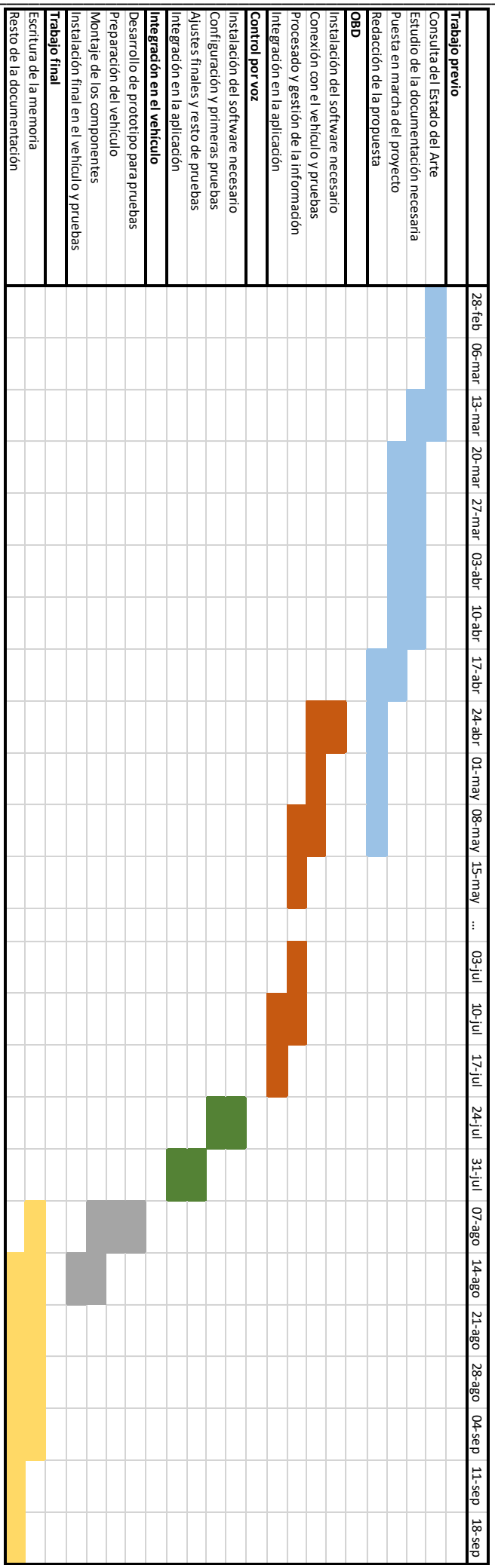
Como puede observarse en el diagrama adjunto al final de esta sección, además de las tres partes ya comentadas de las que se compone este TFG (Conexión con el Motor para extracción de datos, Control por Voz e Integración en el vehículo), se han representado también, el Trabajo previo realizado y el Trabajo Final necesario para la terminación del trabajo. Asimismo, cada uno de estas cinco partes ha sido dividido en sub-apartados, para así poder afinar lo más posible el tiempo que ha sido necesario para realizar cada parte del proyecto. Los datos (temporales) necesarios para la realización de este diagrama se han extraído de unas notas rápidas que el alumno escribía en forma de bitácora, cada vez que dedicaba algo de tiempo al desarrollo de cualquiera de las partes.

Analizando en detalle el diagrama, resulta evidente que el tiempo que se pudo dedicar en los meses de Febrero a Mayo (con mucha mayor carga lectiva), fue mucho menor que el dedicado en los meses de Julio (cuando se le dedicó la mayor parte del tiempo disponible por el alumno) y Agosto (se tuvo que empezar a compaginar con el estudio para exámenes de la convocatoria de Septiembre). Es por esto que las dos primeras partes (Trabajo previo y OBD) se extienden mucho más en el tiempo que, por ejemplo, las partes de Control por voz e Integración en el vehículo.

La columna en blanco colocada entre el 15 de Mayo y el 3 de Julio representa el parón que ha sufrido el proyecto por tener que dedicar el alumno el tiempo necesario al resto de asignaturas y a sus correspondientes exámenes en la convocatoria de Junio.

Por supuesto, y aunque no aparezca de forma explícita en el diagrama, al finalizar cada una de las tres fases principales del proyecto, se dedicaron multitud de horas a la realización de tests, tanto de cada parte por separado, como de la integración y comunicación entre ellas. Algunos de estos tests han sido comentados en la sección 5.1 de esta memoria.

Finalmente, y a modo de conclusión, sería interesante indicar que a pesar de que en el diagrama adjunto no se aprecia bien, las tres partes principales de este trabajo han supuesto un esfuerzo similar, lo que se traduce directamente en el número de horas dedicadas a cada una de ellas. Sin embargo, la cantidad de horas diarias que se pueden dedicar a un proyecto de estas características en los meses estivales, es de lejos mucho mayor que durante el periodo lectivo, con la consiguiente aceleración en las etapas finales. De esta forma, durante los meses de Julio y Agosto se dedicaron una media de entre 4 y 5 horas diarias al proyecto, algo que en periodo escolar resultaría imposible debido al resto de asignaturas que se deben cursar simultáneamente. Durante el citado periodo lectivo (meses de Febrero a Junio), se pudo dedicar de media menos de una hora diaria al proyecto, debido a los motivos anteriormente citados.


28-feb
06-mar
13-mar
20-mar
27-mar
03-abr
10-abr
17-abr
24-abr
01-may
08-may
15-may
...
03-jul
10-jul
17-jul
24-jul
31-jul
07-ago
14-ago
21-ago
28-ago
04-sep
11-sep
18-sep
Trabajo previo
Consulta del Estado del Arte
Estudio de la documentación necesaria
Puesta en marcha del proyecto
Redacción de la propuesta
OBD
Instalación del software necesario
Conexión con el vehículo y pruebas
Procesado y gestión de la información
Integración en la aplicación
Control por voz
Instalación del software necesario
Configuración y primeras pruebas
Ajustes finales y resto de pruebas
Integración en la aplicación
Integración en el vehículo
Desarrollo de prototipo para pruebas
Preparación del vehículo
Montaje de los componentes
Instalación final en el vehículo y pruebas
Trabajo final
Escritura de la memoria
Resto de la documentación

6. Arquitectura de Sistema

Como ya se ha comentado anteriormente, el presente TFG está compuesto por tres partes principales, dos de ellas íntegramente software (Adquisición de Datos del Motor y Control por Voz), mientras que la tercera, la integración en el vehículo, es algo más hardware. En este apartado se va a describir en primer lugar la interacción entre estas tres partes para, a continuación, detallar el funcionamiento de cada una de ellas por separado, además de la inclusión de las dos primeras en una aplicación que centraliza los servicios disponibles. Asimismo, en esta sección también se comentarán algunas funcionalidades extra incluidas en la citada aplicación.

6.1 Visión General del Sistema

A continuación, se adjunta un diagrama de alto nivel en el que se reflejan las tres partes constituyentes de este proyecto, así como las interacciones entre ellas:

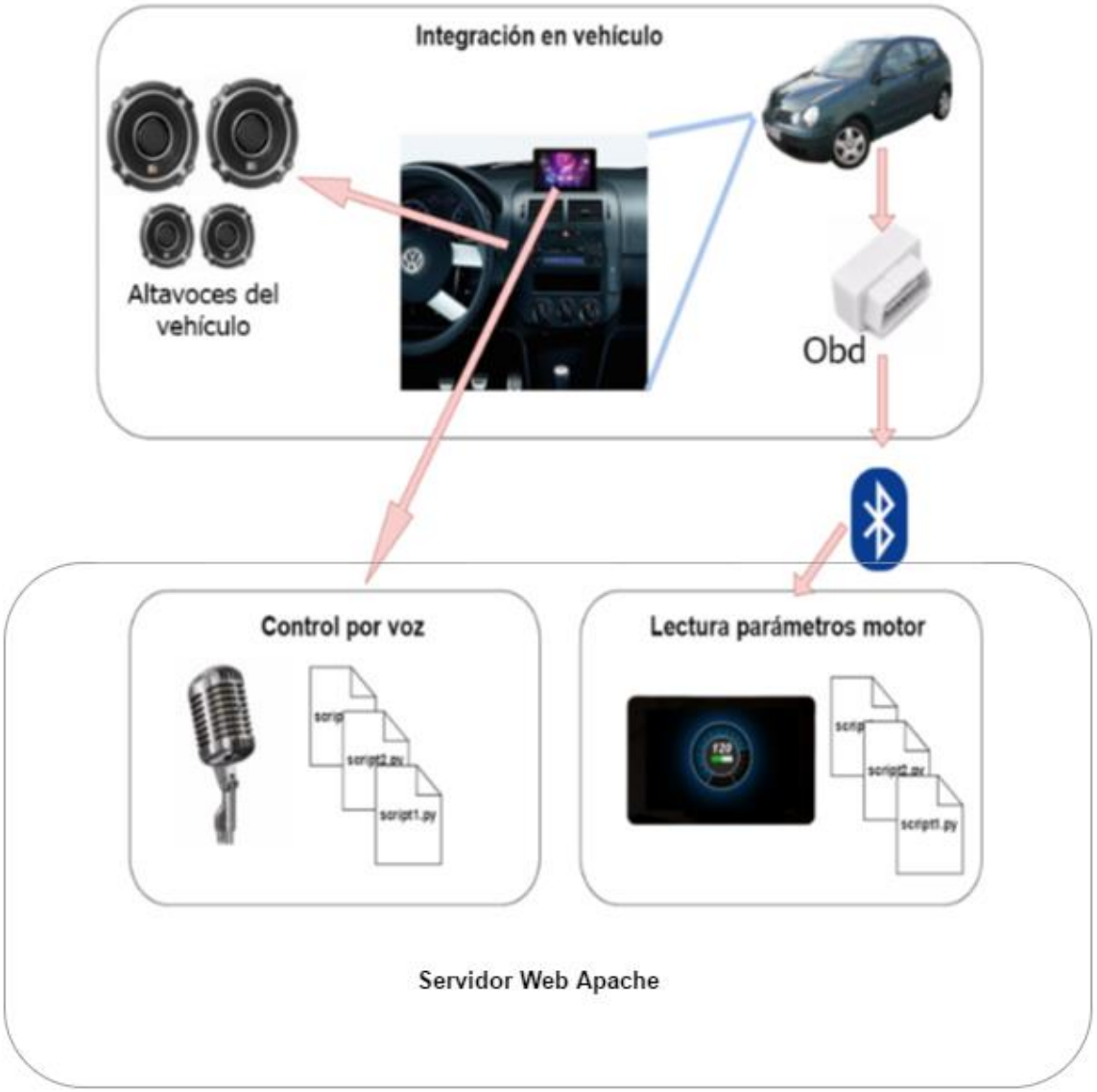


Figura 13 – Visión general del sistema

Como puede observarse en el diagrama anterior quedan reflejados todos los Requisitos Funcionales de este proyecto. Estos requisitos ya se detallaron en una sección previa, pero gracias a este diagrama se puede observar, desde muy alto nivel, cómo interaccionan entre ellos, así como poder distinguir a la perfección las partes constituyentes del presente proyecto.

