Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

MONITORIZACIÓN DE PARÁMETROS DE UN VEHÍCULO USANDO OBD.

Alumno:

Tutor:

Depto.: Ingeniería de Telecomunicación

Junio, 2015

Indice General

Contenido

Memoria del Proyecto

1. RESUMEN

El siguiente trabajo tiene como objetivo crear una plataforma de trabajo mediante la cual poder monitorizar y gestionar las distintas variables del motor de un coche, haciendo uso de un sistema basado en microcontrolador, el servicio GPS, la red de datos GPRS y el protocolo de diagnóstico OBD-II. Las distintas variables medidas del vehículo serán enviadas a un servidor web remoto con acceso a un servidor de base de datos para que dichas variables sean almacenadas. Además se propone el uso de una aplicación web cliente orientada al usuario final para que pueda acceder de una manera intuitiva a la visualización y gestión de la información muestreada.

2. INTRODUCCIÓN

En esta sección haremos una breve introducción al Trabajo Fin de grado, introduciremos las motivaciones que nos llevaron a su realización y comentaremos brevemente la estructura que se ha seguido para la elaboración del mismo.

2.1. Motivación y Contexto

La necesidad de ahorro de energía es una de las primeras prioridades a nivel mundial, sobre todo en los sistemas que utilizan como fuente de energía combustibles fósiles, esto nos lleva al campo de la automoción, y plantearnos la siguiente pregunta, ¿es posible hoy en día con la tecnología de la que disponemos ahorrar combustible con una inversión mínima en nuestro vehículo?

Para dar respuesta a este interrogante se nos ocurrió en primer lugar, monitorizar las variables del vehículo, en concentro las que nos permitieran realizar un cálculo aproximado del consumo del automóvil, permitiendo así una segunda segunda fase de investigación futura en la que se puedan usar dicho datos para crear patrones de conducta y modelos estadísticos del perfil del conductor y rendimiento del vehículo.

Por tanto el ámbito de este trabajo es crear una plataforma que permita tener un mayor información del estado de un vehículo, gracias a la monitorización de las distintas variables para poder conocer en mayor medida la manera en la que el conductor maneja su coche, de

esta manera el usuario podrá monitorizar su vehículo y obtener la capacidad para detectar practicas no demasiado eficientes y corregir su conducción.

2.2. Estructura del trabajo fin de grado

En este apartado se presenta la estructura del trabajo fin de grado. Existen distintos apartados:

Memoria

- **Resumen**. Breve resumen del trabajo realizado.
- **Introducción**. En ella se realiza una aproximación a las motivaciones que originaron este trabajo y se enumeran los distintos apartados de la memoria.
- **Objetivos.** Se describen los objetivos que se persiguen en el presente trabajo fin de grado.
- **Materiales.** Se enumeraran los materiales necesarios para la realización de este trabajo además de introducir y explicar las tecnologías necesarias para la comprensión del mismo.
- **Métodos.** Se explicaran las distintas partes en las que se ha desarrollado el trabajo, además. Están diferenciadas en fases, y se corresponden con los distintos objetivos alcanzados hasta llegar a la finalización total del trabajo.
- **Resultados y discusión.** Se analizaran y discutirán los resultados obtenidos, analizando por un lado los datos obtenidos, el distinto funcionamiento del hardware y las ventajas del uso del protocolo OBD-II

- **Conclusiones.** Se señalan las conclusiones del trabajo en función de los resultados obtenidos, se analizara su utilidad y se valorara su funcionamiento, además se discutirá sobre líneas futuras y tareas a desarrollar a partir del trabajo realizado.
- **Anexos**. Se resume y ubican los documentos adicionales para la realización de este trabajo.
- **Bibliografía.** La enumeración de fuentes relativas al trabajo.

Pliego de Condiciones

- **Especificaciones técnicas.** Detallaremos de manera particular cada componente requerido en la realización de este trabajo.

Estudio Económico

- **Costes Materiales.** Se presentaran las distintas partidas económicas que componen la totalidad de los gastos de la realización del siguiente trabajo.
- **Resumen del presupuesto.** Aquí se facilitara el total del coste del trabajo fin de grado.

3. OBJETIVOS

En esta sección se detalla los objetivos perseguidos por en este trabajo.

Se pretende construir una aplicación que permita monitorizar información del motor de un vehículo haciendo uso de OBD (On Board Diagnostics, sistema de diagnóstico a bordo en vehículos).

Para ello se utilizará un sistema basado en Arduino para recoger los datos del vehículo y enviarlos a una base de datos remota. Esta base contendrá la información del motor y será presentada al usuario utilizando una aplicación web que muestra estadísticas y gráficas.

Para el cumplimiento de tales objetivos se realizaran las siguientes tareas

- En una primera fase se llevará a cabo un estudio del estándar OBD utilizado en Europa (que mejor pueda adaptarse a nuestros vehículos) y codificación de un programa para arduino que permita monitorizar algunos parámetros (variables).

- En segundo lugar, se programará las funciones y aplicaciones de servidor necesarias para el envío de los datos monitorizados a una base de datos remota vía GPRS (implementada utilizando un sistema gestor MySQL).

-En una tercera fase se realizarán las aplicaciones de servidor que presentan al usuario las gráficas y estadísticas de los distintos parámetros.

- En un apartado de líneas de futuro se presentarán las posibles variables que podrían ser útiles para predecir comportamientos del motor y mejorar el rendimiento, por ejemplo mejorar el consumo de combustible.

- Finalmente, se elaborará una memoria del proyecto.

4. MATERIALES

En esta sección se comentaran los materiales utilizados en el trabajo, además de dar una introducción a las tecnologías empleadas para una mayor comprensión del mismo.

En primer lugar hablaremos de que antecedentes hemos encontrado en el mercado actualmente.

Por ultimo hablaremos del estado del arte de las distintas tecnologías empleadas además de sus características más importantes. Dividiremos los materiales en tres categorías, atendiendo a sus naturalezas, por un lado tenemos la placa Arduino. Por otro los elementos encargados de dotar al arduino de interfaces extra y por el ultimo las tecnologías empleadas en las aplicación web.

4.1. Antecedentes

En el mercado encontramos algunas aplicaciones con características parecidas, en concreto la que motivo este proyecto consista en un dispositivo arduino que haciendo uso del protocolo OBD obtenía la información del motor y la mostraba por una pantalla que tenía el arduino conectado. Consideramos que este método de monitorización de la información no es demasiado práctico, ya que cuando el usuario esta condición no debería de distraerse con pantallas ni estímulos que no provengan de la carretera, además, los datos no eran procesados con lo cual no se sacaba ningún partido a la información. A raíz de todo esto decidimos mejorar el dispositivo de la manera que en este trabajo se detalla.

4.2. Arduino

Arduino es una solución hardware que bajo la filosofía del open source, código abierto, propone un sistema de desarrollo basado en un microcontrolador conectado a una placa con distintos líneas de entrada y salida, tanto analógicas como digitales.

Arduino se ha convertido en una de las plataformas favoritas para la mayoría de desarrolladores, tanto a nivel amateur como a nivel profesional, gracias a su alto nivel de compatibilidad con otros sistemas, desde sencillas sondas eléctricas como sensores de humedad, resistencias foto sensibles, etc… hasta sistemas electrónicos más complejos, como router, motores, sistemas demóticos, computadores, etc.

Otra ventaja de este sistema es que cuenta con un entorno de programación sencillo y gratuito y la programación del mismo es bajo es lenguaje de programación C.

El gran éxito de Arduino consiste en crear una herramienta versátil y de gran capacidad para controlar, gestionar y manipular infinidad de sistemas electrónicos.

El proyecto Arduino surge en el año 2005 como un proyecto para estudiantes del instituto IVREA en Italia, el primer modelo era mucho más pesado y costoso, gracias a los esfuerzos de diversos investigadores consiguieron hacerlo disponible para la comunidad de código abierto.

Posteriormente y tras el cierre del instituto IVREA Google colaboro también con el desarrollo de Arduino, creando una placa con compatibilidad hacia Android.

Pero no fue hasta el año 2011 en la feria de Maker cuando se presentara realmente al gran público el primer producto comercial de Arduino.

Actualmente existen en el mercado infinidad de modelos compatibles, gracias a que es Open Source muchas tiendas y empresas se han lanzado al desarrollo de sus propias placas Arduino, compatibles con el entorno de desarrollo oficial y además, compatibles también con la mayoría de periféricos, o shield, propios para el Arduino original.

EL proyecto original Arduino comercializa actualmente las siguientes placas Arduino

Arduino UNO: actualmente en la revisión 3, Ilustración 3.1, es el modelo más básico y completo de Arduino, basado en el microcontrolador ATmega328 cuenta con 14 entradas/salidas digitales, 6 entradas analógicas, un cuarzo a 16Mhz, conexión USB y adaptador Ac-Dc. En cuanto a la memoria, este modelo cuenta con 32KB con 0.5KB reservada para el sector de arranque, además cuenta con 2KB de SRAM y 1KB de memoria EEPROM.

ILUSTRACIÓN 4.1 ARDUINO UNO

Arduino Leonardo: este modelo está basado en el microcontrolador ATmega32u4 y cuenta con 20 entradas/salidas digitales de las cuales 12 pueden ser usadas como entradas analógicas, cuenta con un cuarzo de 16Mhz, conexión Micro USB y Jack de alimentación. Este

modelo cuenta con una memoria de 32KB, de la cual, 4KB está reservado para el sector de arranque, 2.5 SRAM y 1KB de memoria EEPROM.

Arduino Yún: esta placa está basada en el ATmega32u4, ilustración 3.2, y el Atheros AR9331. Este procesador es compatible con distribuciones Linux basadas en OpenWrt (OpenWrt-Yun) y tiene soporte para Ethernet y Wifi. Cuenta con un puerto USB, ranura de expansión para tarjetas de memoria micro-SD y además tiene 20 entradas/salidas digitales, de las cuales 12 pueden ser usadas como entradas analógicas.

ILUSTRACIÓN 4.2 ARDUINO YUN

En el apartado de memoria, este Arduino cuenta con 32KB de los cuales 4KB son usados para el sector de arranque, 2.5 KB de SRAM y 1KB de memoria EEPROM. Además cuenta con 64MB de RAM DDR2 y 16MB de memoria flash, esta memoria viene de fábrica precargada con la distribución OpenWrt-Yun.

Arduino Mega: Por último hablaremos de la placa Arduino Mega, ilustración 3.3, esta placa está basada en un microcontrolador ATmega2560, este modelo es el de mayor tamaño y cuenta con un total de 54 terminales de entrada y salida digitales, 16 entradas analógicas, 4 UARTSs, tiene un cristal de 16Mhz y cuenta con un conexión USB y un conector Jack para la alimentación.

En cuanto a las capacidades de este Arduino contamos con 256KB de memoria flash para almacenado del programa, de las cuales 8KB son destinadas al sector de arranque, 8KB de SRAM y 4KB de EEPROM.

ILUSTRACIÓN 4.3 ARDUINO MEGA

Para este trabajo hemos elegido este modelo de Arduino ya que cuenta con un mayor número de puertos UART los cuales nos serán necesarios para la comunicación con las distintas interfaces que le serán conectadas.

En cuanto a la programación de los Arduino hay que decir que es similar para todas las placas, todos los programas cuentan con la misma estructura general:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

ILUSTRACIÓN 4.4 ESTRUCTURA DE PROGRAMA

Como vemos en la ilustracion3.4, en primer lugar tenemos el *setup*, es la primera parte que se ejecuta cuando se inicializa el programa, y ya no se vuelve a ejecutar más.

Es la parte destinada a las inicializaciones de los distintos elementos necesarios durante la ejecución del programa.

En segundo lugar tenemos el *loop,* es el bucle principal del programa, y se ejecutara de manera cíclica indefinidamente, en esta parte es donde va el grueso del programa.

Hay que tener en cuenta esto a la hora de programar, pues es un programa cíclico y no lineal secuencial como suele ser el software de PC.

Para realizar la tarea de programación la fundación Arduino provee de una interfaz de entorno de desarrollo gratuita llamada Arduino y accesible desde su página web.

Desde ella además de programar podremos validar el código y enviar el programa compilado a la placa por el USB.

Arduino ha servido para abrir la veda a otros fabricantes a animarse a lanzar otros sistemas de desarrollo, actualmente hay muchas alternativas, aquí comentaremos brevemente 2 de estas alternativas

- **Raspberry Pi.** A diferencia de Arduino, el sistema Raspberry pi nos ofrece una computadora completa y totalmente funcional, con la capacidad para mover un sistema operativo completo.

  En el campo de las especificaciones técnicas Raspberry cuenta con un microprocesador a 700Mhz, 512 MB de RAM, conectividad Ethernet con RJ45, memoria flash en formato SD con una capacidad de 2 a 16 GB y dos puertos USBs.

  Por esto, es la placa que suele utilizarse para la elaboración de trabajos softwares o que necesiten mayor capacidad de computación, mientras que para los proyectos con una mayor componente hardware es recomendable el uso de Arduino, ya que es más sencillo, y su programación está orientada a la programación de más bajo nivel.

- **Waspmote.** Esta iniciativa surge de la empresa Libelium, Esta empresa española propone una placa basa de en microcontrolador, en concreto en el modelo ATmega1281a una frecuencia de trabajo de 14Mhz, cuenta con 8KB de memoria SRAM, 4KB de memoria EEPROM y 128KB de memoria Flash. Waspmote utiliza un IDE de programación muy similar al de Arduino, de hecho está basado en el, con lo cual permite que el código pueda ser compatible entre ambos.

  La placa Waspmote tiene más prestaciones de serie que la placa básica de Arduino lo cual hace que el precio de esta sea mayor que la de Arduino, lo que hace que para algunos proyectos en los que no se vaya a utilizar todo esas capacidades adicionales, merezca más la pena utilizar Arduino.

4.3. Interfaces para comunicaciones externas

Son periféricos que dotaran a la placa Arduino de la capacidad de conectarse con las interfaces necesarias para poder medir los datos del vehículo (Cable OBD-II), obtener información relativa a posicionamiento (Antena GPS) y enviar los datos al servidor remoto (GPRS shield).