6.2 Adquisición y Procesado de Datos del Motor

Si se atiende a la cronología del desarrollo, esta fue la primera de las partes constituyentes del proyecto en estar disponible. Esta parte es la que cubre el primer Requisito Funcional detallado en la sección correspondiente, es decir la conexión con el motor para obtener los parámetros típicos de la conducción. En el Anexo de esta memoria se explican pormenorizadamente las dos decisiones de diseño que se han tomado para la realización de esta parte, el mecanismo de Memcache, y el refresco de la página de visualización mediante AJAX, por lo que se avanzará rápidamente en estos dos puntos.

Para comunicar el sistema con el vehículo se utiliza el protocolo estándarizado OBDII ([14]). Este protocolo permite la monitorización y control completo del motor y otros dispositivos del vehículo mediante una serie de códigos, denominados PID's (parameter ID's). Actualmente se encuentra en su segunda versión. Cada PID tiene una misión concreta, desde indicar un posible fallo en uno de los pistones hasta indicar la velocidad actual a la que viaja el vehículo. Consultando el valor que almacenan algunos de estos PID's es posible averiguar parámetros como velocidad instantánea, o las revoluciones a las que se mueve el motor. Para ello es necesario conocer el PID exacto que se desea consultar, para lo cual existen tablas que relacionan el número de cada PID con el parámetro que monitorizan. La versión del protocolo OBDII disponible en el automóvil del alumno permite únicamente la lectura de estos valores de los PID's, no así el envío de órdenes para, por ejemplo, subir o bajar las ventanillas, o incluso arrancar el motor o pisar el freno de forma remota (lo cual podría llegar a ser bastante problemático desde el punto de vista de la seguridad).

La Raspberry Pi se conecta a la red ODBII mediante un dispositivo Bluetooth colocado en el puerto ODB del vehículo lo cual permite el envío de la información necesaria. Gracias a la conexión inalámbrica se puede situar la Raspberry Pi en cualquier parte del vehículo.

Es por este motivo que ha sido necesario instalar el software de gestión Bluetooth necesario para poder comunicarse con el citado adaptador OBD. El adaptador es un *Unotec OBDII Diagnostico* y el resto de sus características se pueden encontrar en [16]. El hecho de que la Raspberry Pi del alumno se tratara del modelo 3, dificultó ligeramente esta comunicación, al llevar el módulo Bluetooth integrado en la placa, en lugar de emplear un adaptador Bluetooth USB como en modelos anteriores (la conexión no se realizaba exactamente igual ya que también se conectó con una Raspberry Pi Model 2 y un adaptador Bluetooth USB para unas pruebas preliminares y resultó ser más sencillo con este modelo).

En segundo lugar, se va a hablar sobre la biblioteca Py-obd, disponible de forma libre en Internet. La principal característica de esta aplicación, es que es capaz de acceder a la mayoría de los PID's del protocolo OBDII (del cual se ha hablado anteriormente), con lo cual se hizo posible acceder a los parámetros del motor deseados, concretamente: Velocidad, medida en kilómetros por hora, Revoluciones del motor, medidas en revoluciones por minuto, y Aceleración, medida mediante la posición de la palanca del acelerador (en un porcentaje). Así, tras ajustar algunos parámetros de la biblioteca, y de reescribir algunas líneas, fue posible el acceso a dichos parámetros en tiempo real (aunque en formato texto en una terminal Linux, por lo que era necesario mostrarlos de una forma algo más funcional y de acuerdo al diseño de interfaces).

Es aquí donde fue necesario establecer la comunicación entre la biblioteca Py-obd y la parte PHP del proyecto, y es donde entra en juego la primera decisión de diseño descrita en detalle en el Anexo. Como se comentará en el citado apartado, finalmente se ha solucionado este problema encapsulando los datos extraídos del motor en peticiones HTTP muy ligeras, que se envían al servidor PHP, y ahí se almacenan en Memcache, que a grandes rasgos es un espacio reservado de memoria caché accesible desde distintos puntos de la arquitectura. Una vez que se han almacenado pueden ser accedidos de forma muy rápida por la página PHP que muestra y refresca los datos por pantalla. Esta decisión fue tomada debido a la gran cantidad de datos por unidad de tiempo necesarios para dar la impresión al usuario de estar visualizando información en tiempo real durante la conducción. El envío de estas peticiones HTTP con los datos instantáneos del motor se realiza cada 10ms, y en cada petición se envían cuatro valores en coma flotante (lo cual supone una tasa aproximada de 25,6 Kilobits/s). En el Anexo de esta memoria aparecen otras alternativas que se estudiaron para implementar esta comunicación. A continuación se muestra un diagrama de secuencia que detalla el camino que siguen los datos desde que son extraídos del motor hasta que son mostrados en tiempo real en la página PHP:

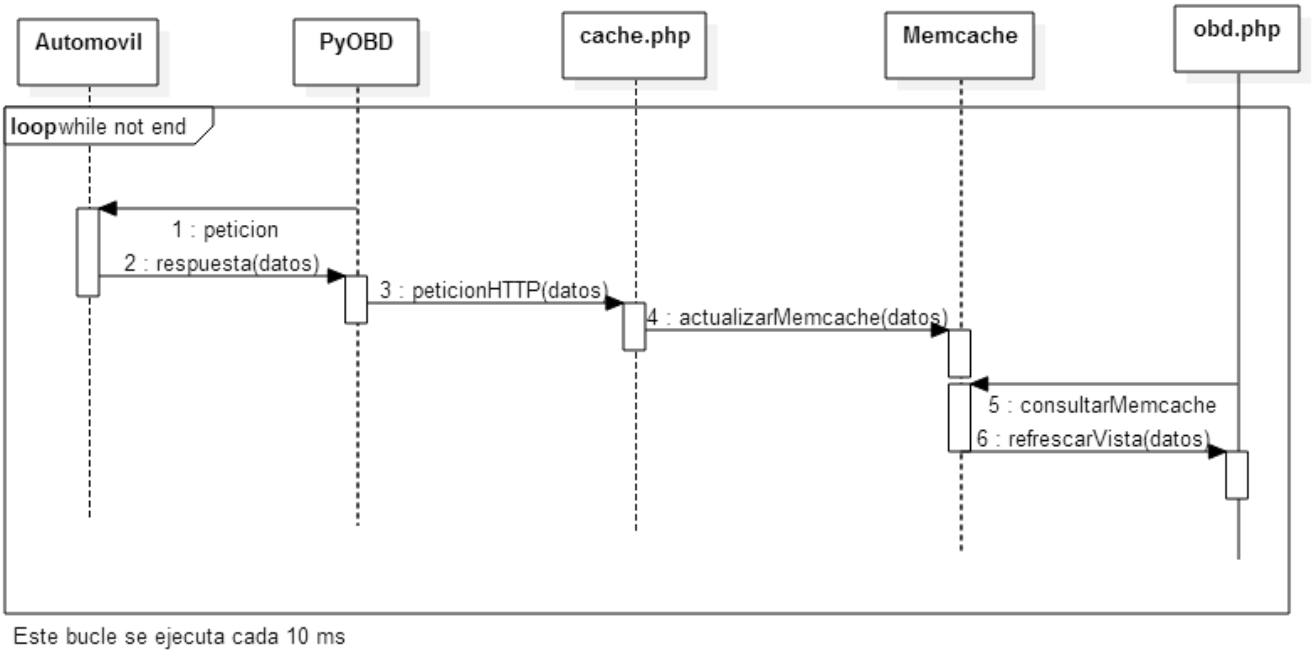


Figura 14 – Diagrama de secuencia de la comunicación con el motor

De nada sirve esto último, si la Vista no es capaz de ir actualizándose a medida que van llegando nuevos datos. Para ello se tomó la segunda decisión de diseño, explicada en detalle en el Anexo de esta memoria, el refresco mediante AJAX. Se empleó esta tecnología para poder refrescar la visualización de los datos del motor de forma muy rápida (al ritmo al que van llegando los nuevos datos a Memcache) y sin necesidad de recargar la página entera. Cada 10 ms, el servidor AJAX actualiza una pequeña parte de la página PHP con los datos provenientes del motor que se encuentran almacenados en Memcache. Se ha decidido emplear AJAX debido a la gran velocidad de refresco que debe tener la página de visualización.

El resultado obtenido finalmente, tras juntar todos estos elementos resulta bastante satisfactorio. Es posible replicar en la pantalla conectada a la Raspberry Pi la mayoría de los datos que aparecen en el cuadro de instrumentos del vehículo, además de algunos otros. A continuación, se muestran algunas fotografías del vehículo en funcionamiento y mostrando los datos correspondientes:

Figura 15 – Parte OBD en funcionamiento (Km/h)

Figura 16 – Parte OBD en funcionamiento (RPM)

Los problemas principales a los que hubo que enfrentarse en esta parte fueron principalmente la conexión entre la Raspberry Pi 3 y el adaptador OBD para acceder a los parámetros del motor (debido a la novedad de esta versión de integrar el módulo Bluetooth en lugar de depender de uno USB, lo cual hacía esta conexión mucho menos flexible), así como obtener una comunicación muy rápida y eficiente entre la biblioteca Py-obd y la parte PHP para su visualización (fue necesario explorar multitud de posibilidades antes de dar con el mecanismo de Memcache, las cuales se han explicado en el Anexo).