4.3.1. On Board Diagnostics

On Board Diagnostics (OBD) es un sistema de diagnóstico y alertas implantado en la mayoría de los vehículos a nivel mundial, se principal objetivo es el control de la emisiones de gases, este sistemas además hace uso de todos las sondas implementadas en el coche para el control de todos los sistemas implicados en la emisión de gases, como la inyección, la entrada de aire, etc. Si algún componente presentara un comportamiento anómalo el propio sistema OBD sería capaz de registrarlo y guardar un informe de estado además de notificarlo al piloto mediante el uso de alguna alerta. El conector OBD no tiene una posición fija dentro del vehículo, y su ubicación depende del fabricante, del modelo y de la revisión del auto. Además no todos los vehículos cuentan con el mismo número de sensores, dependiendo de la gama del modelo tendrá más o menos. Si es estándar el uso de dichos sensores, y el

protocolo de comunicación de los mismos, y la manera de mostrar dicha información (Ver anexo sobre PIDS OBD).

OBD surge en California, EEUU, allá por el año 1988 para reducir la contaminación del aire. California aprobó ese año la "California Air Resource Board" que determino que todos los vehículos a gasolina contaran con OBD-I.

En el año 1996 se intensificaron dichas medidas y esto llevo a la creación de un nuevo estándar, OBD-II. Por esa fecha se introdujo en Europa el OBD, pero ajustándose a las características técnicas de la segunda revisión, es decir, a OBD-II.

En el futuro se habla de implementar un nuevo nivel OBD-III en el que sean los propios automóviles los encargados de notificar a las autoridades cuando el coche presente un mal funcionamiento o contamine más de lo permitido.

El protocolo OBD cuenta, a día de hoy, con tres variantes:

- **OBD-I.** Primera versión del protocolo, era un sistema no muy efectivo, pues solo monitorizaba a algunos componentes y además no eran calibrados para un nivel específico de emisiones.
- **OBD-II.** Esta segunda revisión, más precisa, cuenta con un mecanismo doble para la alerta en caso de fallo. En primer lugar el protocolo enciende un piloto que el vehículo debe de tener en el cuadro un testigo de aviso de avería, MIL. Después, una vez alertado el piloto, este deberá ir a su taller donde el operario podrá hacer uso de una herramienta de diagnóstico compatible con OBD-II donde vera un código de error, que proporcionara información detalla del error, que componente y que anomalía presenta.

Esta versión cuenta con un mayor número de sondas y partes que monitorizar

- **EOBD.** Es una tercera versión del protocolo OBD y es la utilizada en Europa. Es básicamente igual que la OBD-II salvo que cuenta con algunas sondas menos, pero el conector aunque no es igual, si es compatible con el OBD-II.

El funcionamiento del protocolo OBD-II está recogido en la normativa ISO 15031-3:2004. La normativa estipula sobre su ubicación el vehículo, el cual debe de estar siempre en el compartimento de los pasajeros cerca del asiento del conductor.

También incluye lo referente al conector, que debe de ser de 16 pines

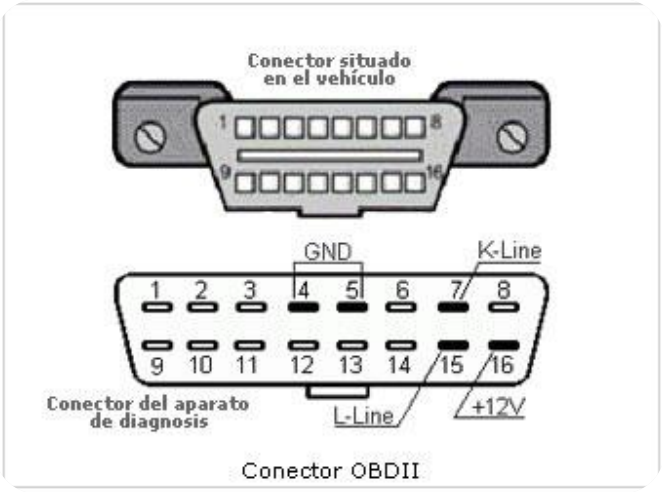

ILUSTRACIÓN 4.5 CONECTOR OBD

2 - J1850 (Bus +)

4 - Masa del Vehículo

5 - Masa de la Señal

6 - CAN High (J-2284)

7 - ISO 9141-2 "Línea K"

(Como vemos no se utilizan todos)

10 - J1850 (Bus -)

14 - CAN Low (J-2284)

15 - ISO 9141-2 "Línea L"

16 - Batería +

Atendiendo a las especificaciones en cuanto a la comunicación en OBD, tenemos tres normas diferentes:

- **ISO 9141**: El tipo de datos que envía es asíncrono, como RS-232 con la salvedad de que este trabaja de manera semiduplex, envía datos en una sola dirección. Puede enviar hasta 12 bytes de información de una sola vez incluyendo CRC.

- **SAE J1850 VPW**: El ancho de pulso es siempre variable, con un voltaje máximo de 7V. Al igual que en el protocolo ISO 9141, la información que puede enviar este en una sola vez se encuentra limitada a 12 bytes.

- **SAE J1850 PWM**: A diferencia del protocolo SAE J1850 VPW encontramos una modulación de ancho de pulso que no podemos variar a nuestro gusto. Se puede aplicar un voltaje máximo de 5V y al igual que el resto de protocolos de OBD2, el mensaje no puede pesar más de 12 bytes.

4.3.2. Global Positioning System

El sistema de posición global, GPS, es un servicio que permite la ubicación de un dispositivo en cualquier parte del mundo, con tan solo unos metros de error. Este sistema fue desarrollado y puesto en funcionamiento por el departamento de defensa de los Estados Unidos sobre los años 70. Actualmente utiliza unos 24 satélites, aunque solo son necesarios 18, los otros se mantienen para sustituir a los principales en caso de avería o imprevisto, orbitando alrededor de la tierra en orbitas no geoestacionarias, ya que de esta manera si se pueden cubrir las zonas polares.

Es necesario establecer una comunicación con al menos 4 satélites para poder determinar la posición de nuestro dispositivo, la cual se determina por triangulación y teniendo en cuenta los tiempos de retardo en las comunicaciones, por eso es fundamental que los relojes de todos los dispositivos que participan en la comunicación, satélites como dispositivo usuario, estén perfectamente sincronizados.

GPS no es el primer sistema de navegación basado en satélites. En el año 1965 los departamentos de defensa, transporte y la agencia espacial norteamericanas, aunaron esfuerzos para desarrollar un sistema mediante el cual pudieran determinar la posición de un elemento en cualquier parte del planeta, que no le afectaran las condiciones atmosféricas, rápido y preciso, condiciones necesarias para poder implementarlo en sistemas de aviación, por tanto llegaron a la conclusión de que la mejor manera de llevar a cabo este cometido era implementando un sistema basado en satélites. El sistema resultante fue el llamado TRANSIT, el cual consistía en

una constelación de 6 satélites en órbita polar baja, a 1074km, lo cual garantizaba la cobertura global, pero el sistema era intermitente, de hecho solo se podía establecer comunicación con el satélite cada 90min y para el cálculo de la posición, eran al menos necesario seguir al satélite durante unos 15min.

El sistema TRANSIST no ofrecía una solución definitiva al problema de posicionamiento global. Por otro lado la URSS también había desarrollado en paralelo un sistema similar llamado TSICADA. Dada la coyuntura internacional fomentada por la guerra fría, hizo que EEUU invirtiera una importante cuantía de dinero en la inversión de un nuevo sistema, NAVSTAR, el cual proponía el uso de 24 satélites en órbita media que de esta manera si fuera global y accesible en todo momento. El primer satélite de este sistema se lanzó en el año 1978 y se pretendía que la constelación estuviera completada 8 años después, pero por diversos problemas el sistema se fue posponiendo y no fue hasta el año 1983 que comenzara realmente la fase inicial del proyecto, ya conocido como GPS.

En el 1984 el gobierno de EEUU decido liberar y dejar que el público civil accediera sus sistema de posicionamiento, fue también este año cuando empezaron a aparecer receptores de GPS destinados al gran público.

El funcionamiento del sistema GPS se basa en la trilateración, para ello cada satélite publica información útil para el receptor, esta información se llama efemérides. Las efemérides de cada satélite incluyen información relativa al estado del satélite, si se le debe considerar o no para la determinación de la posición, su posición en el espacio, su hora atómica, información doppler, etc. Esta información está sujeta a los parámetros orbitales de cada satélite, viéndose

influenciada por los cambios gravitatorios, vientos solares, etc. Típicamente los datos se suelen actualizar cada 4horas.

Para determinar la posición de un elemento, en primer lugar el receptor establece una comunicación con un satélite, este, lo único que puede determinar es que el receptor se encuentra en un punto situado en la superficie de la esfera de radio la distancia emisor-receptor y centro el propio satélite emisor.

Al establecer la comunicación con un segundo satélite, determinan que existen dos esferas con radio emisor1-receptor y emisor2-receptor centradas en los distintos satélites, emisor1 y emisor2. De esta manera el punto a posicionar queda contenido sobre la circunferencia resultante al interseccionar las dos esferas.

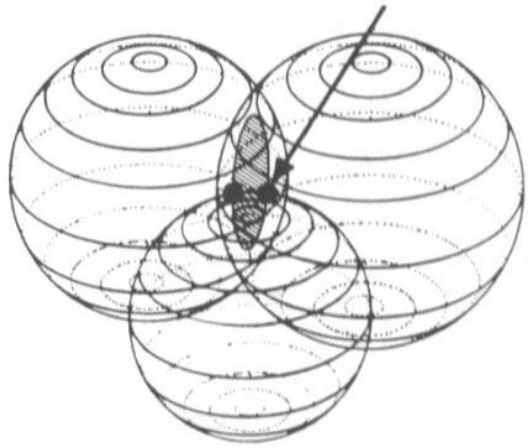

ILUSTRACIÓN 4.6. INTERSECCIÓN DE TRES ESFERAS

Al introducir la información de un tercer satélite más, que interseccione una tercera esfera emisor3-recptor centrada en el emisro3 tenemos que de la circunferencia anterior, esta esfera solo interseeciona en dos puntos.

Por ultimo solo se necesitara un cuarto satélite que permita que los tres satélites emisores y el receptor sincronicen sus relojes, hasta este momento, la intersección de las esferas anteriores determinaba realmente un volumen en lugar de un punto, gracias al sincronismo se pueden implementar mecanismos de corrección que son capaz de convertir ese volumen en un punto univoco.

Actualmente además del sistema de posicionamiento GPS existen otras alternativas. Principalmente dos, la alternativa europea, Galileo, y la alternativa china, Beidou.

El sistema Galileo contara con una red de 30 satélites que orbitaran a 23222 kilómetros y utilizaran la técnica de multilateración. Las principales ventajas de este sistema respecto el GPS son que Galileo presenta una precisión mucho mayor, con una precisión de menos de 4 metros en horizontal y 8 en vertical. También incluye mejoras en el rango y la disponibilidad de la señal, llegando así a lugares donde el GPS no llega. Y lo más importante es que es un sistema totalmente independiente del sistema americano, no siendo dependientes de la disponibilidad del GPS por parte del gobierno americano.

Aunque estas características son muy prometedoras no hay que olvidar que es plenamente especulativo, dado que a día de hoy el sistema aun no es funcional, el proyecto lleva presentando muchos retrasos desde su lanzamiento, y apenas un par de satélites están a día de hoy en órbita.

Por otro lado está el sistema chino, Beidou. El funcionamiento de este sistema es muy diferente al GPS, este sistema utiliza satélites en orbitas geoestacionarias, con lo cual necesita de menos satélites, pero limita la zona de cobertura, en este caso, solo es accesible desde china y sus países colindantes, además este sistema solo necesita dos satélites y una estación en tierra para determinar la posición del receptor. Beidou ofrecerá dos servicios diferenciados, por un lado un servicio abierto con un margen de error de 10metros y un segundo que solo será autorizado para determinados clientes con un servicio más precioso y con mayor seguridad.

La fase uno del proyecto Beidou está en funcionamiento y se prevé que haya una segunda generación que aumente el número de satélites y sea funcional para el año 2020.

4.3.3. General Packet Radio Service

General Packet Radio Service (GPRS) es una red de comunicaciones telefónica basada en el envío de información utilizando radioenlaces. Puede usar dos tecnologías diferentes de tráfico de información, por un lado conmutación de paquetes y por el otro la conmutación de circuitos, característica heredada de su antecesora la red GSM, Global S. La principal ventaja de que implemente estos dos modos de funcionamiento es que permite por un lado que el tráfico de datos pueda ser enviada por la red de conmutación de paquetes(IP) y así ser tarificada por cantidad de bytes enviados y por otro lado que el tráfico de voz emplee la conmutación de circuitos(x25). De esta manera aseguramos que los enlaces abiertos estén siempre ocupados.

Las características generales de esta red son:

- Mejora y simplifica el acceso inalámbrico a las redes de datos
- Accede a nodos IP y .x25
- Mejora el tiempo de acceso de GMS
- Mejora la tasa de datos
- Se puede facturar por r volumen de trafico
- Posibilidad de conexión permanente
- Transmisión asimétrica
- Coste casi nulo de establecimiento de conexión
- Uso más eficiente del espectro
- Se pueden compartir canales.

Esta red también es denominada como red 2.5G, ya que su cometido era suplir las carencias y limitaciones de la red GSM, 2G, pero no supuso un salto cualitativo suficiente, más bien sirvió como nexo de unión con la siguiente generación, 3G, redes UMTS.

En el año 1992, la European Telecommunicatios Standars Institute, ETSI, estandarizo y para uso comercial los servicios GSM, los cuales incluían, telefonía, mensajería, fax y datos

En el año 1996 se lanzó una segunda revisión del estándar GSM, estableciendo un marco para la mejora de los servicios ya existentes.

Por último y en el año 2000 la estandarización GSM alcanzo la fase 2+, la cual incluyo mejoras en la codificación de voz y en los servicios de transmisión de datos. Esta evolución de la red GSM se denominó GPRS añadiendo además la posibilidad de conmutación de paquetes.

La arquitectura de esta red necesita de los siguientes elementos

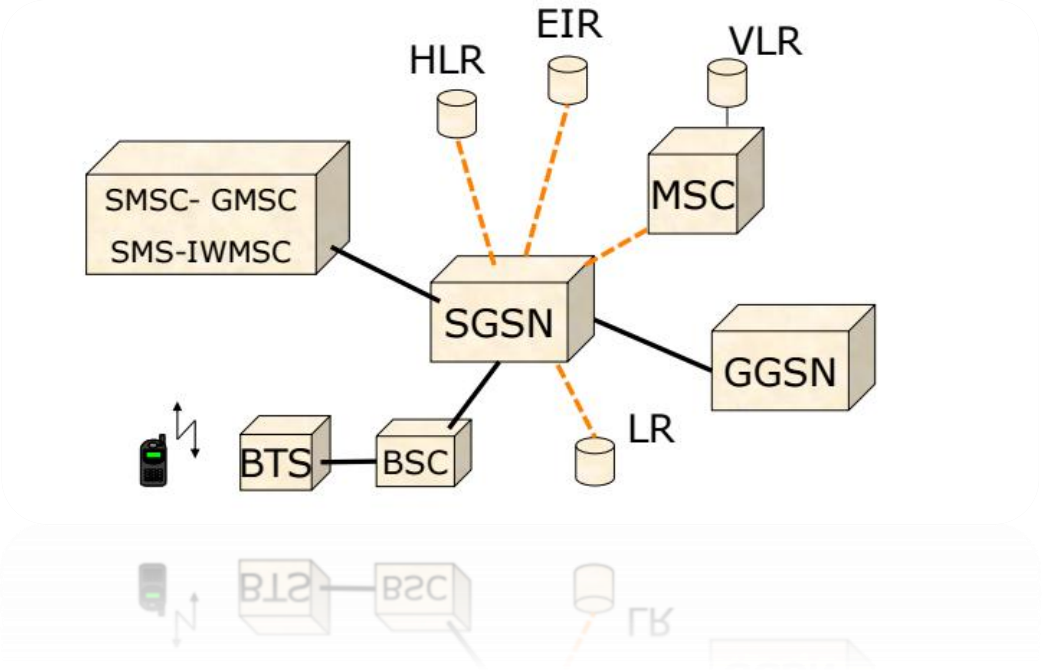

ILUSTRACIÓN 4.5 TOPOLOGÍA RED GPRS

Como vemos en la ilustración 4.7, el dispositivo móvil se conecta con la estación base más cercana (BTS), la cual está conectada a una estación base de control (BSC), que agrupa a un conjunto de BTS. Las BSC son elementos que eliminan complejidad en la red. Estas estaciones están conectadas a una Serving GPRS Support Node, SGSN, estos nodos son los responsables de distribuir los datos dentro su área de servicio, además tienen acceso a los registros de la red, con lo cual además de encaminar tráfico son los encargados de la autenticación de los usuarios en la red.

Los registros de la red son Home Location Register (HLR), Equipament Identity Register (EIR) y Location Register (LR) estas bases de datos contienen la información relativa al usuario, localización y estado de los servicios.

Los SGSN están conectados con las centralitas de conmutación de circuitos, MSC, y con los nodos Pasarela de GPRS, GGSN, encargados de convertir y adecuar la información a las redes destino, IP o X.25.

Además existen también las pasarelas frontera, BG, que son los elementos de interconexión entre las redes de los distintos operadores.

Los servicios ofertados por esta red son:

- PTP: Punto a punto vía IP (modo paquete) o .X25 (orientado a conexión).
- PTM: Punto a multipunto, servicio de difusión en un área, grupos IP o multicast.
- Otros: SMS, WAP, MMS.

Para poder acceder a esta red es necesaria una tarjeta SIM, que contenga el Mobile Station ISDN (MSISDN) y el Internacional Mobile Subscriber Identity (IMSI), y un teléfono con un Internacional Mobile Equipament Identitty (IMEI).

Además es necesario configurar en nuestro terminal un Access Point Name (APN), el cual contiene toda la información necesaria para poder conectarnos con la red de datos de nuestro operador.

Esta red sigue actualmente en funcionamiento y coexiste tanto con el sistema anterior, GSM, como las redes que han ido llegando después, UMTS, HDSPA y LTE.

4.4. Aplicaciones web

En cuanto a las aplicaciones web están realizadas usando la tecnología JAVA, concretamente la extensión para páginas web, Java Server Pages junto con las herramientas de Google, como Google Maps y Google Charts.

4.4.1 Java

Java es un lenguaje de programación de propósito general y orientado a objetos, su propósito era que un mismo código escrito una vez fuera capaz de ser ejecutado en cualquier dispositivo o plataforma. Es decir, el código se escribe y compila una vez, y no es necesario compilarlo cada vez para cada plataforma. Java es actualmente uno de los lenguajes de programación más populares en uso, sobre todo en aplicaciones cliente y aplicaciones servidor web.

El lenguaje Java deriva del lenguaje C y C++ aunque elimina en casi su totalidad las utilidades de bajo nivel. Las aplicaciones Java están orientadas a ser ejecutadas bajo su máquina virtual (JVM).

Java fue creado en el año 1991 en las instalaciones de Sun Microsystem, fue desarrollado como una herramienta dentro del proyecto The Green Project, por el un equipo formado por trece personas dirigidas por James Gosling.

En el año 1994 java sufrió un cambio en su orientación apuntando hacia el Web, con la idea de que gracias a los nuevos navegadores web internet se convirtiera en un medio masivo y para

el gran público, por ello crearon su propio prototipo de navegador web compatible con java, HotJava. Desde ese año java 1.0a estaba disponible, pero no fue hasta el año siguiente que el navegador como java vieran la luz pública, la acogida fue tal que hasta otros navegadores como Netscape, el más importante en esa época, incluyo java en su navegador.

Actualmente java va por su versión SE8 y está presente en la mayoría de plataformas y dispositivos actuales, desde el mundo móvil, en plataformas como Android y BalckBerry, como en los sistemas operativos Windows y todos los basados en Unix, como MacOS y todas las distribuciones de Linux.

Nos centraremos en esta última debido a que es la tecnología empleada en la elaboración de las páginas web de este trabajo.

Java Server Pages (JSP) es una tecnología orientada a la creación de páginas web que combinen código html con instrucciones y métodos java. Las aplicaciones JSP se ejecutan en el servidor, y generan un código resultante, que puede ser html o cualquier otro, como xml, JavaScript. Es una tecnología que simplifica enormemente la creación de páginas web dinámicas, aunque es necesario que la aplicación web sea lanzada por un servidor especial compatible con dichas tecnologías, como por ejemplo el servidor Tomcat de la fundacion Apache.

La principal ventaja de este sistema es que permite al desarrollador crear clases java compatibles con multitud de plataformas y sistemas. Esto permite crear interfaces web de manera muy sencilla compatibles con las clases java programadas previamente.

Las aplicaciones web JSP cumplen con los principales objetivos principales de java:

- Están orientadas a objetos.
- Permite que el código se ejecute en diversos sistemas operativos
- Puede ser ejecutado en sistemas remotos de manera segura.

La programación de un JSP es muy sencilla, basta con crear una estructura de página web HTML y donde sea necesario se introduce una marca y a continuación se introduce todo el código java que se dese. Al llegar a esa marca el servidor ejecutara esas líneas de código, por tanto al usuario final no llegaran las instrucciones java, sí no el resultado de esas líneas.

Vemos un ejemplo:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body>
        <h1>Ejemplo de fichero JSP</h1>
        </br>
        <%  //Marca de comienzo de codigo Java
            out.println(6+3);
        %>
    </body>
</html>
```

ILUSTRACIÓN 4.6. FICHERO JSP DE MUESTRA

Este fichero JSP se ejecutaría en el servidor y generaría el siguiente código html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body>
        <h1>Ejemplo de fichero JSP</h1>
        </br>
        9
    </body>
</html>
```

ILUSTRACIÓN 4.7. FICHERO HTML DE SALIDA

4.4.2. API Google

Primero de todo explicaremos que es una interfaz de programación de aplicaciones (API). Es el conjunto de métodos o procedimientos que ofrece una biblioteca externa para ser utilizada en un software determinado independiente. Uno de los cometidos fundamentales de las APIs es dotar al software de unas funciones de uso general, por ejemplo, dibujar un mapa tomando como centro las coordenadas que queramos. Esto se consigue añadiendo un nivel de abstracción extra, es decir, consideremos una API como una caja negra, que no sabemos, o no tenemos por qué conocer cómo funciona, solo necesitamos conocer las entradas que solicita, y para esas entradas que salidas nos va a dar. Esto hace realmente sencillo su implementación en cualquier código.

En el caso concreto de Google, las API son un conjunto de herramientas que Google pone a disposición de los desarrolladores de manera gratuita para que las utilicen en sus páginas web. De esta manera pueden implementarse en nuestro sitio web las aplicaciones de Google como mapas, su buscador, su traductor, etc.

El funcionamiento de sus API es sencillo, Solo basta conocer sus estructura general a la hora de implementarlas y de qué manera hay que formatear y enviarle los datos. También podemos configurar las salidas que nos dan, adecuando su tamaño, colores, formato etc…

En este trabajo haremos uso de dos de estas.

- **Google MAPS** será la herramienta utilizada para la visualización de la información relativa a posicionamiento y la encargada de mostrar la información en un mapa.
- **Google Charts** Esta herramienta será la encargada de mostrar los datos en gráficas.

5. MÉTODOS

En esta sección hablaremos de la metodología llevada a cabo para el cumplimiento de todos los objetivos del presente Trabajo fin de Grado. Podemos dividirla en 6 fases claramente diferenciadas, atendiendo a los objetivos parciales alcanzados por cada una de ellas.

5.1. Fase 1. Creación de la infraestructura

En esta primera fase, o fase previa, se llevaron a cabo las tareas necesarias para la creación de una plataforma que sirva como base para el uso del dispositivo hardware Arduino.

En primer se definió una base de datos donde guardar la información generada por el dispositivo junto con la información facilitada por el usuario y otra generada por la propia aplicación, la base de datos sigue la siguiente estructura

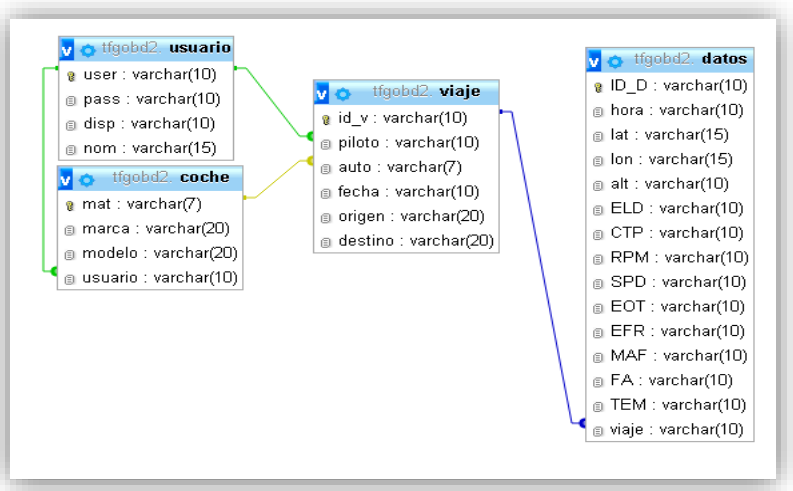

ILUSTRACIÓN 5.1 ESTRUCTURA BASE DE DATOS

La base de datos consta de cuatro tablas, usuario, coche, viaje y datos.

- **Tabla usuario**

En esta tabla se guardara toda la información relativa a los usuarios finales que utilizaran el dispositivo, consta de los siguientes campos:

User: un identificador de usuario facilitado por el usuario, será necesario para poder acceder a la información.

Pass: Contraseña de usuario, segunda parte de las credenciales del usuario, también será necesaria para autentificar al usuario.

Disp: Un nombre de dispositivo que identifique de manera única e univoca a cada dispositivo Arduino.

Nom: El nombre de usuario, un campo informativo que deberá ser facilitado por el usuario.

- **Tabla Coche**

En esta tabla se almacenará todo lo relacionado con el vehículo, formada por:

Modelo: Modelo del vehículo que será monitorizado.

Marca: Marca del fabricante del automóvil.

Mat: Matrícula del vehículo.

Usuario: nombre de usuario poseedor del vehículo o al que se le ha asignado dicho auto.

Nótese que por la definición dela base de datos, un usuario puede tener varios coches, pero un coche solo puede tener un único usuario asignado.

- **Tabla Viaje**

Esta tabla está pensada para el alojamiento de la información vinculada a cada viaje que se realice con el dispositivo. Consta de:

Id_v: identificador de viaje, es un número que identifica de manera única a cada viaje, se genera de manera automática por la aplicación web encargada del almacenaje de la información

Piloto: usuario responsable de realizar el viaje

Auto: Vehículo que realiza el viaje.

Fecha: fecha de la realización del viaje.

Origen: Punto de partida del viaje realizado.

Destino: Punto final del viaje realizado.

- **Tabla Datos**

Esta es la tabla más importante de la base de datos porque es en la que se almacenará la información obtenida por el dispositivo. Está formada por los siguientes campos:

Id_d: Identificador único de dato, se genera de manera automática por la aplicación web encargada de guardar los datos.

Hora: Hora a la que se ha tomado la muestra.

Lat: valor de la coordenada latitud del vehículo.

Lon: valor de la coordenada longitud del vehículo.

Alt: metros sobre el nivel del mar a los que se encuentra el vehículo.

ELD, CTP, RPM, SPD, EOT, EFR, MAF, FA, TEM, son los campos destinados a guardar la información de las variables obtenidas por el protocolo OBD-II, que son detallados posteriormente en esta sección.

Nótese que como en el caso de los usuarios y los vehículos, un viaje puede tener muchos datos, pero un dato solo puede estar contenido en un viaje.

Además de la base de datos en esta fase fue necesario crear una aplicación web que fuera capaz de estar a la espera del envío de datos por parte del dispositivo, para ello se programó la aplicación llamada guarda.jsp (esta aplicación está en el cd adjunto).

Esta aplicación consta de un fichero que hace uso de la tecnología jsp. Esta aplicación procesa el mensaje de petición GET (protocolo HTTP) y extrae la información de cada parámetro (variable).

El formato y el orden de los datos esperados es el siguiente:

```
guarda.jsp?DISP="dis"&USER="user"&HORA="hora"&LAT="lati"
  &LON="lon"&ALT="alt"&ELD="eld"&CTP="ctp"&RPM="rpm"
  &SPD="spd"&EOT="eot"&EFR="efr"&MAF="maf"&FA="fa"
  &TEM="tem"
```

ILUSTRACIÓN 5.2 ESTRUCTURA DEL COMANDO GET

Siendo:

Disp.: Un identificador de dispositivo, es un valor Alfanumérico y único que identifica perfectamente a cada Arduino, se asigna de manera manual en la programación del dispositivo basado en microcontrolador.

User.: El nombre de usuario del conductor propietario del dispositivo, debe ser también único se le debe asociar un dispositivo en la base datos, para que los datos puedan ser correctamente almacenados.

Hora.: hora a la que se toma la muestra de datos desde el dispositivo, la asigna automáticamente el dispositivo en función de la información proporcionada por el GPS.

Lat.: Valor de la Latitud en el instante determinado, facilitada por el servicio GPS.

Lon.: Valor de la Longitud en el instante determinado, facilitada por el servicio GPS.

Alt.: Valor de la altitud en el instante determinado, facilitada por el servicio GPS.