6.3 Control por Voz

La segunda parte del proyecto en estar operativa fue el Control por Voz. Asimismo, representa la parte que cubre el segundo Requisito Funcional detallado en la sección 4 de esta memoria. De nuevo aquí se describen brevemente las dos decisiones de diseño principales que se han empleado, ya que serán explicados en detalle en el Anexo de esta memoria. Estas decisiones son la Comunicación entre dos procesos por medio de un fichero, y el Estilo de comunicación asíncrono empleando un comando de activación y otro de fin. A continuación se muestra un diagrama de muy alto nivel en el que se muestra la idea principal del sistema de control por voz.

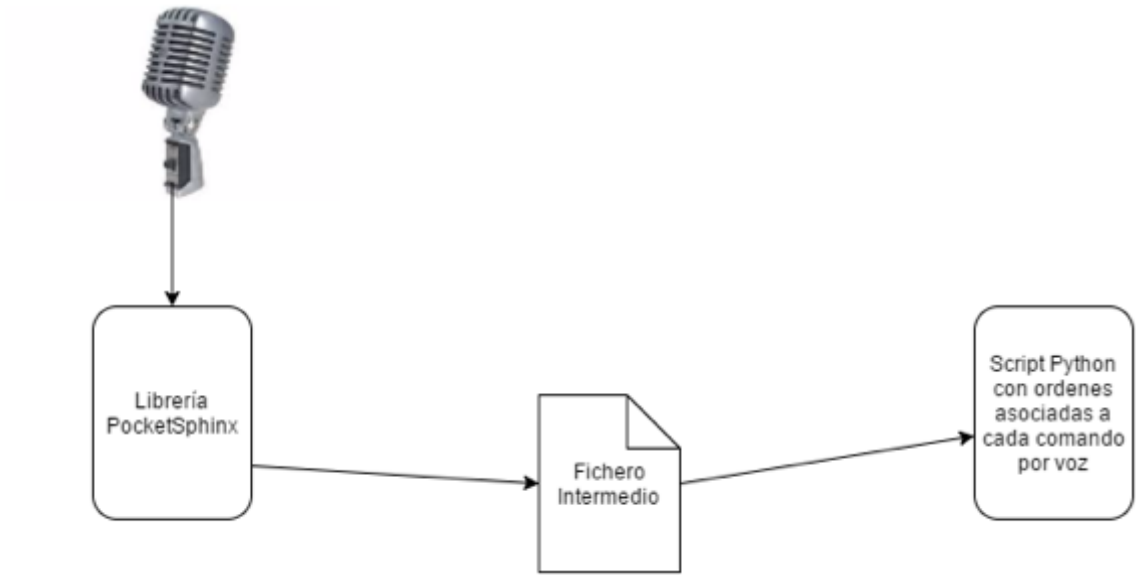


Figura 17 – Diagrama de alto nivel del procesado de los comandos de voz

Para implementar el control por voz se evaluaron diversas opciones, cuyas características se detallan en la tabla siguiente. El requisito principal para poder emplear alguna de estas librerías en el proyecto es que fuera capaz de funcionar sin conexión a Internet, lo cual limitó enormemente las posibilidades, dejando como única candidata la librería PocketSphinx. Si de cara al futuro se decide implementar una conexión a Internet en el vehículo se podría valorar el cambio a cualquiera de las otras opciones.

Nombre	Offline	Open-source	Comentarios	Referencia
PocketSphinx	Si	Si	No necesita una conexión a Internet para detectar palabras conocidas.	[4]
Jasper	No	Si	Muy configurable.	[16]
Google STT	No	Si	Prácticamente el mismo software que el implementado en teléfonos Android. Tiene límite diario de uso.	[17]
Voice Recognition by Oscar Liang	No	Si	Conectado permanentemente a la base de conocimiento Wolfram, lo cual permite ofrecer respuestas a algunas preguntas en lenguaje natural.	[18]

Figura 18 – Tabla Comparativa entre diferentes softwares de reconocimiento de voz

Una vez instalada y configurada esta biblioteca, se puede proceder al reconocimiento de las palabras conocidas por la misma, por medio de un micrófono USB. El alumno no dispone de este tipo de micrófonos (se dispone de un par con entrada auxiliar de 3.5mm), por lo que se ha empleado a modo de dispositivo de entrada una webcam USB con micrófono integrado, cuyo funcionamiento es igual que el de un micrófono a todos los efectos. La configuración de la entrada de audio no ha resultado para nada sencilla, debido principalmente a la escasa y en algunos casos contradictoria documentación disponible.

Después de realizar multitud de pruebas, ajustando parámetros tanto en la captura de la entrada de audio, como en la propia biblioteca PocketSphinx, se ha llegado a la conclusión de que el comportamiento de este reconocedor mejora considerablemente cuando las palabras a reconocer pertenecen al idioma Inglés. Con palabras en castellano, confundía habitualmente palabras como "Entrar" y "Motor", por lo que se ha optado por comunicarse con el sistema en inglés. A continuación aparece la lista de palabras que el sistema es capaz de reconocer, así como la acción que se espera que desempeñe:

Ok	Comando de activación
Polo	
End	Comando de finalización
Right	Simular una pulsación de la tecla Derecha
Left	Simular una puslación de la tecla Izquierda
Up	Simular una pulsación de la tecla Arriba
Down	Simular una pulsación de la tecla Abajo
Enter	Simular una pulsación de la tecla Intro
Scape	Simular una pulsación de la tecla Escape
Music	Acceder a la parte multimedia desde cualquier pantalla
Radio	Acceder a la parte multimedia desde cualquier pantalla
Motor	Acceder a la parte OBD desde cualquier pantalla
Shutdown	Apagar el sistema desde cualquier pantalla

Figura 19 – Tabla de comandos disponibles

Se han escogido estas palabras para el control por voz y no otras, ya que resulta necesario desplazarse por el menú de opciones en la aplicación que centraliza todos los servicios disponibles. De esta forma, es posible desplazarse en las cuatro direcciones, seleccionar e ir hacia atrás mediante comandos por voz, además de integrar algunos accesos directos a las opciones más importantes como puedan ser la Adquisición de datos del motor o el Reproductor multimedia.

Cuando el sistema fue capaz de reconocer una serie de palabras, se hizo necesario asignar una acción o significado a cada una de ellas. Para ello, se ha desarrollado *ad hoc* un script en Python que realiza exactamente esa misión. Aquí es donde entra en juego la tercera decisión de diseño explicada en profundidad en el Anexo, la comunicación entre la biblioteca PocketSphinx y este script mediante el uso de un fichero intermedio. Esta decisión de diseño basa su funcionamiento en un fichero en el que PocketSphinx escribe cada vez que reconoce una palabra y desde el que el citado script en Python lee constantemente. Cuando el script Python detecta que se ha reconocido una nueva palabra, le asigna una de las opciones que tiene implementadas. Es en este script donde se ha implementado el estilo de comunicación con palabra clave de inicio y de fin, que representa la cuarta y última decisión de diseño. Esta decisión se detalla en el Anexo de esta memoria.

Se ha decidido implementa también una funcionalidad interesante, que permite indicar al usuario mediante un sistema de iconos en pantalla cuándo el sistema se encuentra "escuchando", es decir, el tiempo que transcurre entre que se pronuncia el comando de activación y el de finalización. De esta forma, los usuarios pueden saber en todo momento si el comando por voz que acaban de pronunciar ha sido entendido o no por el sistema.

Las opciones disponibles en este script se corresponden con las funcionalidades de la aplicación que centraliza todos los servicios del Sistema (y de la cual se hablará un poco más adelante), es decir: Posibilidad de apagar el sistema, acceder directamente a la parte OBD, acceder a la parte multimedia, así como desplazarse en las cuatro direcciones en los menús, aceptar la opción seleccionada e ir hacia atrás. Algunas de estas opciones han sido implementadas mediante la ejecución de comandos en bash dentro del propio script Python (por ejemplo, apagar el sistema o acceder a la parte multimedia), sin embargo, para el desplazamiento por los menús, seleccionar opción e ir hacia atrás, se ha empleado la biblioteca Uinput, que se encarga de simular mediante software la pulsación de teclas en un teclado. De esta forma, si se desea ir hacia la derecha, el comando por voz "Right" se mapea con una pulsación en la tecla derecha del teclado, ofreciendo así mucha libertad de movimiento dentro de los menús de la aplicación.

Los principales problemas que han aparecido en esta parte se han debido principalmente a la captura de la entrada de audio mediante un micrófono (Webcam USB en este caso). Se tuvieron que probar diversas opciones (la mayoría de las cuales referidas a una Raspberry Pi 2, no el modelo 3 como la del alumno), hasta que al final, después de configurar algunos parámetros, se consiguió grabar y reproducir sonido. Resulta curioso que una característica tan extendida en otros sistemas (computadores de sobremesa o dispositivos móviles) resulte tan complicada en un dispositivo de las características de una Raspberry Pi. Una vez superado esto, el resto no supuso grandes dificultades.

6.4 Aplicación que centraliza los servicios disponibles

Con las dos partes anteriores finalizadas, se hizo necesaria su integración dentro de una aplicación desde la que poder acceder a ambas, además de ofrecer algunas otras opciones que se detallarán a continuación. Esta aplicación ha sido desarrollada en lenguaje PHP y desplegada sobre el mismo Servidor Web en la que se tenía alojada la parte PHP de la adquisición de datos del motor (se trata de un servidor Apache que corre localmente en la Raspberry Pi).