Eld.: Porcentaje medido por el protocolo OBD-II referente a la carga del motor en ese momento.

Ctp.: Temperatura del refrigerante del motor, expresada en grados y obtenida por el protocolo OBD-II.

Rpm.: Numero de revoluciones por minuto del motor, obtenida por el protocolo OBD-II.

Spd.: Velocidad del vehículo en kilómetros hora, obtenida por el protocolo OBD-II.

Eot.: Temperatura del aceite del motor en grados, obtenida por el protocolo OBD-II.

Efr.: Ratio de combustible en función del tiempo, expresado en Litros partido hora, obtenido por el protocolo OBD-II.

Maf.: Flujo de masa de aire, expresado en gramos partido segundo, obtenido por el protocolo OBD-II.

Fa.: Ratio de equivalencia entre el combustible y el aire, obtenida por el protocolo OBD-II.

Tem.: Temperatura del motor.

El funcionamiento de la aplicación es el siguiente. En primer lugar verifica que el usuario que envía los datos es un usuario valido, es decir que este registrado en la base de datos. Una vez comprobado esto, se cerciora de que el identificador de dispositivo asociado al usuario sea verdaderamente suyo. Una vez comprobado esto se considera que el usuario es válido y se procede a la extracción de la información de las variables mandadas.

Antes de mandarlas a la base de datos se asigna un identificador de viaje, Id_v, recordemos que era imprescindible para poder almacenar el dato.

Para asignar el Id_v la aplicación comprueba la hora del último registro valido almacenado para ese usuario, si la diferencia de tiempo es menor de 5 minutos, considera que se encuentra en el mismo viaje, por tanto el id_v será el mismo que tuviera en el último dato almacenado. Sí por el contrario la diferencia de tiempos fuera mayor, generara un identificador de viaje nuevo.

Después de tener un id_v valido es necesario la generación de un identificador de dato, que recordemos, también debe ser único, para la generación de este identificador la aplicación solo necesita mirar el último registro añadido a la tabal datos, e incrementarlo en uno.

Una vez generados los identificadores, la información está lista para ser procesada y almacenada en la base de datos.

5.2. Fase 2. Envío de información

En la segunda fase se pretendía conseguir el envío de datos desde el Arduino, haciendo uso del GPRS shield, un placa basa en SIM9000. En primer lugar hay que insertarle al GPRS shield una tarjeta SIM operativa y con crédito para que pueda conectarse.

El GPRS shield dota al Arduino de la capacidad de conectarse a las redes GPRS, pudiendo así realizar conexiones vía internet

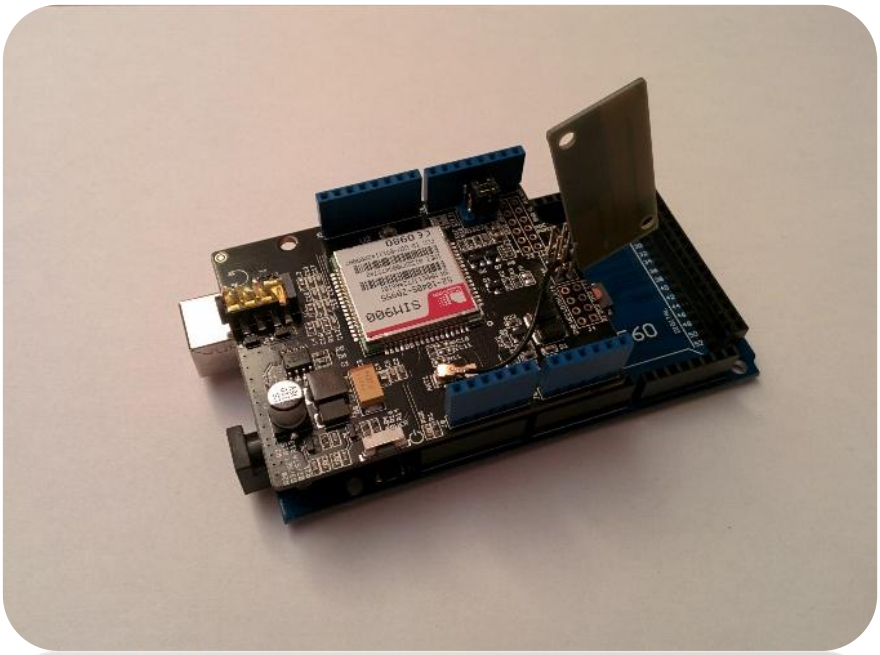

ILUSTRACIÓN 5.3 ARDUINO CON GPRS SHIELD

La conexión del GPRS shield con el Arduino es sencilla, ya que el shield es un shield compatible con Arduino. El shield GPRS tiene dos vías de comunicación con el Arduino, por un lado podemos utilizar la comunicación Hardware serial, donde la comunicación se haría por el

Serial del Arduino, terminal 0 y terminal 1, o usar el Software serial, dejando así libre el Serial del Arduino, de esta manera usaríamos los terminales 7 y 8.

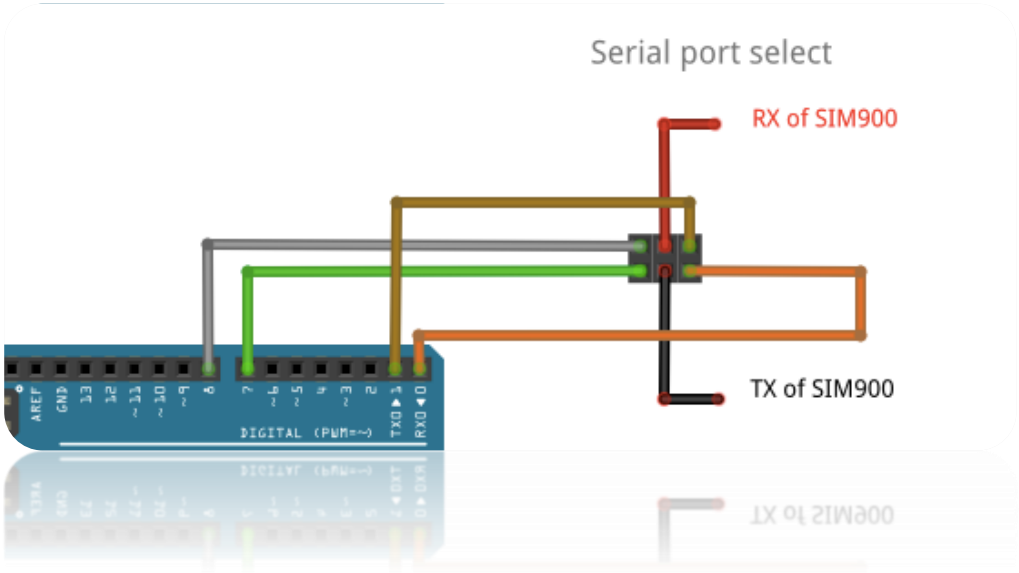

ILUSTRACIÓN 5.4 TIPOS DE CONEXIÓN UART

Como vemos en la ilustración 5.3 para cambiar el modo de funcionamiento solo necesitamos colocar los jumpers en la posición deseada.

Para inicializarse GPRS shield, podemos o bien utilizar el interruptor que tiene situado en el lateral de la placa, o mediante u pulso alto enviado atreves de la patilla 9 del Arduino.

Una vez conectado correctamente y ajustado el modo de funcionamiento tenemos nuestro Arduino dotado de todas las funcionalidades propias de los dispositivos SIM9000 (Ver anexo SIM9000).

Para hacer uso de las funcionalidades SIM9000, tenemos que utilizar los comandos AT. Gracias a estos comandos podemos desde realizar llamadas hasta conectarnos a la red, como

el uso de los comandos GET y POST de HTTP. En nuestro caso concreto necesitaremos conectarnos a la red y montar una petición GET para él envió de un dato de prueba un servidor.

La secuencia de comandos AT que habría que mandar al Shield seria:

```
AT+CGDCONT?                      |
AT+CGACT?                        |
AT+CMEE=1                        | Comandos para la configuracion del APN
AT+CGATT=1                       |       y
AT+CGACT=1, 1                    | el registro en la red gprs
AT+SAPBR=3,1,"CONTYPE","GPRS"    |
AT+SAPBR=3,1,"APN","internet"    |
AT+SAPBR=1,1                     |
AT+SAPBR=2,1                     |
AT+HTTPINIT                         |
AT+HTTPPARA="URL","url"             |
AT+HTTPACTION=0                     | Parte encarga de la comunicacion http
AT+HTTPREAD                         |
AT+HTTPTERM                         |
```

ILUSTRACIÓN 5.5 COMANDOS AT

La configuración del APN depende del operador con el que tengamos hayamos contratado el acceso a internet móvil. En nuestro caso el ISP es Simyo, y la configuración del APN sería la siguiente:

```
APN: gprs-service.com
Nombre de usuario: déjalo en blanco
Contraseña: déjalo en blanco
Autentificación: déjalo en blanco
Comprimir datos: Desactivado
Comprimir cabeceras: Desactivado
```

ILUSTRACIÓN 5.6 CONFIGURACIÓN APN SIMYO

Teniendo en cuenta esto, el código para poder realizar la petición desde nuestro Arduino sería:

```
GPRS.println("AT+CGDCONT?");
delay(900);
GPRS.println("AT+CGACT?");
delay(900);
GPRS.println("AT+CMEE=1");
delay(900);
GPRS.println("AT+CGATT=1");
delay(900);
GPRS.println("AT+CGACT=1, 1");
delay(900);
GPRS.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
delay(900);
GPRS.println("AT+SAPBR=3,1,\"APN\",\"gprs-service.com\"");
delay(900);
GPRS.println("AT+SAPBR=1,1");
delay(900);
GPRS.println("AT+SAPBR=2,1");
delay(900);
GPRS.println("AT+HTTPINIT");
delay(900);
GPRS.println(buff);
delay(900);
GPRS.println("AT+HTTPACTION=0");
delay(900);
GPRS.println("AT+HTTPREAD");
delay(900);
GPRS.println("AT+HTTPTERM");
delay(150);
```

ILUSTRACIÓN 5.7 RUTINA ENCARGADA DE LA CONEXIÓN

Siendo buff, ver ilustración 5.2, la cadena de texto que contiene la url del servidor y el recurso GET con los parámetros que vamos a mandar.

Vemos que lo único que hay que hacer es utilizar la interfaz GPRS, que es una comunicación serial virtual que hay que definir, para mandar los comandos AT al GPRS shield.

Para finalizar esta parte además de configurar el dispositivo Arduino con el GPRS Shield, es necesario disponer de la aplicación web descrita en la Fase 1 y mandar un dato, al cual, podamos acceder de alguna manera para poder ver que la comunicación se realizaba correctamente.

Una vez completada la fase 2 continuamos el trabajo con la fase siguiente.

5.3. Fase 3. Servicio GPS

En esta fase, se pretendía utilizar y conectar una antena GPS a nuestra placa Arduino y mostrar las coordenadas de latitud, longitud y el valor de la altitud junto con la hora de la mediccion.

En esta fase además del Arduino necesitaremos una placa que nos permita el acceso al servicio GPS.

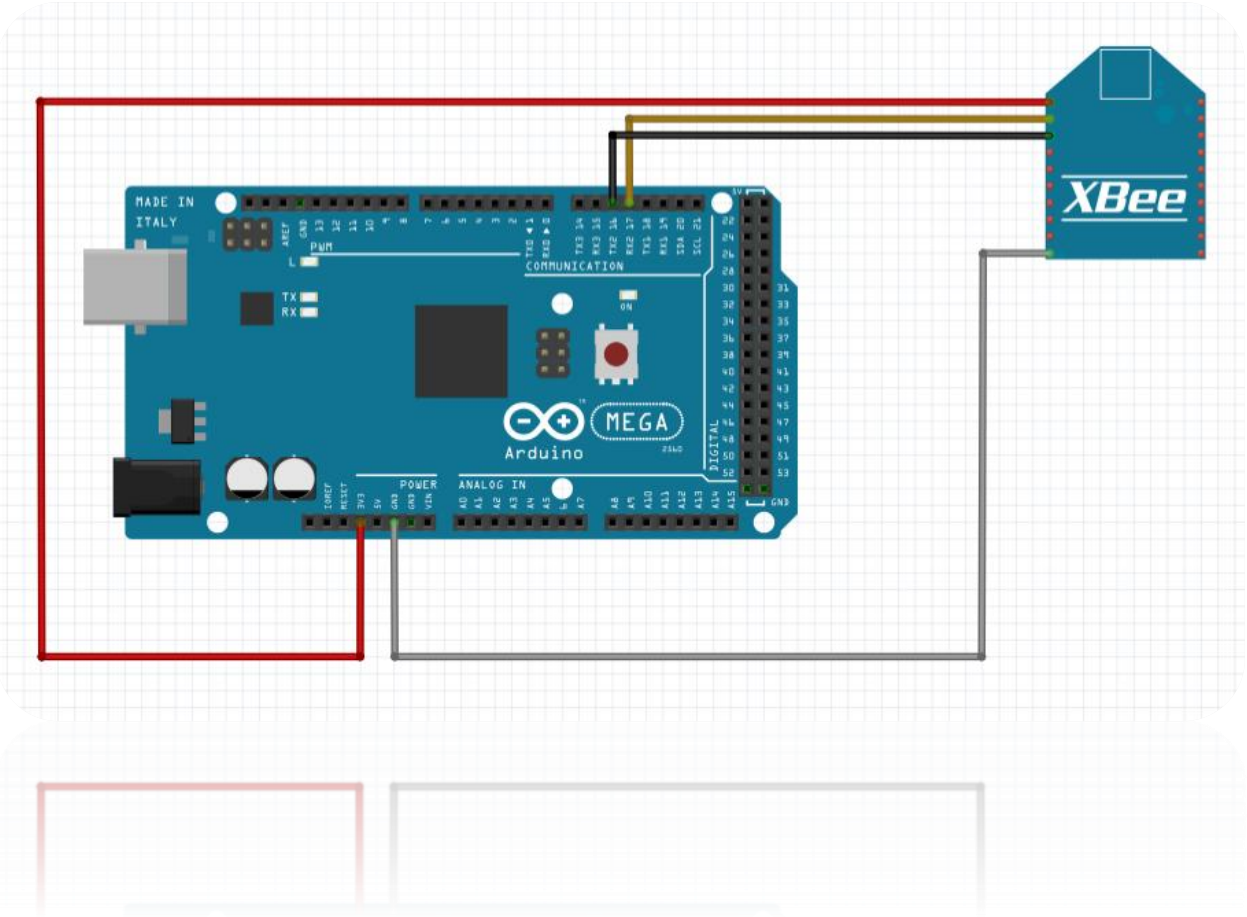

ILUSTRACIÓN 5.8. ARDUINO Y GPS BEE KIT

Como vemos en la ilustración 5.8, hemos usado una placa del tipo xBee. De esta placa usaremos 4 de sus terminales, concretamente los de Alimentación, Vin, y Vout junto con los terminales de comunicación Rx y Tx,. La conexión con nuestro Arduino es sencilla, utilizaremos los terminales de la placa dedicados a la alimentación para alimentar la antena y usaremos el Hardware Serial 2 para el envío y recepción de datos.