La aplicación consta de nueve opciones, algunas de las cuales aún no están disponibles, pero se ha decidido colocarlas ya para ahorrar tiempo en el futuro cuando se decida implementar dichas funcionalidades. Las opciones que se ofrecen en el momento de la redacción de esta memoria son: Apagar el sistema mediante un botón, acceso completo al Sistema Operativo Raspbian si el usuario lo desea, acceso a Internet (siempre que la Raspberry Pi disponga de un punto de acceso a Internet), un reproductor para disfrutar de contenidos multimedia, y finalmente, el acceso a la información proveniente del motor. El control por voz queda integrado a la perfección en esta aplicación, y permite su manejo sin necesidad de tocar la pantalla táctil o emplear ratón y teclado.

La aplicación al completo ha sido diseñada para poder ser controlada mediante pulsaciones simples de ratón (o en la pantalla táctil) o mediante un teclado. Esto es así debido a que los elementos necesarios se han hecho “pinchables” y, además, todos ellos constituyen una suerte de “carrusel” por el que el usuario se puede ir desplazando si decide usar teclado. Además, debido a esto último, el control por voz permite a la perfección el manejo de la misma, ya que se han conseguido generar pulsaciones de teclado por software frente a algunos eventos de voz.

El aspecto de la aplicación diseñada aparece en las siguiente imágen, y debido a la forma en la que se ha desarrollado, resultaría muy sencillo de cambiar su interfaz estética por otra que pueda gustar más al usuario. Aunque aún no esta disponible, podría ser interesante disponer entre varias “skins” o aspectos configurables e intercambiables por el usuario.

Figura 20 – Interfaz de la aplicación desarrollada

El desarrollo de esta parte no ha supuesto grandes problemas al alumno, más allá de descubrir la biblioteca Uinput para simular pulsaciones de teclado mediante software.

6.5 Integración en el vehículo

Una vez que todas las partes anteriores quedaron terminadas se comenzó a integrar el sistema en el salpicadero de un vehículo real, concretamente en el del alumno, un Volkswagen Polo de 2004. Esta es la parte correspondiente al tercer Requisito Funcional que se detalló en la sección 4 de esta memoria. Las partes "físicas" a colocar en el vehículo fueron principalmente la pantalla táctil y la Raspberry Pi, además claro de todos los cables necesarios, desde la alimentación de ambos dispositivos, hasta la salida de audio auxiliar para que el sonido salga por los altavoces del coche (la radio del vehículo dispone de entrada auxiliar de 3.5mm).

En la siguiente foto puede apreciarse cómo era el salpicadero del vehículo escogido antes de empezar a colocar ningún componente.

Figura 21 – Aspecto inicial del salpicadero

La primera idea fue la de colocar la pantalla táctil en el frontal del salpicadero, en el hueco 2DIN retirando para ello la radio y la bandeja portaobjetos y ocupando el hueco que dejan. Sin embargo, el inconveniente de esta aproximación era donde colocar entonces con la radio del vehículo, ya que era necesaria en el proyecto si se deseaba que el sonido saliera por los altavoces del vehículo (por contar con el citado conector de 3.5mm). Además, era necesario desviar demasiado la vista de la carretera para mirar hacia la pantalla táctil, lo cual podría poner en riesgo la conducción.

Así que finalmente se optó por colocar la pantalla táctil en el cajón portaobjetos de la parte superior del salpicadero. La principal dificultad de esta segunda idea, era la necesidad de variar la posición de la pantalla para poder levantarla y recogerla (como sucede en algunos coches de alta gama), ya que no se quería dejar fija y siempre visible. Para ello, se empezó planteando un circuito eléctrico que moviera un motor, primero en un sentido de giro para levantar la pantalla, y posteriormente en el sentido contrario para recogerla.

Se realizaron dos implementaciones distintas de este circuito, variando los componentes electrónicos que lo constituían. Esto fue así debido a que la primera aproximación no funcionaba según lo esperado, ya que sufría una pequeña realimentación en uno de sus componentes que hacía que el motor girara en un sentido, pero no en el contrario. Es importante recalcar en este punto, la inestimable ayuda que el alumno recibió de manos de Eduardo Jarabo Baún, de la empresa "EdJar Instalaciones", tanto en el asesoramiento a la hora de adquirir el material, como a la hora del montaje del circuito. A continuación, se va a pasar a describir el funcionamiento de las dos versiones de este pequeño circuito:

El primer circuito consistió en la construcción de una pequeña placa integrada en la que soldaron diferentes componentes. Concretamente, estos componentes fueron dos relés, empleados uno para controlar el giro del motor en un sentido, y el otro para controlarlo en el otro sentido, así como un fusible y un portafusible para proteger el circuito de posibles picos de tensión, dos finales de carrera, empleados para que al llegar a la posición deseada el motor se pare, y un par de botones, uno para activar el giro en un sentido y otro para el otro sentido. Como se ha comentado anteriormente, uno de los relés sufría una pequeña realimentación, por lo que dejaba de funcionar impidiendo el giro en uno de los sentidos. A continuación, se puede observar una foto de este primer circuito desarrollado.

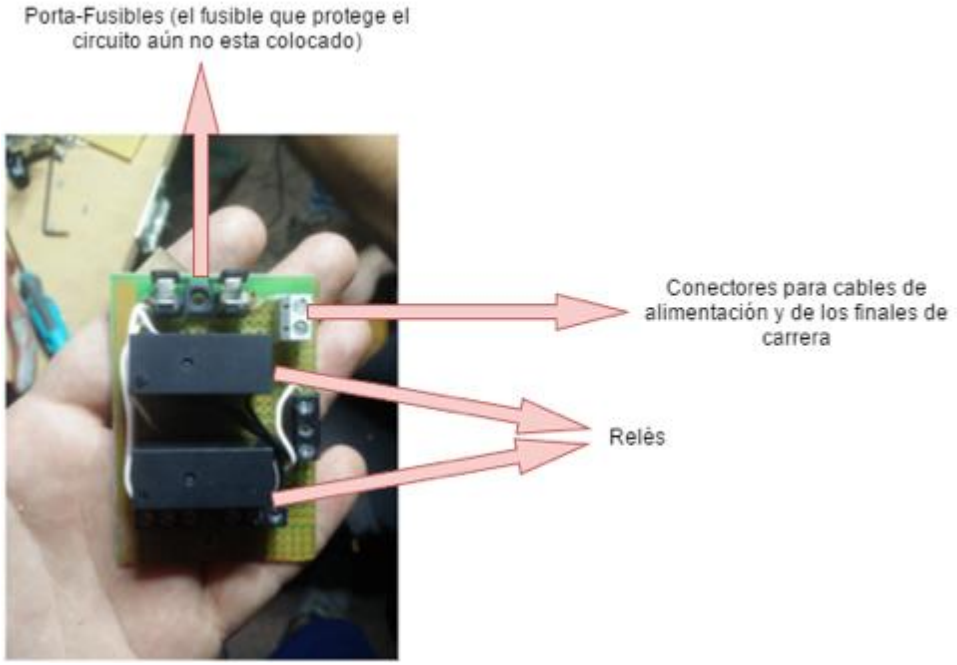


Figura 22 – Primer circuito (vista de frente)

Figura 23 – Primer circuito (vista trasera)

Para el segundo circuito se decidió simplificar todo lo posible el primer diseño y se pudieron reutilizar algunos de sus componentes. Este nuevo diseño se basa en el uso de un conmutador de tres posiciones, que permiten el giro del motor en un sentido o en otro y su parada, así como dos finales de carrera, empleados con el mismo fin que en la primera versión, es decir, para que el motor, al llegar a un tope se pare y no continúe girando indefinidamente. Seguidamente se adjunta una foto de esta segunda versión.

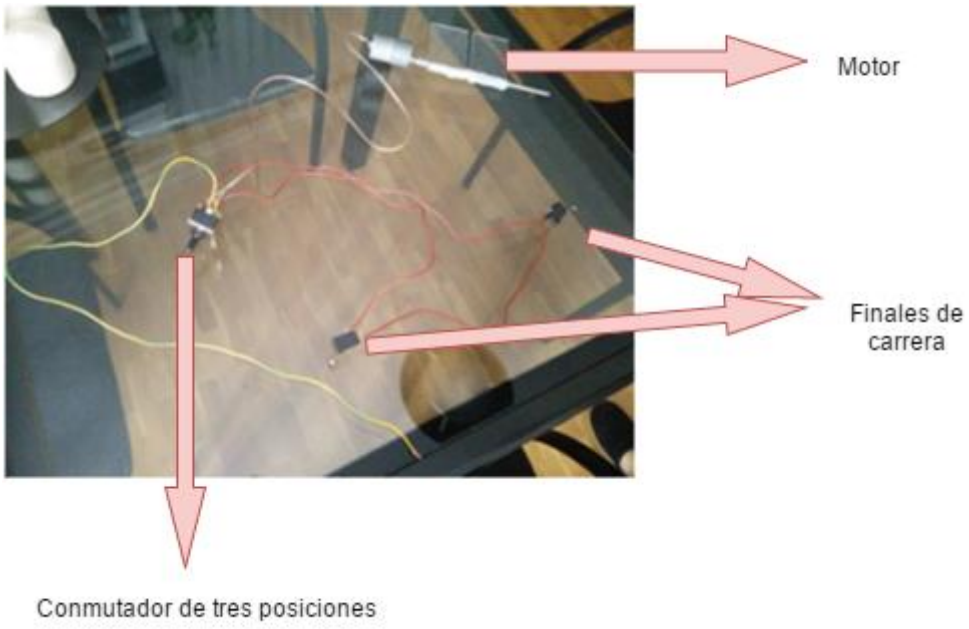


Figura 24 – Segundo circuito

Tanto el primero como el segundo diseño se alimentan igual, mediante un par de cables extraídos de la caja de fusibles del vehículo (estos cables se instalaron en un taller certificado debido al riesgo que supone operar en esta parte del automóvil, además de los escasos conocimientos del alumno en este campo). Este par de cables ofrecen 5 voltios de tensión que permiten alimentar tanto el circuito comentado como un segundo adaptador de mechero que se ha colocado para alimentar la Raspberry Pi y la pantalla por medio de USB.