En este caso la programación se facilita bastante ya que podemos hacer uso de una librería externa llamada TinyGPSPlus que nos simplifica bastante el uso del servicio GPS

El código para la obtención de los datos GPS seria el siguiente:

```
while (Serial2.available() > 0){
    if (gps.encode(Serial2.read())){
        if(gps.location.isValid()){
    Serial.println(gps.location.lat());
    Serial.println(gps.location.lng());
        }

        if(gps.altitude.isValid()){
    Serial.println(gps.altitude.meters());
        }

        if(gps.time.isValid()){
    Serial.println(gps.time.hour());
          Serial.println(gps.time.minute());
        }
    }
}
```

ILUSTRACIÓN 5.9 RUTINA ENCARGADA DE LA COMUNICACIÓN GPS

Simplemente tenemos que hacer uso de la librería GPS y utilizar sus funciones para obtener la información. La información obtenida es mostrada por el Serial estándar del Arduino que es el utilizado para mostrar información vía USB en el PC.

Vemos que gracias a esta librería no tenemos que conocer las primitivas utilizadas en el servicio GPS, ni los comandos necesarios para realizar la conexión.

5.4. Fase 4. Comunicación con el automóvil

Ahora buscamos la comunicación del Arduino con el vehículo. Para ello haremos uso del cable OBD-II, gracias a este protocolo podremos comunicar el Arduino Mega con el coche. Además conectaremos el Arduino a un PC para poder ver por el los datos medidos con el Arduino en tiempo real.

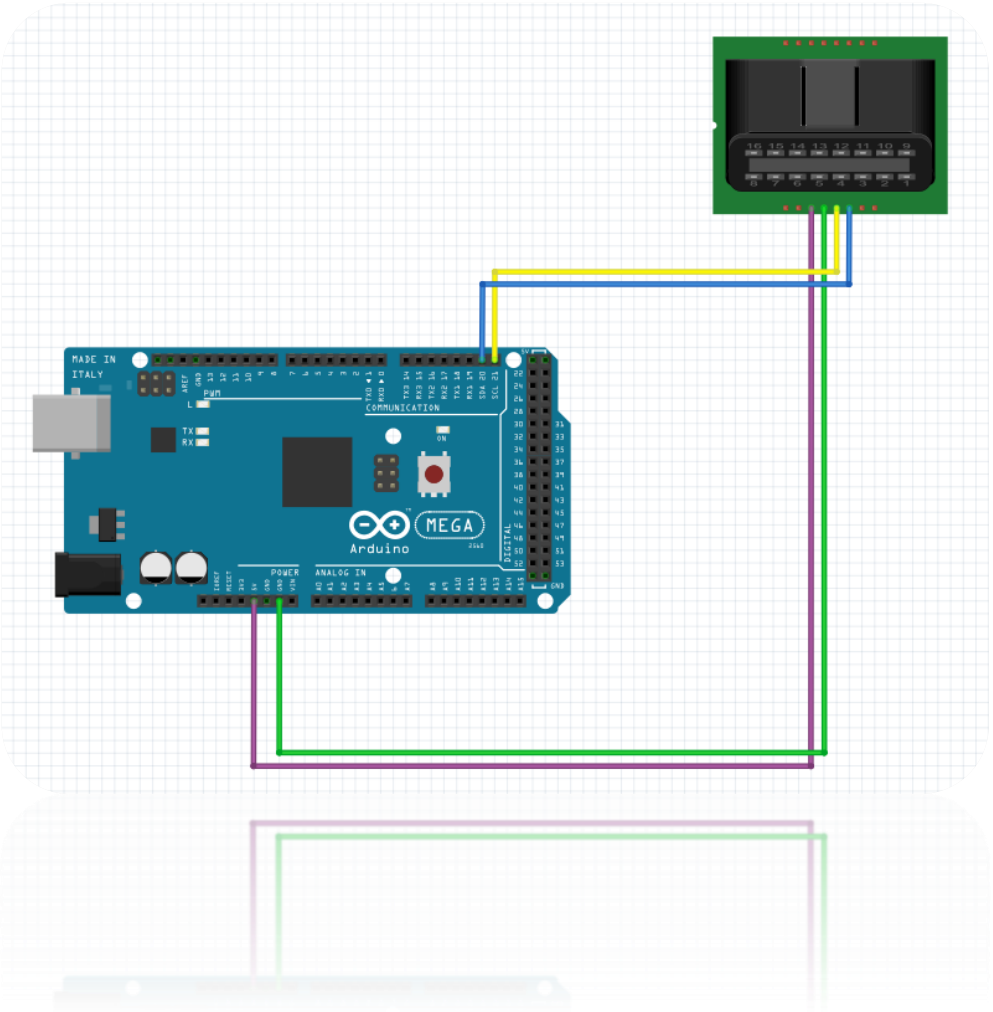

ILUSTRACIÓN 5.10 ARDUINO MEGA Y CABLE OBD

En primer lugar es necesario conectar el cable a la placa. En nuestro caso el cable utiliza el protocolo de comunicación I2C (Ver anexo I2C). El cable cuenta con 4 terminales, de los cuales 2 son para datos y 2 son para comunicaciones. Para la alimentación utilizaremos los pines de la fuente de alimentación de la placa y para datos usaremos los terminales 43 y 44 del Arduino, los llamados SCL y SDA que son los que se implementan para el protocolo I2C.

Para esta fase fue necesario además del Arduino Mega, el cable y realizar las pruebas en vehículos. Se utilizaran tres coches.

- Hyundai Terracan 4WD (vehículo 1)
- Citroën C4 (vehículo 2)
- Ford Focus (vehículo 3)

La primera dificultad de esta fase viene dada por la diferente ubicación del conector OBD según el fabricante y el modelo, En el caso del vehículo 1 el conector se situaba debajo del volante en el hueco creado entre este y los pedales. En el caso del auto 2 el conector se situaba en oculto bajo el cenicero, con lo cual hasta que este no era retirado no era accesible el puerto. Por ultimo en el vehículo 3 el puerto se encontraba tras una placa de plástico cerca del volante y pegado casi a la puerta del conductor.

Una vez localizado y con el motor apagado enchufamos el cable, en este experimento no es necesario proveer al Arduino de una batería adicional, ya que por el puerto OBD consigue la

alimentación suficiente. Una vez este todo conectado, véase ilustración 5.11, procedemos al arrancado del vehículo.

ILUSTRACIÓN 5.11 OBD, ARDUINO Y PC

Desde que se introduce la llave en el contacto vemos que en el cable OBD ya empiezan a encenderse distintas luces azules, al cabo de un rato se apagan, y aunque el coche aun no esté en marcha algunas variables OBD ya pueden ser monitorizadas.

Para comprobar que el sistema funcionaba correctamente, se creó un programa para al Arduino que monitorizara el valor de las **RPM** del motor y lo mostrara por pantalla. Se eligió esta variable porque era fácilmente modificable aun con el vehículo parado y además podía verse su valor en el cuadro del coche, de esta forma se podía testear que la medición era correcta.

Una vez comprobado que el valor obtenido vía Arduino coincidía con el valor mostrado por el cuadro de control del coche se procedió a medir las demás variables.

Las variables que fueron monitorizadas son:

- **Carga del motor (ELD):** Es equivalente al par motor que tiene que suministrar un motor para vencer a la resistencia que se opone a su movimiento.
- **Temperatura del refrigerante del motor (CTP):** Es la temperatura del refrigerante del motor.
- **Revoluciones del motor (RPM):** Numero de vueltas por minuto a la que gira el motor.
- **Velocidad (SPD):** velocidad a la que se desplaza el vehículo en un determinado momento
- **Temperatura del aceite del motor (EOT)**: Temperatura a la que se encuentra el aceite del motor.
- **Ratio de combustible en función del tiempo (EFR):** expresado en Litros partido hora.
- **Flujo de masa de Aire (MAF):** Es la cantidad de aire que ingresa al motor
- **Proporción de aire – combustible (FA):** es la relación entre la masa de aire para combustible presente en el motor.
- **Temperatura (T):** Temperatura a la que se encuentra el motor

El código resultante fue el siguiente:

```
byte pids[]={"ENGINE_LOAD","COOLANT_TEMP","RPM","SPEED",
  "ENGINE_OIL_TEMP","ENGINE_FUEL_RATE","MAF_FLOW","Fuel-air",
  "T aire"};

int valores[9];

  Serial.println("Medimos las variables del coche");
  for (int i=0; i<Total_Variables; i++){
    valores[i]=obd.getValor(pids[i]);
  }
```

ILUSTRACIÓN 5.12 RUTINA ENCARGADA DE LA MEDICIONES

En primer lugar y como se ve en la ilustración 5.12 hemos creado un vector con todos las variables que deseamos monitorizar. Después solo necesitamos pasarle a la función getValor los PIDS y estanos devolverá el valor devuelto por el motor.

De esta manera guardaríamos todos los valores devueltos por el protocolo obd-ii en el vector valores.

Hay que aclarar que primero hay que definir cuáles son los valores en hexadecimal para cada variable que deseamos monitorizar. La tabla con todas las equivalencias se encuentra en el anexo PIDS de OBD-II.

Tras realizar las mediciones en el vehículo 2, observamos que algunas variables no eran monitorizadas, concretamente la EOT y la EFR, que daban 0 de valor mientras que FA y TEM no devolvían ningún valor.

El vehículo 1 solo se utilizó para las primeras pruebas en el solo se midió el valor de RPM.

En el vehículo 3 se monitorizaron las mismas variables que en el vehículo 2.

En todos los casos, es decir en todas las pruebas en todos los vehículos, ha habido variables que no han devuelto valor por parte del coche. Pese a esto todas las variables se han mantenido como variables a monitorizar por el dispositivo, aunque adecuando el software para que no se produzcan errores cuando estas no sean medidas o no sean valores esperados.

5.5. Fase 5. Agrupar Funciones

Una vez finalizados los experimentos individuales se procedió a unificar todos los códigos necesarios para cumplir los objetivos de las fases anteriores en un único programar. Para que de esta manera el arduino fuera capaz de realizar a la vez y de manera ordenada todas las tareas necesarias para el cumplimiento de los objetivos.

Para una mayor comodidad a la hora de programar y para facilitar la comprensión del condigo final, se creó una librería para Arduino que recoge todas las funciones y métodos necesarios para el desempeño de las tareas anteriores, la librería está incluida en el cd adjunto,

Además, para asegurar la integridad de todos los componentes y que estos no se deterioraran en las pruebas se proveyó al dispositivo de una caja fabricada con piezas de Lego que permite que todo quede compactado y con un formato fácilmente transportable

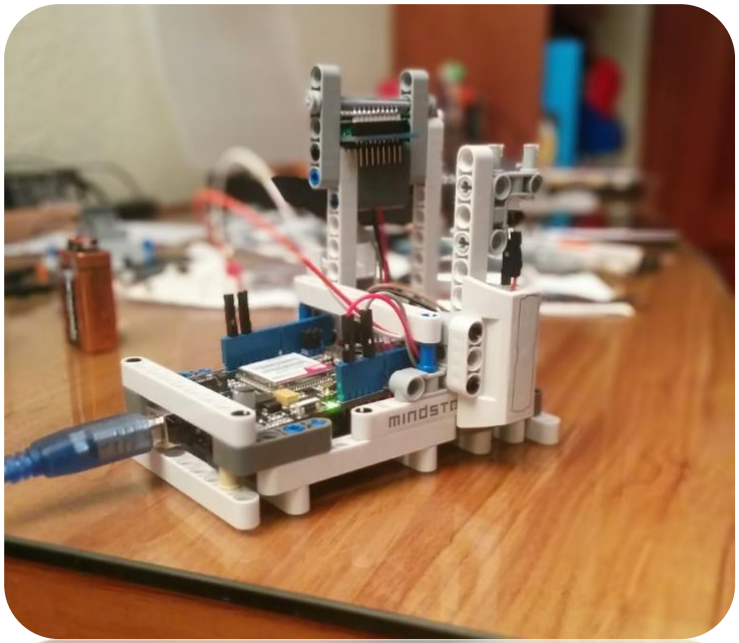

ILUSTRACIÓN 5.13 CAJA CONTENEDORA 1

Este fue el primer encapsulado, permitía una gran maniobrabilidad a la hora de realizar cambios de hardware, pero no ofrecía la suficiente estabilidad a la hora de colocar el dispositivo en el coche. Aunque la idea de poner la antena GPS en alto era ventajosa, no suponía una mejora

significativa en cuanto la mejora de la señal, presentando sin embargo un riesgo para la integridad de la misma al estar tan expuesta.

Por estos motivos se decidió realizar un cambio en el diseño de la caja contenedora, dando como resultado la versión 2

Además del cambio de forma, se le añadieron unas patillas de goma que mejoran considerablemente su agarre y estabilidad, aguantando sin cantearse el paso de badenes y demás obstáculos que se pueden encontrar en la carretera.