Por supuesto, uno de los primeros pasos en la realización de este trabajo fue la medición del espacio disponible y el reparto de los componentes en los diferentes lugares que ocuparían más adelante, así como la construcción de un pequeño prototipo "físico" sobre el que poder colocar y ajustar las diferentes piezas sin necesidad de encontrarse el alumno en el vehículo.

Una vez construido el circuito en el exterior del vehículo, y habiendo comprobado su correcto funcionamiento, se pudo empezar a colocar en el sitio que iba a ocupar. Para ello, fue necesario desmontar la totalidad del frontal, desatornillando multitud de tornillos y separando varias pestañas. En este punto, el trabajo se volvió bastante incómodo, debido al reducido espacio de trabajo, y a la poca maniobrabilidad de la que se disponía en el interior del vehículo. Sin embargo, poco a poco los componentes del circuito fueron encajando en sus posiciones y tras muchos ajustes, se consiguió que el circuito funcionara colocado en el salpicadero del vehículo. En este punto aún no se había colocado la pantalla táctil ni la Raspberry Pi en su lugar definitivo. A continuación se muestra una foto del montaje:

Figura 25 – Montaje del circuito en el vehículo

Después de algunas pruebas de funcionamiento, se llegó a la conclusión de que era fundamental reducir la velocidad de giro del motor, ya que, a pesar de que el motor giraba a muy pocas revoluciones por minuto (80 concretamente), la amplitud del giro necesario era muy pequeña, apenas con un cuarto de vuelta bastaba para levantar la pantalla los 90 grados deseados. Sin esta reducción de velocidad, se corría demasiado riesgo, ya que la pantalla táctil podría golpear en el salpicadero y, al tratarse de un componente delicado, podría romperse y dejar de funcionar. Es por eso que, después de barajar diferentes opciones, se optó por adquirir un nuevo componente, concretamente un reductor de velocidad para motores de Corriente Continua. Este reductor se puede encontrar en [19].

Una vez que todos los componentes se encontraban en su sitio, se empezaron a realizar diversas pruebas; se probó el sistema de giro con y sin la pantalla táctil para que se ajustara bien al espacio disponible, se configuró adecuadamente el reductor de velocidad para que el motor girara de forma correcta, se buscaron posibles vibraciones con la pantalla desplegada y el coche en movimiento… Se muestran a continuación varias imágenes del sistema final, integrado en el vehículo del alumno:

Figura 26 – Resultado final (pantalla plegada)

Figura 27 – Resultado final (pantalla levantada)

Durante la realización de esta parte del TFG se han presentado varios problemas, el primero de ellos fue la pequeña realimentación de la primera versión del circuito, lo cual retrasó enormemente todo el resto de esta parte. El hecho de trabajar en un entorno "físico" de muy reducidas dimensiones provocó también algún quebradero de cabeza al alumno. Y finalmente la necesidad de reducir la velocidad de giro del motor tampoco facilitó nada las cosas. Finalmente, y tras no poco esfuerzo, se ha conseguido un sistema de giro para la pantalla táctil del proyecto que queda perfectamente integrado en el salpicadero, y además se han obtenido una serie de valiosos conocimientos, no directamente relacionados con el terreno informático, lo cual siempre resulta muy satisfactorio.

7. Conclusiones

Se ha implementado un sistema de Infotenimiento que cumple con creces los objetivos planteados al comienzo del proyecto. Estando todo terminado, y finalmente en su sitio, se puede decir que las sensaciones generales han sido realmente positivas. La integración e interacción entre las partes constitutivas de este proyecto, producen una enorme sinergia entre ellas, dotando de un gran valor añadido al producto final.

Se han adquirido multitud de conocimientos, principalmente en el terreno informático al tener que tratar con protocolos desconocidos para el alumno (OBDII), o software novedoso con el que no se ha estado en contacto durante los cuatro años que ha durado el Grado. Asimismo, y dada la naturaleza de este proyecto, también se han adquirido una serie de conocimientos, no directamente relacionados con la Informática, como por ejemplo algunas competencias básicas de electrónica que serán de enorme utilidad para el alumno en un futuro. Debido a su envergadura, el presente se trata de uno de los proyectos más grandes al que el alumno se ha enfrentado hasta ahora. Este proyecto también ha permitido redescubrir la enorme importancia de las primeras fases de la Ingeniería del Software, el Análisis y el Diseño. Generalmente son fases a las que no se les dedican los recursos necesarios, y en este caso, planificando adecuadamente desde un primer momento, se ha podido ahorrar una increíble cantidad de tiempo.

En cuanto a las dificultades encontradas, el principal escollo a salvar han sido la calibración y ajustes que han sido necesarios en la fase de integración del sistema en el vehículo. La escasa experiencia en temas electrónicos, así como la incomodidad del lugar de trabajo, dificultaron bastante esta tarea, sin embargo, el hecho de tener que lidiar con un problema en un entorno "físico", al que el alumno no está acostumbrado, ha sido realmente interesante y muy aprovechable. El resto de dificultades, que aparecieron en las otras dos partes del proyecto (en las partes "más software"), fueron resueltas con mayor o menor esfuerzo.

Una de las primeras conclusiones a las que el alumno ha llegado, ha sido la de que, dedicando el tiempo y esfuerzo necesarios, se pueden lograr resultados realmente buenos. Se ha conseguido desarrollar (con ayuda de algunas bibliotecas ya implementadas) un sistema de Infotenimiento que dista poco de las soluciones comerciales integradas en los vehículos más modernos o de alta gama, todo ello a un precio muchísimo más reducido, y en muchos aspectos mejor y más flexible que aquellos.

La segunda conclusión es que para proyectos de un tamaño similar a este y más grandes, resulta de vital importancia reutilizar trabajo ya disponible, ya que permite avanzar mucho más rápido y evita el "tener que reinventar la rueda". Por supuesto, la utilización de estas bibliotecas debe realizarse de forma ética, indicando en todo momento a sus autores y respetando su Licencia si la tuviera, como aquí se ha hecho. Si en el tiempo dedicado a este TFG hubiese sido necesario implementar cualquiera de las dos bibliotecas de terceros que se han empleado (Py-obd para el acceso a los datos del motor y PocketSphinx para el reconocimiento de voz), el proyecto en lugar de estar compuesto por tres partes, a duras penas hubiese estado compuesto por una de ellas.

Finalmente, y como ya se dijo en las primeras secciones, el alumno va a seguir desarrollando nuevas funcionalidades para el sistema de cara al futuro. Debido a la arquitectura desarrollada, donde la flexibilidad es un pilar fundamental, se espera que, una vez desarrolladas estas nuevas características, su integración en el sistema no provoque demasiados quebraderos de cabeza. Es una verdadera lástima no tener aún disponible alguna característica más, pero los recursos temporales son finitos, y se considera que se ha realizado una gestión bastante eficiente de los mismos.

8. Bibliografía

[1] http://www.eleconomista.es/ecomotor/motor/noticias/6887322/07/15/Cuantos-coches-hay-en-Espana-El-parque-crecio-en-2014-por-primera-vez-desde-2012.html

[2] http://www.elmundo.es/motor/2016/04/07/570625cee2704efc4c8b458d.html

[3] https://github.com/Pbartek/pyobd-pi

[4] http://cmusphinx.sourceforge.net/

[5] http://www.instructables.com/id/OBD-Pi/

[6] http://www.popularmechanics.com/cars/how-to/a15446/diy-touchscreen-dashboard-raspberry-pi/

[7] http://i-carus.com/

[8] http://www.carberry.it/

[9] https://www.tesla.com/en_EU/

[10] https://www.volkswagen.es/es.html

[11] http://www.renault.es/

[12] https://www.pioneerelectronics.com/PUSA/Car/GPS-Navigation/AVIC-8200NEX

[13] http://www.kenwood.es/car/multimedia/receptores/DDX9716BTS/?view=details

[14] https://es.wikipedia.org/wiki/OBD#OBD_II

[15] https://www.pccomponentes.com/unotec-obdii-diagnostico-para-coche-bluetooth-pc-android

[16] http://jasperproject.github.io/

[17] https://cloud.google.com/speech/

[18] https://oscarliang.com/raspberry-pi-voice-recognition-works-like-siri/

[19] https://www.amazon.es/CEBEK-Regulador-Velocidad-Motores-Cargas/dp/B017WE9POI/ref=sr_1_2?ie=UTF8&qid=1474645131&sr=8-2&keywords=regulador+de+velocidad+cebek

9. Anexos

9.1 Explicación en detalle de las decisiones de diseño

En esta sección se va a describir con detalle las decisiones de diseño que se han tomado durante la realización de este trabajo. Acompañando a la explicación de cada una de ellas se adjunta uno o varios diagramas explicativos, con el fin de dejar lo más claro posible su funcionamiento. Las dos primeras decisiones explicadas en esta sección se corresponden a la parte de adquisición de datos del motor, mientras que las dos últimas a la parte de control por voz. Estas decisiones ya han sido comentadas brevemente en la sección de Arquitectura del Sistema, aunque ahora se entra mucho más en detalle en lo referente a su funcionamiento.

9.1.1 Comunicación con el puerto OBD

En primer lugar, se va a detallar la decisión de diseño que se ha empleado para la comunicación entre la Raspberry Pi y el motor del automóvil. En un apartado anterior de esta memoria, se han detallado algunas de las características de esta comunicación (protocolo, software empleado, etc…). La comunicación y adquisición de los datos del motor la realiza una biblioteca escrita en Python, llamada Py-obd, que ha sido encontrada en la web y adaptada a las necesidades del alumno. El problema principal que tuvo que solventarse en esta parte fue el envío de los datos adquiridos por dicha biblioteca a la aplicación que los procesa y muestra, la cual está escrita en PHP. A modo de resumen, el problema consiste en comunicar los datos generados (adquiridos del motor) por un servicio Python corriendo en segundo plano, con una página PHP que debe refrescarse constantemente para mostrar los parámetros de la conducción en tiempo real. Pues bien, esta comunicación no resultó para nada sencilla. Se intentaron diversas opciones, (algunas de las más representativas se recogen cronológicamente en la tabla adjunta) hasta que al final se descubrió un mecanismo denominado Memcache, el cual permite almacenar variables en Memoria Cache software para así poder acceder a ellas desde diferentes puntos de la arquitectura. Es decir, permite la creación de un “espacio” en memoria cache mediante software, en el cual se pueden almacenar variables para ser compartidas entre diferentes entidades, en este caso concreto, dos páginas PHP.