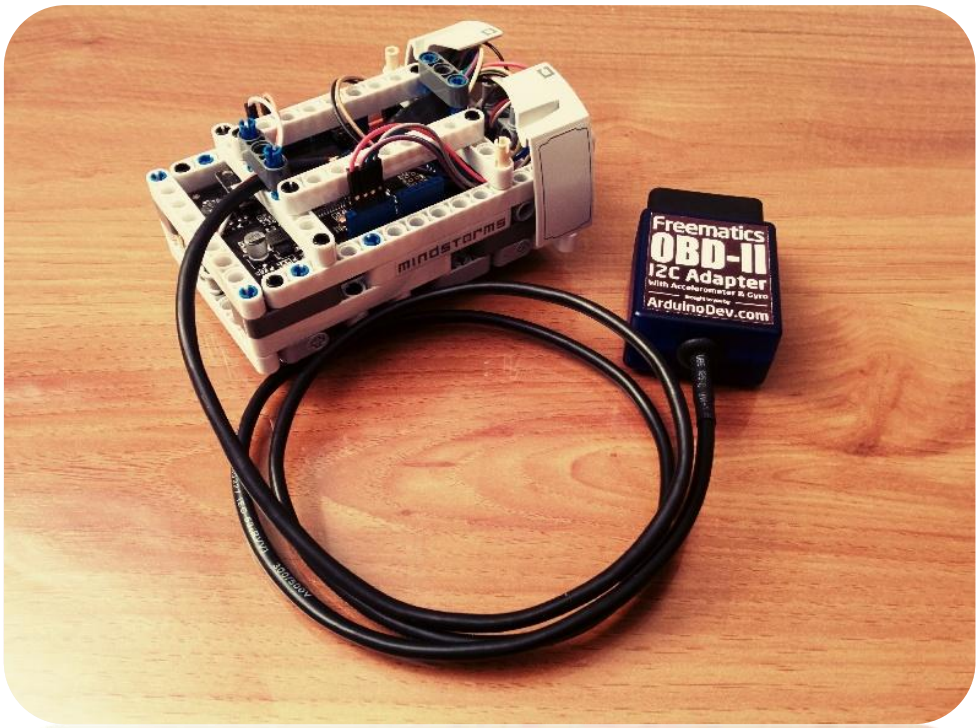

ILUSTRACIÓN 5.14 CAJA CONTENEDORA 2

El esquema final del dispositivo seria como el que se indica en la siguiente figura:

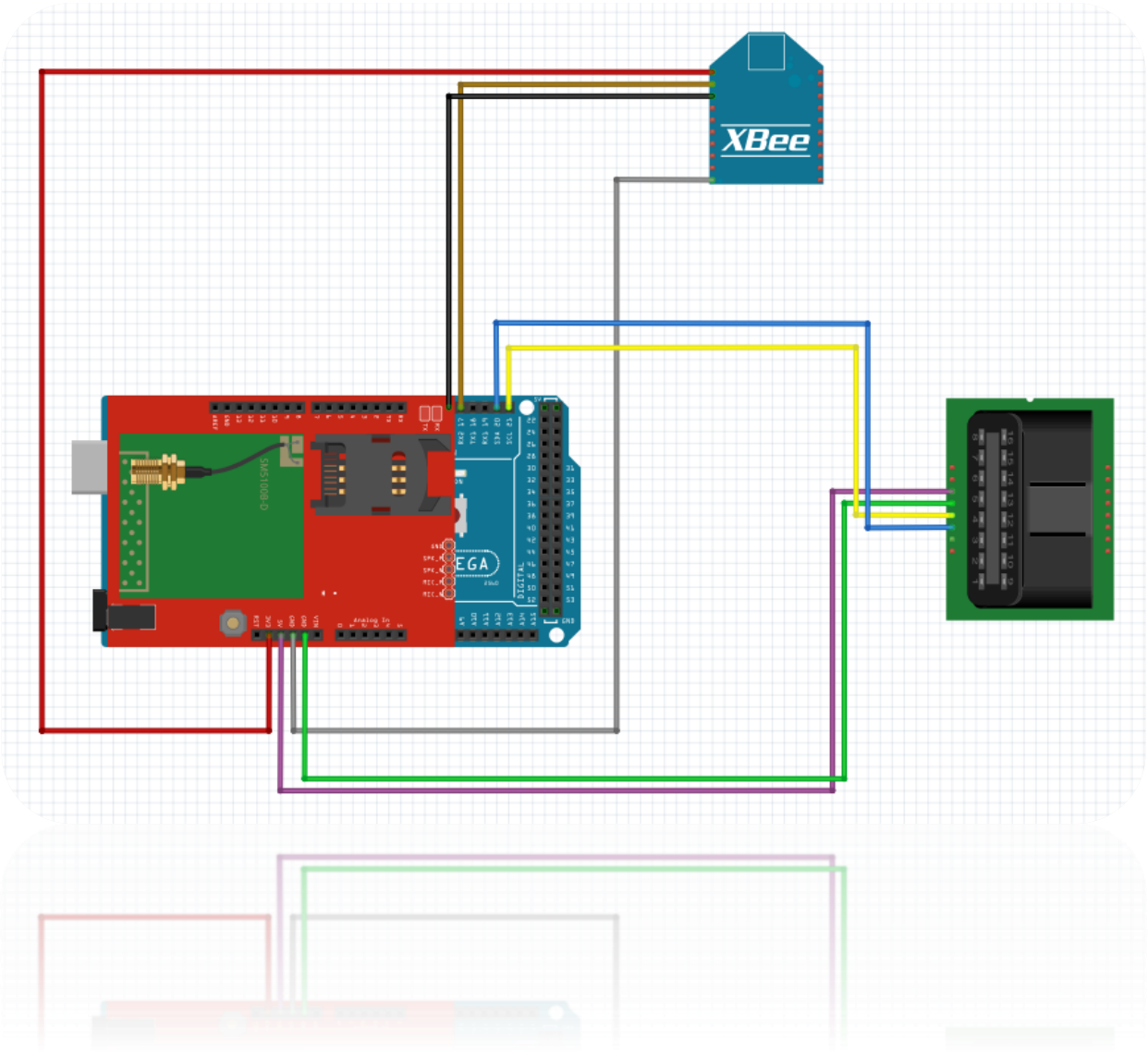

ILUSTRACIÓN 5.15 ESQUEMA FINAL DEL DISPOSITIVO

También se doto al dispositivo de un led indicativo para que el usuario pueda comprobar en todo momento cuando recopila y manda información o está en periodo de espera.

Aquí concluiría la parte relativa al dispositivo hardware, tanto a nivel hardware como a nivel software.

5.6. Fase 6. Aplicación web Cliente

En esta fase final se pretende implementar una solución basada en web, que permita al usuario acceder a la información de manera sencilla, clara y ordenada.

Para ellos se crea la aplicación web llamada Auto_client, esta consta de tres páginas web con tecnología html, y una cuarta que hace uso de la tecnología jsp.

ILUSTRACIÓN 5.16 INDEX.HTML

- **Index.html**

  Esta es la página web principal, ver ilustración 5.16, y sirve como menú y acceso a las demás y nos da información de las virtudes de la plataforma.

- **Graph.jsp**

  Es la encargada de realizar las graficas

Por último y más importante está

- **home.jsp**

  Esta página hace de panel de control para el usuario, desde aquí el usuario se validara y podrá acceder a toda la información de los viajes realizados y los datos medidos por el dispositivo

  Esta página web se encarga de validar al usuario y de mostrar la información del viaje seleccionado

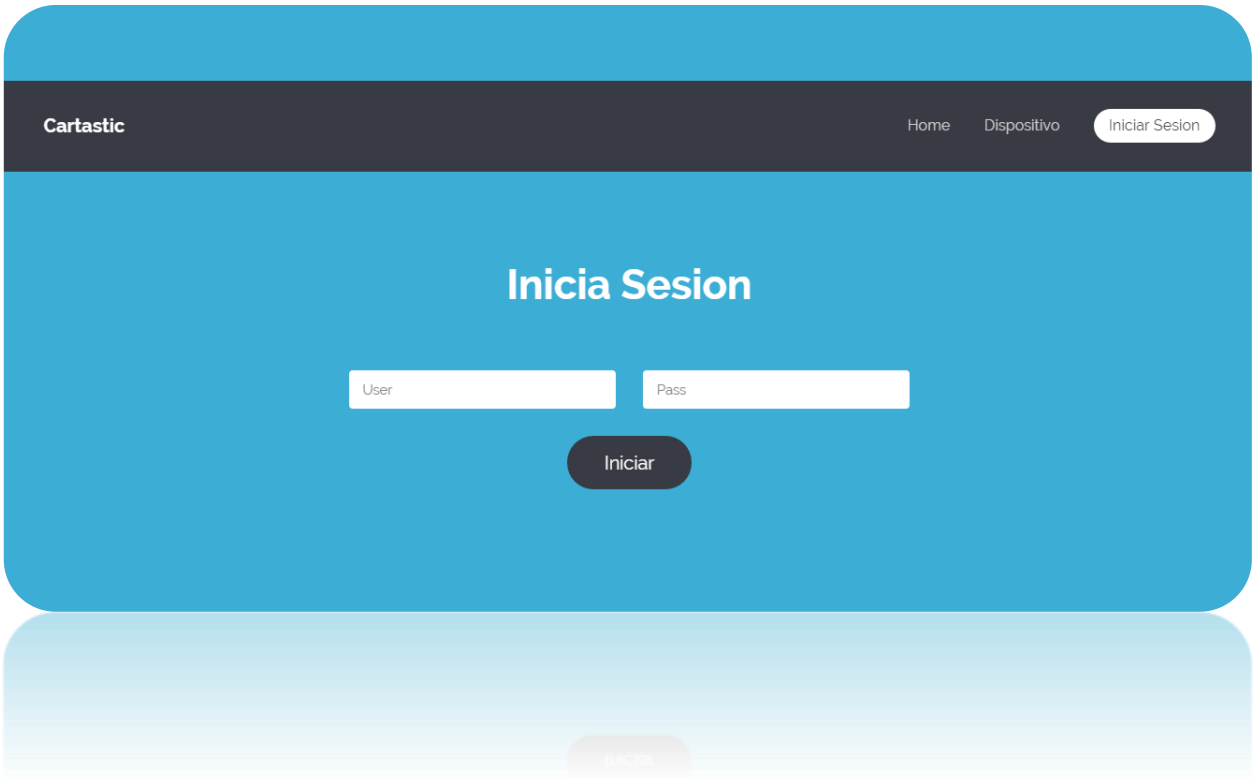

ILUSTRACIÓN 5.17 HOME.JSP INICIO DE SESION

Al acceder a la aplicación nos encontramos en primer lugar con una página de bienvenida, en ella encontraremos información general de nuestro sistema. Desde aquí podremos acceder a nuestra área personal, para ello solo tendremos que pinchar donde pone home.

Tras pulsar se nos mandara a una nueva página web, home.jsp. En primer lugar tendremos que identificarnos y pulsar en iniciar. Si nuestro usuario y contraseña son correctos accederemos a un menú, donde mediante una pestaña desplegable podremos acceder al identificador de viaje que queramos ver. Unas vez seleccionado pincharemos en mostrar y se nos mostrara el mapa con el ruta dibujada en el con unas marcas.

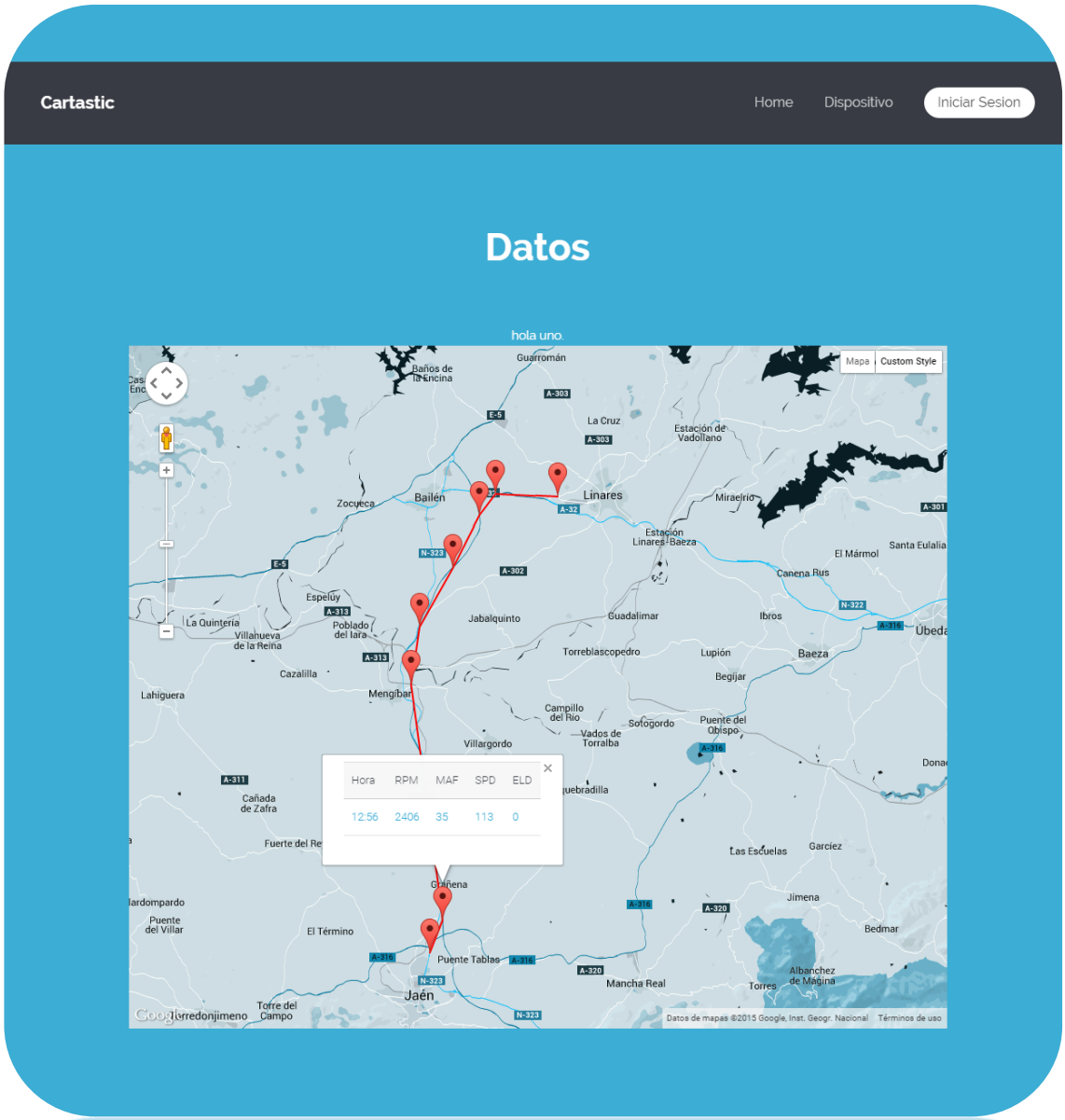

ILUSTRACIÓN 5.18 HOME.JPS MOSTRANDO UN VIAJE

Cada marca representa una medición de nuestro dispositivo. Si queremos ver como una variable se ha comportado a lo largo del tiempo que ha durado el viaje, basta con hacer click sobre el nombre de dicha variable. Automáticamente se nos abrirá una ventana emergente con la gráfica de dicha variable a lo largo del tiempo.