Nombre de la opción	Descripción	Ventajas aparentes	Problemas encontrados
Separar la biblioteca Py-obd en dos partes	La primera parte sería la encargada de realizar la conexión con el puerto OBD y solo se ejecutaría una vez. La segunda parte realizaría peticiones instantáneas al motor y se ejecutaría dentro de un bucle en la página PHP.	Sencillez debido a la estructura original de la biblioteca Py-obd.	Era necesario enviar un objeto Python muy complejo de una parte a la otra. La única forma de hacerlo era serializando dicho objeto y Python no permite la serialización de objetos tan complejos.
Enviar los datos desde la biblioteca Python al PHP por medio de una petición HTTP	Los datos se enviarían encapsulados en una petición HTTP que sería recogida por el servidor PHP para así poder actualizar la visualización de los datos.	Velocidad de respuesta ofrecida por las peticiones HTTP. Sencillez al implementar tanto en Python como en PHP.	El servidor PHP no sabía qué hacer con la petición HTTP que recibía, ya que no era capaz de actualizar con esos datos una página (petición) que ya había resuelto.
Enviar los datos desde la biblioteca Python al PHP por medio de una petición HTTP y una variable de sesión	Los datos se enviarían encapsulados en una petición HTTP que actualizaría una variable de sesión encargada de actualizar la visualización de los datos.	Velocidad y sencillez de las peticiones HTTP. Sencillez al implementar tanto en Python como en PHP.	Por motivos de seguridad, la sesión de PHP no se comparte entre diferentes peticiones. Es decir, existían dos sesiones diferentes incapaces de comunicarse.
Comunicar la biblioteca Python y el servidor PHP mediante sockets	La biblioteca Py-obd crearía y abriría el socket, enviando los datos a medida que los fuera obteniendo. El PHP se mantendría a la espera de que llegaran peticiones para procesarlas y actualizar la vista.	Comunicación aparentemente sencilla entre procesos.	Se presentaron aquí diversos problemas de permisos, ya que el usuario dueño de los dos servicios comunicados por el socket debía ser el mismo, algo que no era así, y no fue posible modificarlo.
Enviar los datos desde la biblioteca Python al PHP por medio de una petición HTTP y Memcache	Esta idea es muy similar a la opción con petición HTTP y variable de sesión, pero con un mayor nivel de abstracción, ya que sí que permitía la comunicación de variables entre dos sesiones diferentes.	Velocidad y sencillez de las peticiones HTTP. Sencillez al implementar tanto en Python como en PHP.	Este fue el método elegido. Lo más difícil de esta opción fue descubrirla, debido a la escasa documentación de la que dispone. También hubo un pequeño problema de versiones desactualizadas que se pudo resolver fácilmente.

Figura 28 – Tabla comparativa de opciones de comunicación Python-PHP

Detallando el proceso de intercambio de información con la solución alcanzada, se tiene en primer lugar la biblioteca Py-obd que, al ser adaptada a las necesidades del alumno, se encuentra corriendo en segundo plano. Este proceso se lanza cuando se entra en la opción correspondiente de la aplicación desarrollada que centraliza los servicios disponibles. Esta biblioteca se encarga de crear la conexión con el motor empleando un adaptador Bluetooth conectado al puerto OBD del vehículo y a la Raspberry Pi. La citada biblioteca se encarga de realizar peticiones de datos al puerto OBD del vehículo cada 10 milisegundos. Los datos obtenidos en cada petición al motor se encapsulan en una petición HTTP que se envía a una página PHP que procede a su procesado. En esta primera página PHP, se introducen o actualizan los datos recibidos en la petición HTTP y se almacenan en Memcache, con el objetivo de que sean accesibles desde otra página PHP que será

la encargada de mostrarlos. Esta segunda página PHP que visualiza los datos debe refrescarse constantemente para actualizar los datos que se muestran en cada momento. Este refresco se realiza mediante tecnología AJAX, y es la segunda decisión de diseño que se va a detallar. A continuación, aparecen un par de diagramas que detallan lo comentado hasta ahora, el primero de ellos muestra una visión de alto nivel de la parte de Conexión con el Motor del proyecto, mientras que el segundo representa como se ha realizado la comunicación por medio de un Diagrama de Secuencia.

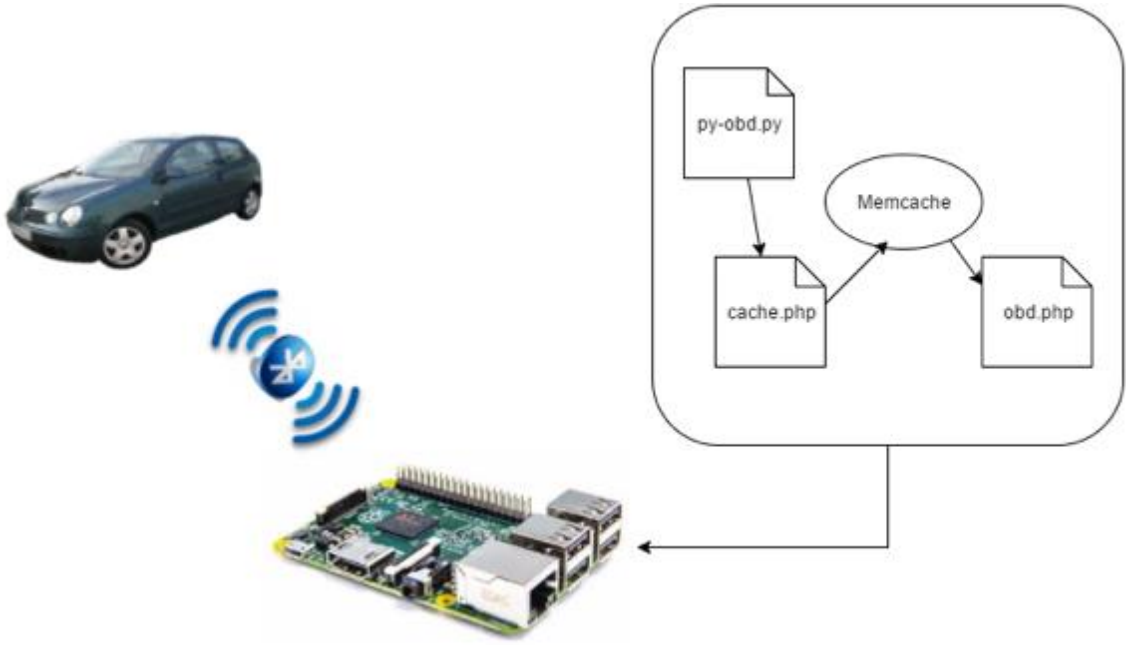


Figura 29 – Diagrama de alto nivel de la comunicación con el motor

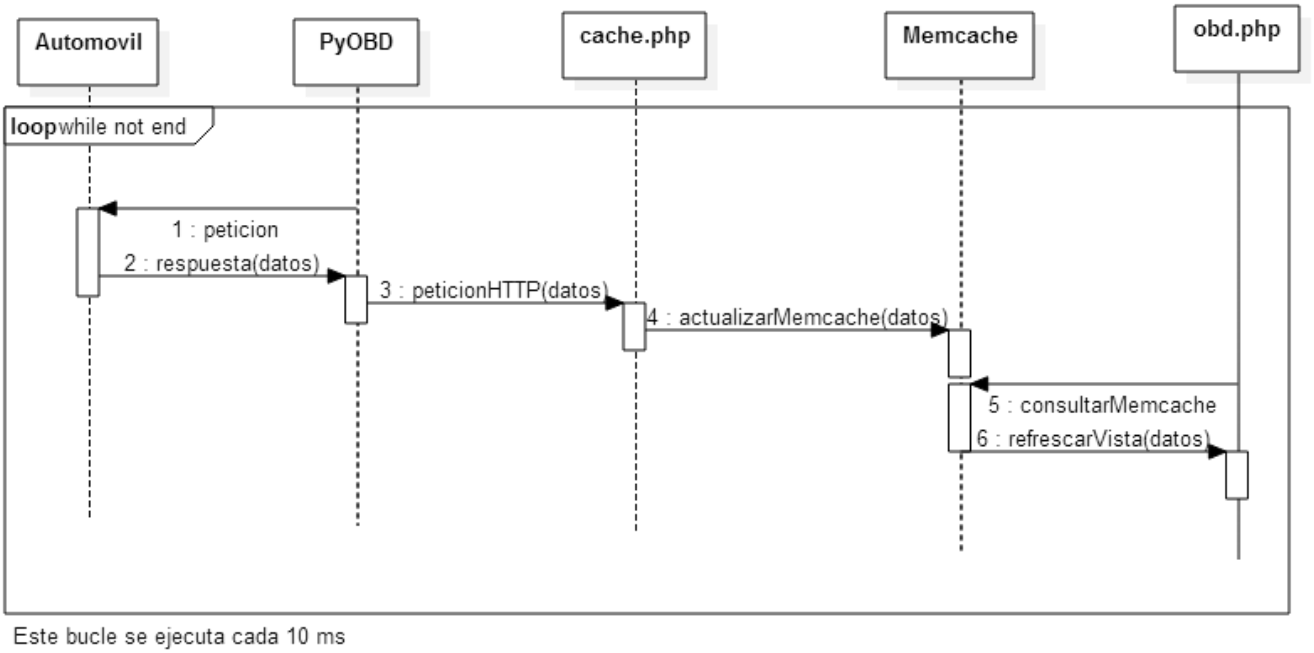


Figura 30 – Diagrama de secuencia de la comunicación con el motor

Esto representa la parte del Modelo de la arquitectura Modelo-Vista-Controlador que se explicará en la segunda decisión de diseño.