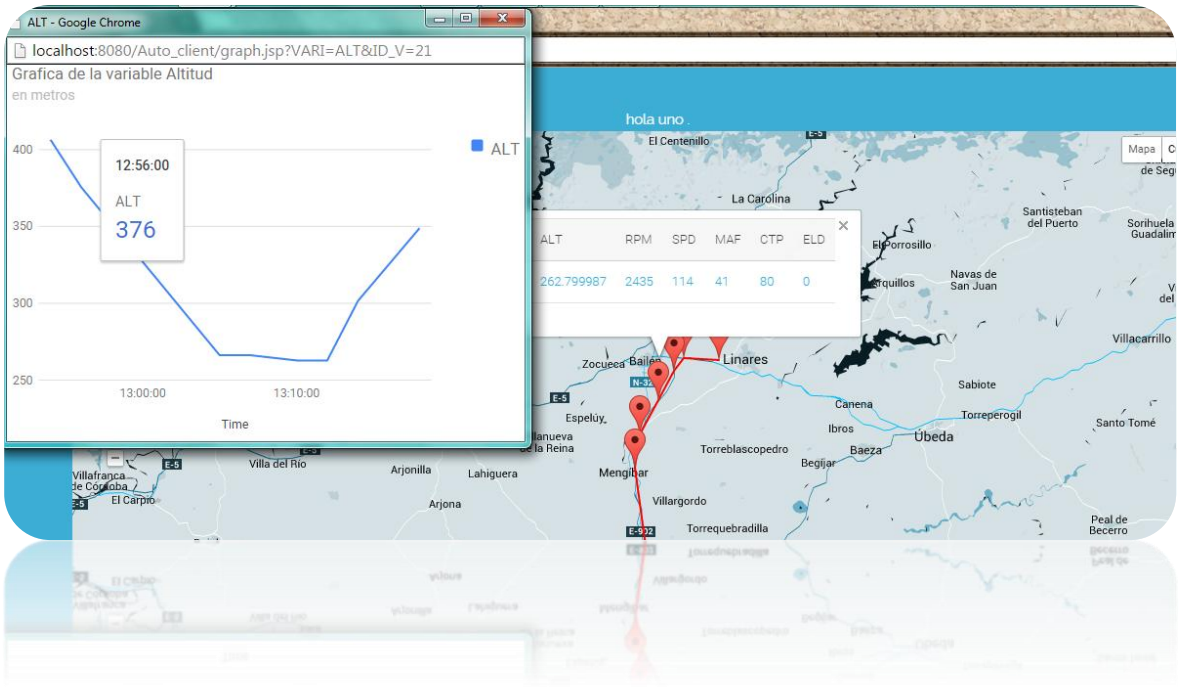

ILUSTRACIÓN 5.19 HOME.JPS Y GRAPH.JSP

6. RESULTADOS Y DISCUSIÓN

En esta sección hablaremos sobre los resultados obtenidos en los distintos experimentos llevados a cabo.

Primero comentaremos los viajes realizados en el experimento y a continuación presentaremos los datos.

Después se discutirá la información y comentaremos obtenida

6.1. Datos

Se han realizado varias sesiones de pruebas con el vehículo 2, Citroën C4, sin salir del garaje, conectando el dispositivo al coche y a un PC para comprobar que el dispositivo era capaz de medir y que los valores obtenidos coincidían con los esperados, es decir, los ofrecidos por el cuadro de control del coche.

Después se realizaron dos viajes, ambos con el mismo origen destino, concretamente partiendo de Linares y con llegada a Jaén y vuelta de Jaén dirección Linares.

Linares – Jaén

Viaje 1	Viaje 2

ILUSTRACIÓN 6.1 IDA 1

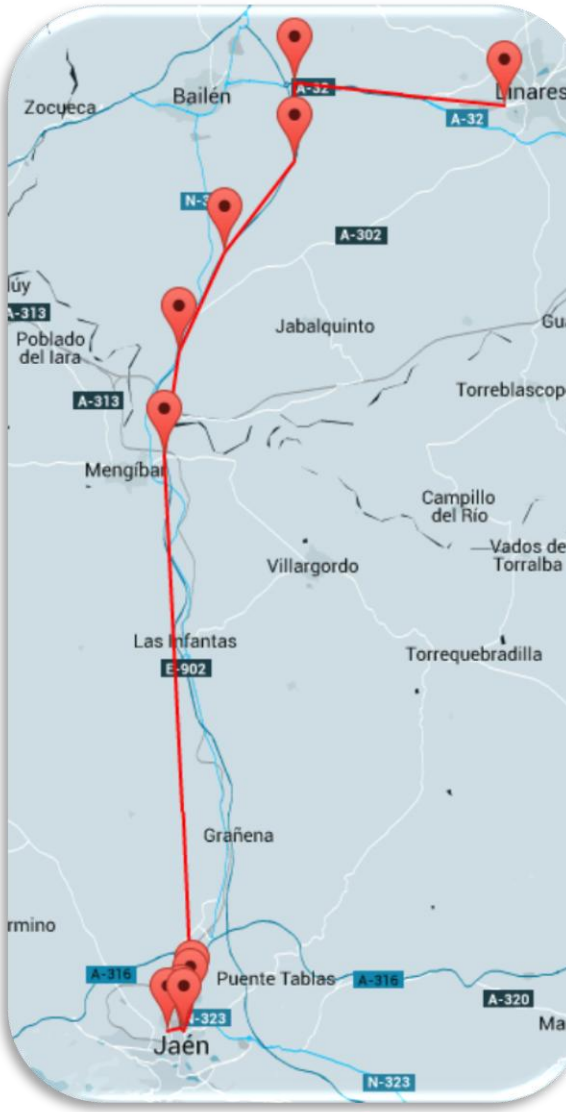

ILUSTRACIÓN 6.2 IDA 2

En estas dos ilustraciones, ilustracion 6.1 e ilustracion 6.2, vemos el recorrido seguido en los dos viajes en el trayecto Linares - Jaen

Jaén – Linares

Viaje 1	Viaje 2

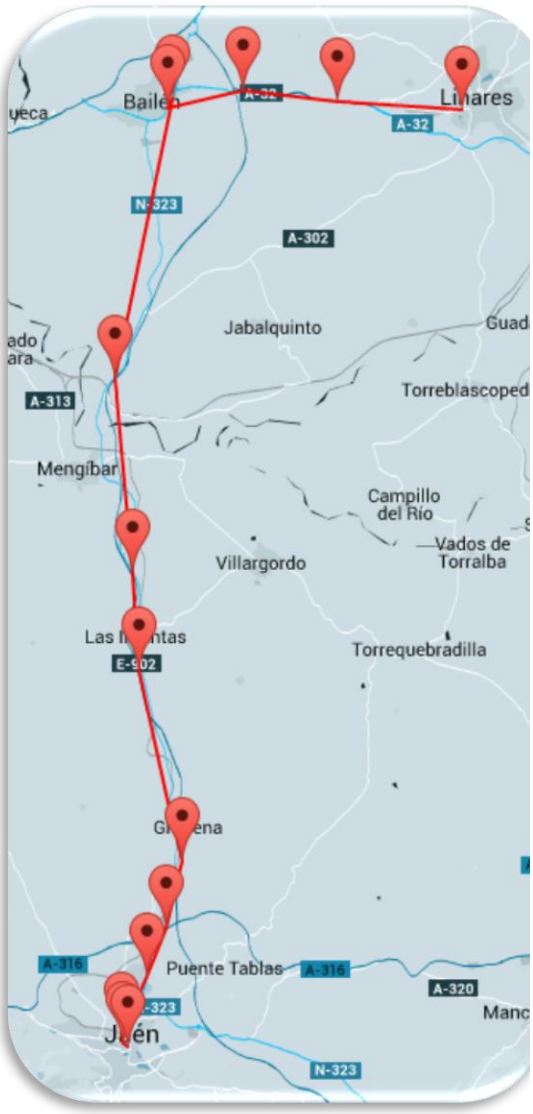

ILUSTRACIÓN 6.3 VUELTA 1

ILUSTRACIÓN 6.4 VUELTA 2

En las ilustraciones 6.3 y 6.4 vemos el viaje seguido en el trayecto Jaén Linares

(En el viaje de vuelta primero, se realizó un pequeño desvió entrando en la ciudad de Bailen).

A continuación se mostraran los datos obtenidos en ambos viajes. La información se presenta de manera pareada para poder ver con mayor claridad las diferencias entre los dos viajes.

Viaje Linares – Jaen

- **Altitud**

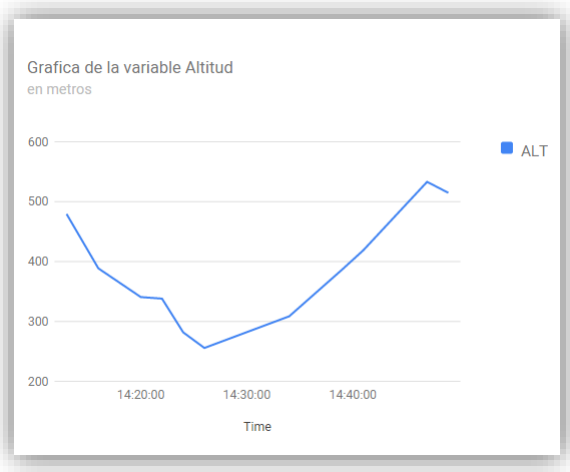

ILUSTRACIÓN 6.5 ALTITUD IDA1

ILUSTRACIÓN 6.6 ALTITUD IDA2

En estas graficas vemos la diferencia de altitud experimentada en el viaje, la mediccion la hemos obtenido de la antena GPS que incluye el dispositivo. Vemos que aunque la informacion no es precisa, si es coherente, con una recalibracion o un factor de correccion podria ser mas exacta.

➢ RPM

ILUSTRACIÓN 6.7 RPM IDA1

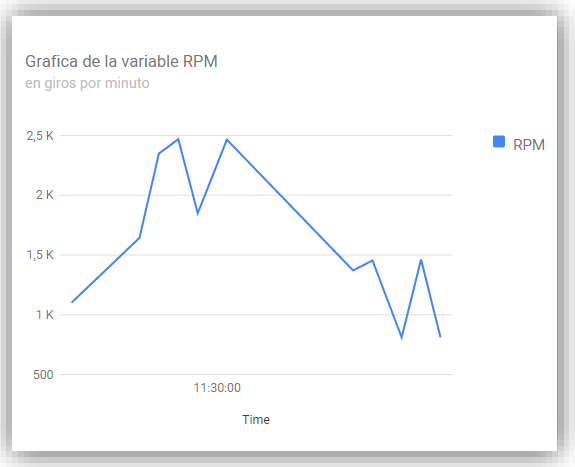

ILUSTRACIÓN 6.8 RPM IDA2

En estas graficas podemos ver la evolucion de las revoluciones del motor, medicion realizada por el protocolo OBD-II. Vemos, sobre todo en la primera, como se distingue perfectamente la etapa del viaje de fuera de la ciudad de la parte atraves de la ciudad.

➢ Velocidad

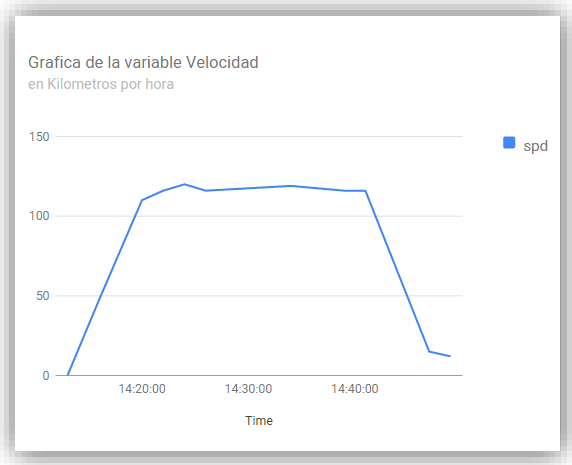

ILUSTRACIÓN 6.9 VELOCIDAD IDA1

ILUSTRACIÓN 6.10 VELOCIDAD IDA2

En estas otras graficas vemos la variacion de velocidad, medidas tomadas por el protocolo OBD-II, en ellas igual que en el caso anterior vemos diferenciada las medicciones realizadas en ciudad y las mediciones de ciudad.

➢ MAF

ILUSTRACIÓN 6.11 MAF IDA 1

ILUSTRACIÓN 6.12 MAF IDA 2

En estas ilustraciones vemos la variacion del flujo de masa de aire que entra en el motor, estos valores se han obtenido gracias al protocolo OBD-II.

➢ CTP

ILUSTRACIÓN 6.13 CTP IDA 1

ILUSTRACIÓN 6.14 CTP IDA 2

En estas graficas vemos como fue variando la temperatura del refrigerante del motor, los datos fueron extraidos por el protocolo OBD-II.

➢ Consumo

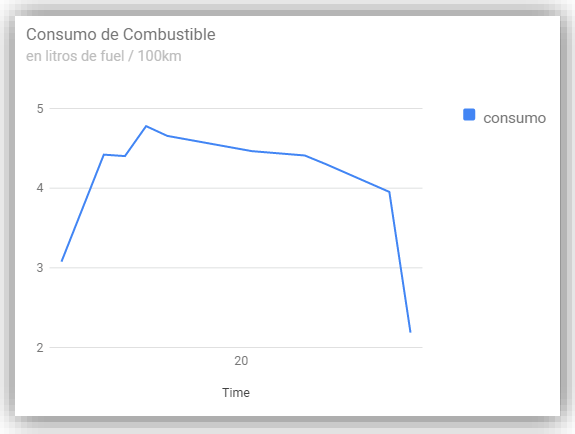

ILUSTRACIÓN 6.15 CONSUMO IDA 1

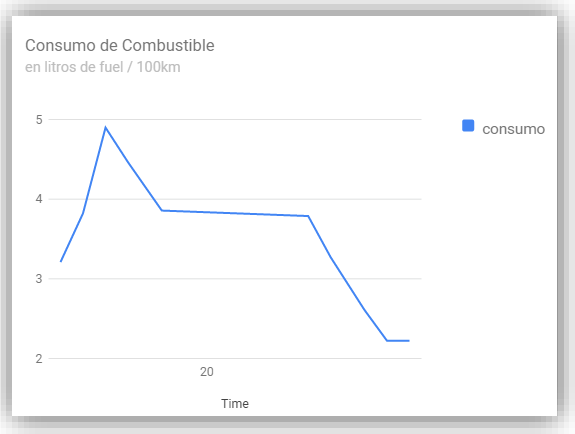

ILUSTRACIÓN 6.16 CONSUMO IDA 2

Por ultimo tenemos estas graficas en las que vemos el consumo del vehículo por sectores. Esta variable no se mide directamente, si no que ha sido calculada en función de algunos parámetros obd. El primer método utiliza las variables MAF y Velocidad medidas del coche y se describe asi:

```
Distancia instanea: D = Velocidad * t/3600
Fuel instantaneao: F = 1/(14.75 * 6.26) * MAF * t/60
MPG = D / F
```

ILUSTRACIÓN 6.17 FORMULA CALCULO CONSUMO

Para el segundo las variables obd necesarias son las RPM, la carga del motor. El método está recogido en el anexo anexo cálculo de consumo.