9.1.2 Refresco de la página PHP de visualización de datos

Como se ha comentado en la decisión de diseño explicada anteriormente, la página PHP que muestra los datos procedentes del motor debe refrescarse automáticamente para poder mostrar, en cada momento los datos correspondientes a ese instante. Uno de los requisitos fundamentales para ello es que el refresco sea lo más rápido posible, por lo que la opción de recargar constantemente la página PHP no es una opción válida, además de que se verían "parpadeos" mientras se carga la totalidad de la página. Es por eso que se ha optado por refrescar únicamente una parte de la página mediante tecnología AJAX. Esta técnica, basada en Javascript y XML asíncronos, permite precisamente eso, refrescar una parte de la página cargando nuevos datos procedentes del servidor de forma asíncrona, todo ello de forma rápida y transparente para el usuario. El funcionamiento de AJAX es muy sencillo, realiza una petición asíncrona a otra página PHP, y esta le devuelve como respuesta los datos necesarios para recargar la página desde la que ha salido la petición. En este proyecto concreto, la petición se realiza a una página PHP que consulta en Memcache para ver si el valor de los datos ha cambiado con respecto a la última petición, en cuyo caso los devuelve para su mostrado.

Se ha decidido usar este sistema de refresco de pantalla con el fin de respetar al máximo la arquitectura de Modelo-Vista-Controlador. Como puede observarse, la parte AJAX desempeña el papel de Controlador, Memcache y la página PHP que la consulta representan el Modelo, mientras que la página que visualiza los datos realiza las funciones de la Vista.

Gracias a este sistema, se ha conseguido simular de forma simple y muy eficiente el comportamiento de un cuadro de mandos de un vehículo, en el que en todo momento se conocen los datos actuales de la conducción. A continuación, se muestra un diagrama de secuencia con la interacción AJAX entre las dos páginas PHP, acompañado de un par de capturas de pantalla de la interfaz desarrollada.

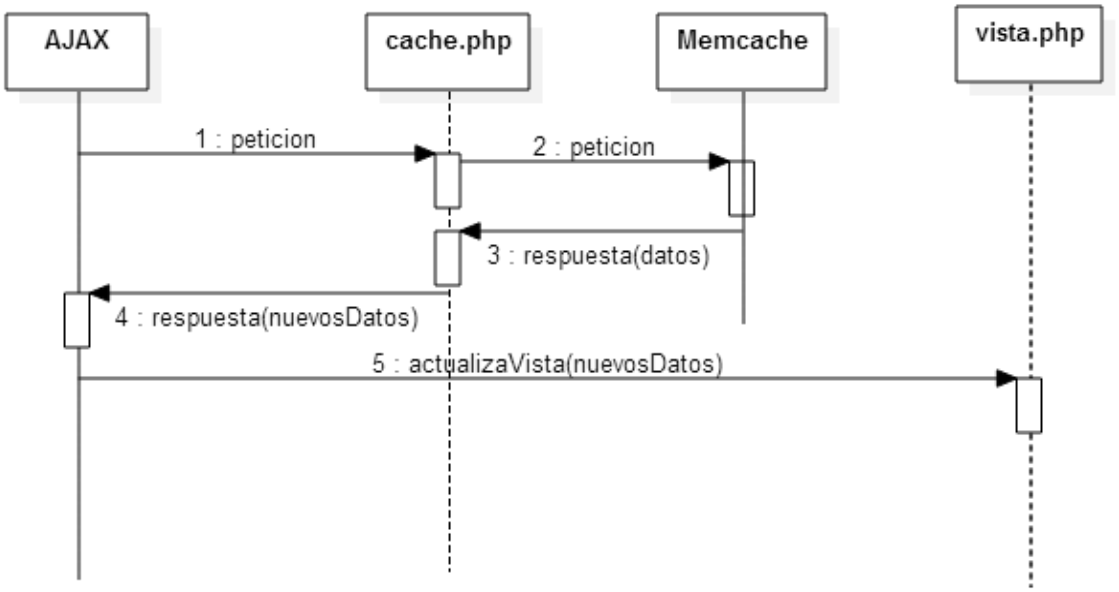


Figura 31 – Diagrama de secuencia del mecanismo Memcache

Figura 32 – Interfaz visualización datos del motor 1 - Velocidad

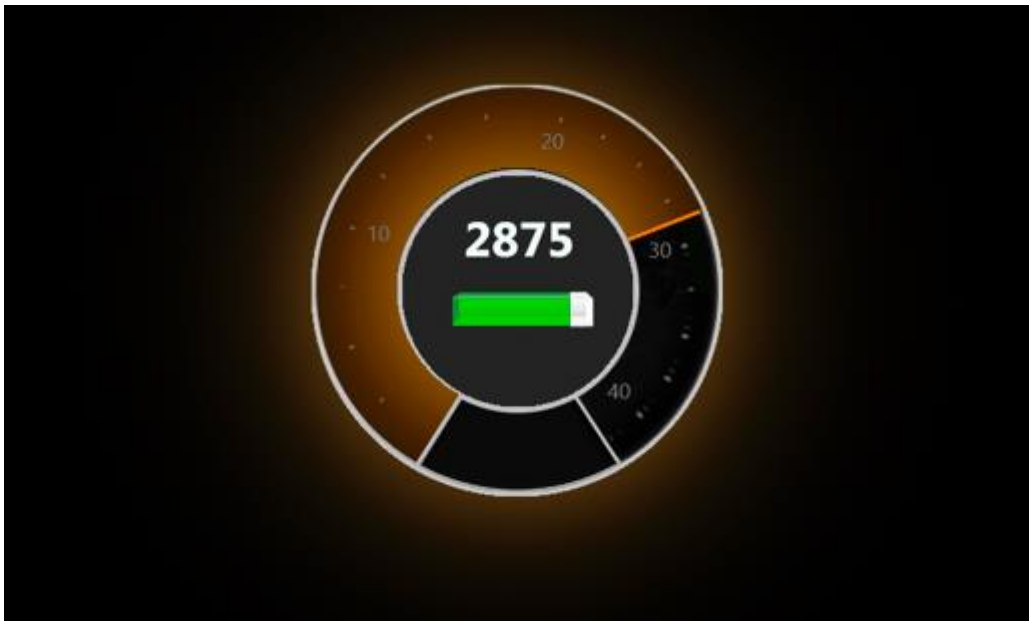


Figura 33 – Interfaz visualización datos del motor 2 - RPM

9.1.3 Procesado de comandos por voz

La tercera decisión de diseño que se ha tomado para el desarrollo de este TFG permite asociar acciones a algunos comandos de voz, lo cual permite un control casi total del sistema, así como una forma muy sencilla de agregar nuevos comandos en el futuro. Para ello, ha sido necesario que la biblioteca que se ha utilizado para el reconocimiento de voz, PocketSphinx (escrita en C y de la que ya se ha hablado en el apartado anterior), comunique las palabras que reconoce a un script en Python desarrollado *ad hoc* por el alumno, para realizar las acciones oportunas. De nuevo, aquí se trata de comunicar dos procesos escritos en diferentes lenguajes (al igual que sucedía entre la biblioteca de adquisición de datos del motor y la parte de visualización de los mismos). La diferencia principal con respecto a la comunicación necesaria en la parte OBD es que aquí no es necesario que la comunicación sea tan veloz.

Se ha optado por tanto por utilizar un fichero de texto intermedio en el que la biblioteca de reconocimiento va escribiendo a medida que va reconociendo palabras, y a su vez el script de Python lee en bucle en busca de cambios en dicho fichero. Si el script de Python detecta que PocketSphinx ha reconocido una nueva palabra, la procesa y ejecuta la acción asociada a ella.

La razón principal por la que se ha tomado la decisión de emplear un fichero, en lugar de un sistema de comunicación más complejo, como el que fue necesario en la primera decisión de diseño explicada, ha sido principalmente la sencillez de este sistema. Al no necesitar mucha velocidad en la comunicación, el propuesto es probablemente el método más sencillo y con menor tasa de errores, así como muy depurable si apareciese algún error.

A continuación, se muestra un diagrama de muy alto nivel, que recoge el sistema de comunicación implementado.

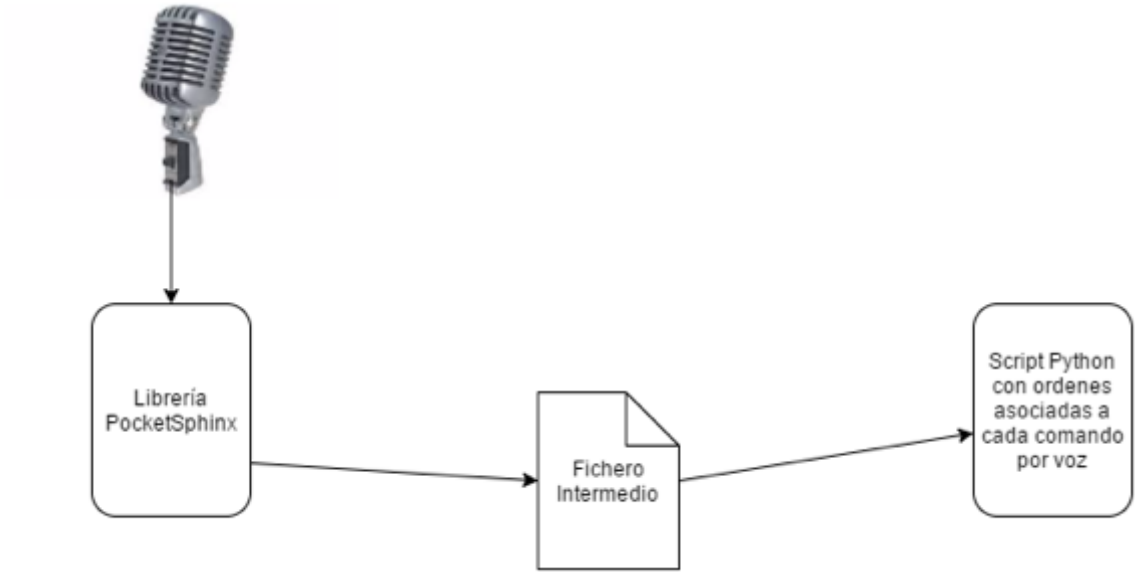


Figura 34 – Diagrama de alto nivel del procesado de los comandos de voz

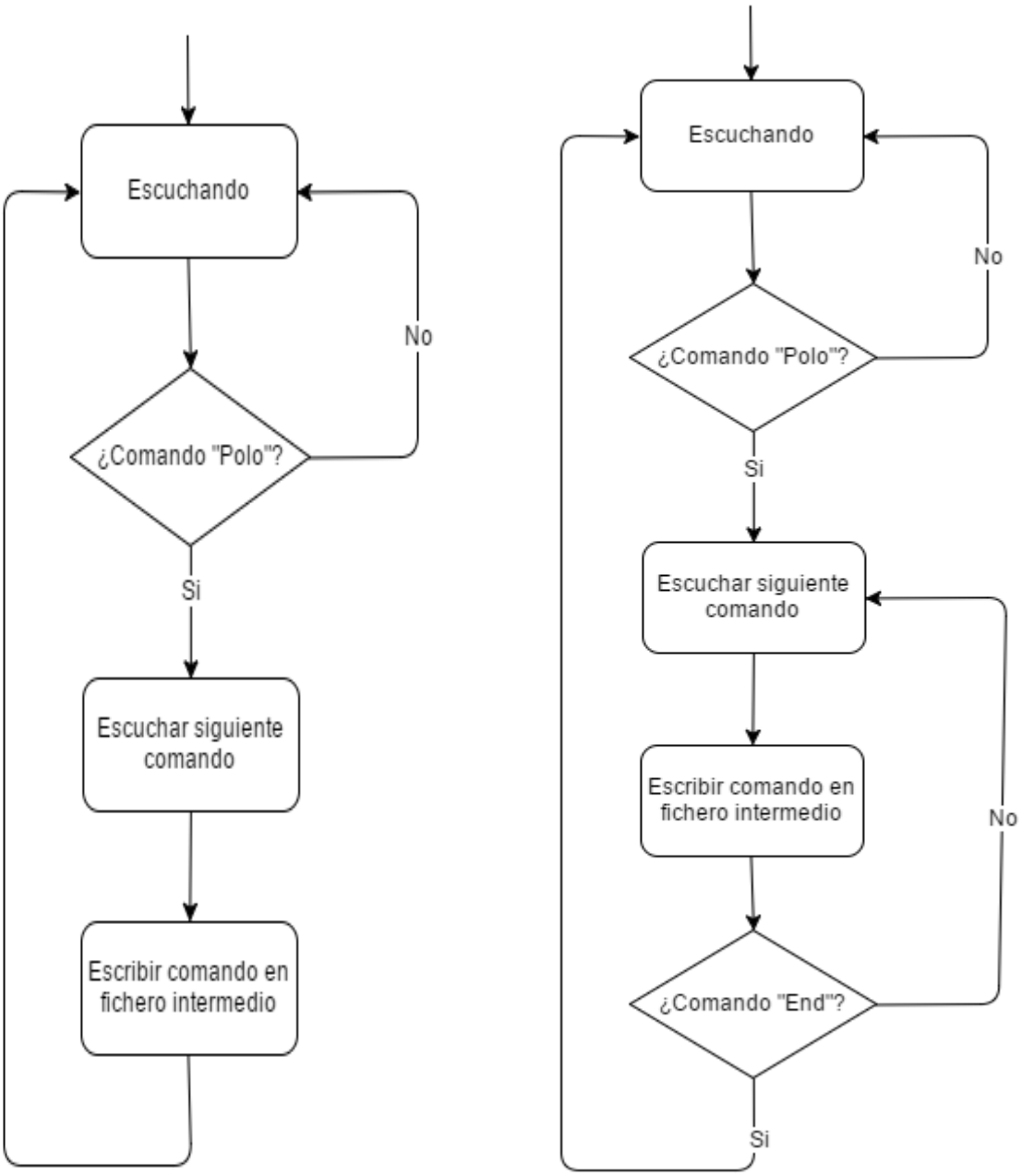
9.1.4 Control por voz estilo “Ok Google”

Finalmente, la cuarta y última decisión de diseño que se va a detallar, modela el estilo de comunicación por voz con el sistema que se ha implementado. Debido a las características del proyecto, uno de los requisitos fundamentales de esta parte era que el sistema no podía actuar en cuanto detectara una palabra conocida, ya que cualquier conversación entre los pasajeros del vehículo podría interferir con el correcto funcionamiento del mismo.

Se ha decidido por tanto desarrollar un sistema que permanezca escuchando en busca de una palabra clave para, a continuación, escuchar los siguientes comandos por voz y ejecutar sus acciones asociadas. Este comportamiento se mantiene hasta que se detecta otra palabra clave de finalización, ante la cual, el sistema de reconocimiento de voz vuelve a estado “latente”. De esta forma, el usuario puede estar seguro de que el control por voz únicamente se activa cuando él lo desee, evitando así posibles malos funcionamientos.

A pesar de que el citado, es el método mediante el que funciona el sistema final, se han valorado otras opciones, llegando incluso a implementar un par de ellas en etapas tempranas del desarrollo. Una de estas opciones únicamente escuchaba el siguiente comando por voz tras reconocer la palabra clave, lo cual hacía al sistema altamente ineficiente.

Seguidamente se adjunta un Diagrama de Transiciones que representa el funcionamiento tanto de esta versión preliminar, como de la versión final, permitiendo apreciar a la perfección la diferencia entre ambos:



1ª Versión Versión Final

Figura 35 – Diagrama de alto nivel del estilo de comunicación "Ok Google"

9.2 Fotos

En este apartado se incluyen algunas fotos tomadas durante el desarrollo del proyecto. Cada una se acompaña de una breve descripción de lo que representan:

Figura 36 – Detalle de la Raspberry Pi 3 y latrasera de la pantalla táctil

Esta foto fue tomada al poco tiempo de adquirir la pantalla táctil para la Raspberry Pi. Como puede observarse, esta se conecta a través del puerto DSI integrado en la placa.

.

Figura 37 – Espacio disponible en el cajón portaobjetos del vehículo del alumno.

Este reducido espacio ha sido el lugar donde se ha colocado la pantalla táctil y el circuito que la levanta y baja.

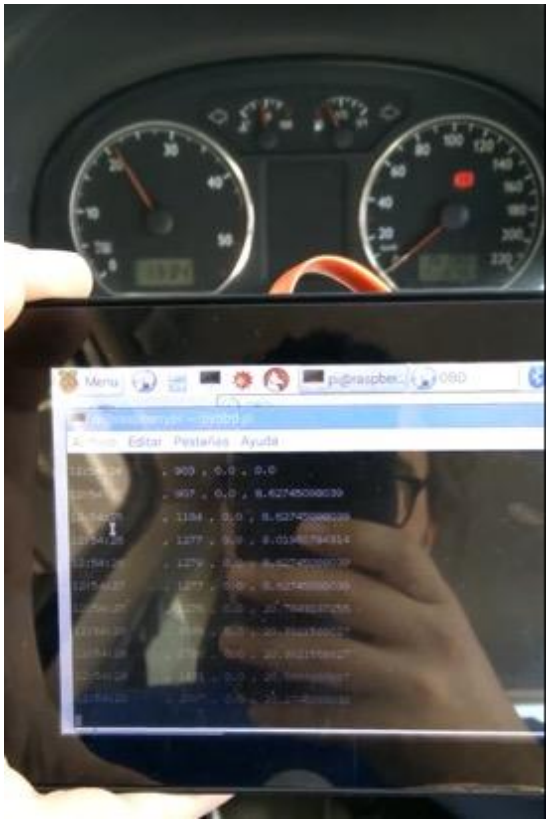


Figura 38 – Lectura de parámetros OBD en modo texto

Una de las primeras pruebas satisfactorias de lectura de parámetors en modo texto. La interfaz gráfica aún no estaba desarrollada en el momento de tomar esta fotografía.

Figura 39 – Prueba extrema realizada en la interfaz OBD

Se simuló que el vehículo viajaba a 160 Km/h. Por supuesto esta prueba se realizó fuera del vehículo, haciendo creer a la interfaz que esos datos llegaban del motor cuando realmente no era así.

Figura 40 – Primeras pruebas de control por voz (1).

En esta imagen aparece la webcam empleada por el alumno para captar los comandos por voz.

Figura 41 – Primeras preubas de control por voz (2).

Aquí se puede ver al alumno visualizando mediante SSH la interpretación que realiza el sistema ante los comandos por voz. En la terminal de la izquierda aparece lo que detecta la librería PocketSphinx y en la derecha el filtrado y asingación de la acción. A continuación se muestra un extracto de cada una de las terminales:

Terminal izquierda:	*Terminal derecha:*
No te he escuchado	*Right*
Escuchando	*Right*
Right	*Enter*
Escuchando	
Right	
Escuchando	
Enter	

Figura 42 – Detalle del reductor de velocidad.

Fue necesaria la integración de este componente para evitar que la pantalla táctil sufriera daños debido a la gran velocidad de rotación a la que giraba el motor que la levantaba y ocultaba. Esta foto está tomada al poco de adquirir este componente.

Figura 43 – Salpicadero del vehículo en pleno proceso de desmontado.

Hubo que retirar todo el frontal del salpicadero del vehículo del alumno para poder colocar el circuito que levanta y oculta la pantalla táctil.

Figura 44 – Colocación de la pantalla táctil

Esta foto se tomó poco después de terminar el circuito que levanta y oculta la pantalla. El salpicadero aún está desatornillado y con todos los cables a la vista.