Vemos que en este recorrido, en el viaje 2 hay en las gráficas picos, esto se debe a factores externos a la hora de realizar un viaje, en nuestro caso se debe a tráfico en la carretera y mucho tráfico a la entrada en Jaén.

Por el otro lado tenemos los viajes de vuelta

Viaje Jaen – Linares

- **Altitud**

ILUSTRACIÓN 6.17 ALTITUD VUELTA1

ILUSTRACIÓN 6.18 ALTITUD VUELTA2

En estas ilustraciones vemos la variacion de altitud. Las medicciones fueron realizas por la antena GPS, vemos que comoen el caso anterior, el error cometido es alto.

➢ RPM

ILUSTRACIÓN 6.19 RPM VUELTA1

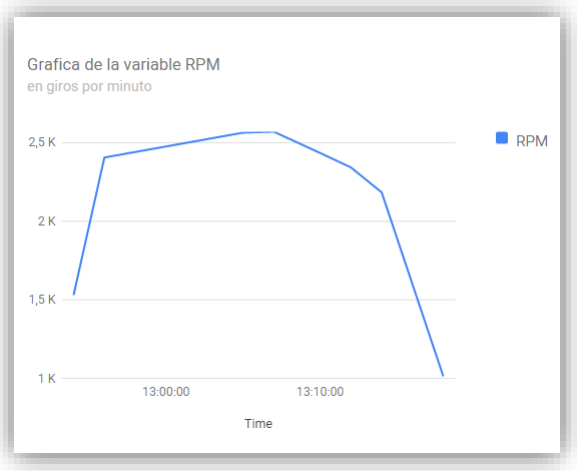

ILUSTRACIÓN 6.20 RPM VUELTA2

En estas graficas vemos la variacion de las revoluciones por minuto del motor a lo largo de los viajes, los datos fueron obtenidos del protcolo OBD-II.

➢ Velocidad

ILUSTRACIÓN 6.21 VELOCIDAD VUELTA1

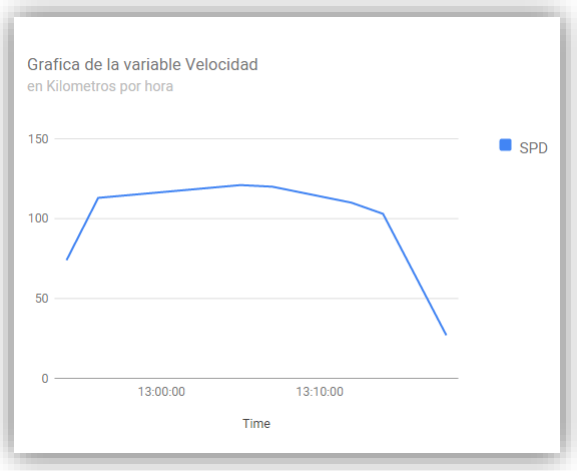

ILUSTRACIÓN 6.22 VELOCIDAD VUELTA2

Aquí podemos ver las graficas con la variacion de velocidad. Estos datos han sido obtenidos por el protcolo OBD-II.

➢ MAF

ILUSTRACIÓN 6.23 MAF VUELTA1

ILUSTRACIÓN 6.24 MAF VUELTA2

En estas graficas vemos la comparacion de la variable flujo de entrada de masa de aire al motor durante el viaje. Los datos han sido obtenidos por el protocolo OBD-II.

➢ CTP

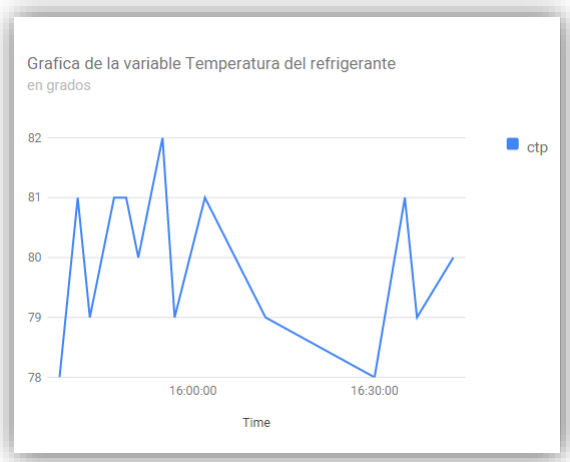

ILUSTRACIÓN 6.25 CTP VUELTA1

ILUSTRACIÓN 6.26 CTP VUELTA2

Aquí tenemos la variacion de temperatura experimentada por el refrigerante del motor a lo largo de los dos viajes. Los datos han sido monitorizados gracias al protocolo OBD-II.

➢ Consumo

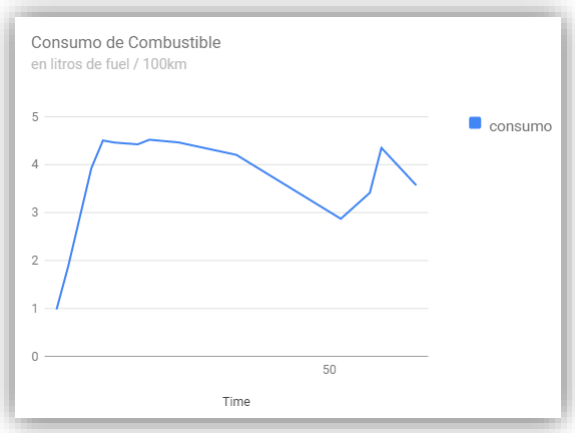

ILUSTRACIÓN 6.27 CONSUMO VUELTA1

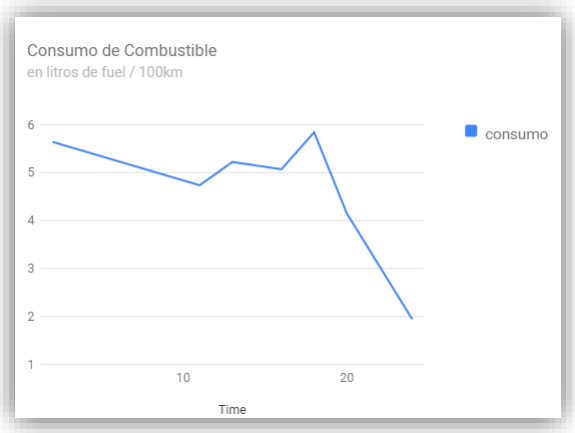

ILUSTRACIÓN 6.28 CONSUMO VUELTA2

En ultimo lugar tenemos las graficas con el calculo de consumo de combustible por tramos. Recordamos que este valor no es medido si no calculado. Ver Anexo Calculo de Consumo.

En este viaje, sobre todo en la vuelta 2, vemos que las curvas son más suaves que en los demás, esto se debe a que en este viaje tuvimos condiciones muy favorables en la carretera sin apenas tráfico y unas condiciones climatológicas buenas.

6.2. Discusión

Aunque el dispositivo está programado para tomar muestras cada minuto y medio, realmente las muestras no eran tomadas a esa frecuencia, creemos que esta pérdida de muestras se debe a dos factores.

Por un lado el retardo que se obtiene mientras el GPS obtiene conecta y obtiene una posición valida del satélite y por otro la latencia que se manifiesta cuando la calidad de la red de datos telefónica no es todo lo buena que cabría esperar, tal problema lo vemos claramente en el segundo viaje, tanto a la ida como a la vuelta, cuando en la zona cercana a Mengibar el dispositivo no envió ninguna muestra, se comprobó otro dispositivo que hiciera uso de la red y se confirmó que pese a que el operador que en esa zona hay cobertura de hasta 3G, lo cierto es que ese día al pasar por ahí el indicador del Smartphone indicaba que la red disponible era EGPRS, una red con unas características inferiores a la GPRS normal.

No obstante podemos considerar los viajes realizados de cortos, pues no llegaron ni a una hora, y los kilómetros recorridos fueron unos 50. El dispositivo está pensado para viajes más largos de más distancia o de más rato en funcionamiento.

Esto nos hace pensar que quizás sea buena idea ir guardando los datos en el Arduino y volcarlos en el servidor cada cierto tiempo, en lugar de ser como ahora que mide y envía, así de esta manera no perderíamos información por latencias en la red.

Hemos comprobado en los experimentos que el dispositivo es estable y sólido, aguanta bien los viajes y las vibraciones típicas de un coche en movimiento, además no ha tenido problemas

de funcionamiento por el calor, pese que en los dos viajes estuvo sobre el salpicadero y expuesto al sol todo el rato.

Mencionar las latencias producidas por las interfaces móviles que traban en parte la labor del dispositivo ya que reducen la frecuencia en la que el dispositivo envía los datos.

Por último, hablar sobre los vehículos y el protocolo OBD, hemos sufrido una gran falta de información oficial por parte de los fabricante. Toda la información referente al protocolo OBD-II ha sido obtenida a través de comunidades de usuarios que compartieron sus experiencias en internet. Sin ir más lejos, el hecho de encontrar el puerto de comunicaciones se convertía en una tarea tediosa, ya que su ubicación no era mencionada en ninguno de los manuales adjuntos al vehículo por el fabricante, de nuevo se tuvo que recurrir a comunidades especializadas para obtener la ubicación para cada modelo.

Además existe otro inconveniente y es que cada modelo incluye unos sensores si y otros no, algo que creemos debería de estar algo más estandarizado. Por ejemplo que todos los vehículos del mismo segmento o de la misma gama, ofrezcan los mismos sensores sin importar el fabricante.

7. CONCLUSIONES

En esta sección hablaremos de las conclusiones alcanzadas tras la realización de Trabajo y de la plataforma tras su uso en las pruebas.

7.1. Objetivos

En primer lugar concluir en que todos los objetivos del trabajo han sido completados satisfactoriamente.

Por un lado teníamos el primer objetivo que consistía en la creación de una aplicación basada en Arduino que permitiera la monitorización de la información del motor haciendo uso del protocolo OBD. Esta parte resulto bastante sencilla, en primer lugar había que importar la librería necesaria, establecer una comunicación con el vehículo y proceder a la petición de los valores de las variables a monitorizar, después simplemente hubo que guardarlas en un vector para su posterior uso. Esta parte no presento una gran complejidad, más allá de que hay que tener en cuenta que no todos los vehículos poseen las mismas sondas y por tanto no pueden acceder a la misma información, por tanto, habría que cerciorarse primero de si el vehículo que se quisiera monitorizar es compatible y ofrece información interesante.

Además como extra nos pareció interesante añadir además la información del tiempo, información sobre la posición, por eso se le añadió al dispositivo una interfaz de comunicación con el servicio GPS, el uso de la antena GPS por parte del Arduino no añadió demasiada

complejidad al código, aunque como pega, esta función si suele añadir bastantes retardos a la hora de monitorizar la información, debido a que el GPS suele necesitar un tiempo indeterminado para establecer comunicación con los satélites, unas veces es despreciable y otras veces se demora en varios minutos.

En segundo lugar teníamos el objetivo de una vez recogidos dicho datos almacenarlos en una base de datos remota. Esta parte si resulto algo más difícil de realizar, se invirtió gran cantidad de tiempo hasta conseguir una comunicación con una aplicación remota. Además no debemos de olvidar que para esta parte se hace uso del servicio de telefonía móvil, por tanto que el dispositivo funcione bien y envíe datos con la frecuencia establecida depende del estado y disponibilidad del servicio, y de la calidad de la señal y la disponibilidad de cobertura.

Por ultimo está el objetivo de crear una aplicación web que sea capaz de extraer la información de la base de datos y la muestre de manera una manera atractiva y clara. Para el desempeño de este objetivo hicimos uso de las APIs de Google, Maps y chart, Gracias a la tecnología JSP, utilizar y modificar los JavaScript necesarios para su visualización ha resultado realmente sencillo, aunque tanto Maps como chart han dado muchos problemas a la hora de cuadrarse y ajustarse a los tamaños de la página web, aunque en líneas generales su uso es sencillo gracias a la gran cantidad de información que hay disponible en la propia página de las APIs.

Podemos concluir con que el sistema completo es útil y practico. Tener en cuenta que no existe en el mercado nada similar, hay dispositivos parecidos pero se limitan solamente a mostrar

la información de manera puntual o a guardarla en la memoria del aparato, no hemos encontrado ninguno que convine la tecnología OBD con la posibilidad de almacenar dicha información en un servidor remoto. Esta es sin duda la gran ventaja de nuestra plataforma, poder disponer de la información perfectamente ordenada y clasificada de manera automática.

Además la información es procesada para realizar un cálculo aproximado del consumo medio en cada viaje, esto supone un gran avance para poder introducir medidas de ahorro, y ver que dichas medidas suponen de verdad una mejora. O en el caso de utilizar esta plataforma para una empresa que disponga de una flota de vehículos, con el uso de los dispositivos podría detectar de manera rápida y sencilla si hay algún vehículo que presente un consumo excesivo, de esta forma se podrían realizar los cambios pertinentes para que se normalice, ahorrando así combustible para la empresa.

En resumen podemos concluir que la plataforma es útil y fiable. Ha sido probada en experimentos reales, en experimentos con viajes y vehículos en condiciones normales, además de ser puesta a prueba en situaciones controladas fuera de los márgenes de funcionamiento esperables, y el resultado ha sido siempre bueno.

Como pegas que estropean la experiencia de usuario podríamos mencionar por un lado la interfaz de conexión a la red GPRS, que en cuanto la calidad del servicio baja se resiente rápidamente y demora en exceso el envío de información.

Y por el otro lado el retardo ocasionado por el acceso a los datos de posicionamiento del GPS.

Como ventajas tenemos el poco consumo que tiene el dispositivo, lo cual hace que no sea necesario el uso de baterías externas ni de ninguna fuente de alimentación adicional, por el cable de conexión a la interfaz OBD-II consigue toda la energía necesaria para alimentar todos los periféricos. Gracias a esto el dispositivo apenas genera calor, con lo cual no hay que añadir ningún tipo de elemento de disipación de calor.

Además el tamaño hace es otra ventaja pues no es un dispositivo demasiado grande y es fácilmente ajustable en el coche, cuenta con una gran estabilidad y gracias a las patillas de goma añadidas en la segunda revisión de la caja contenedora no tenemos que preocuparnos por que el dispositivo se deslice o caiga de su posición. El dispositivo es compacto y manejable.

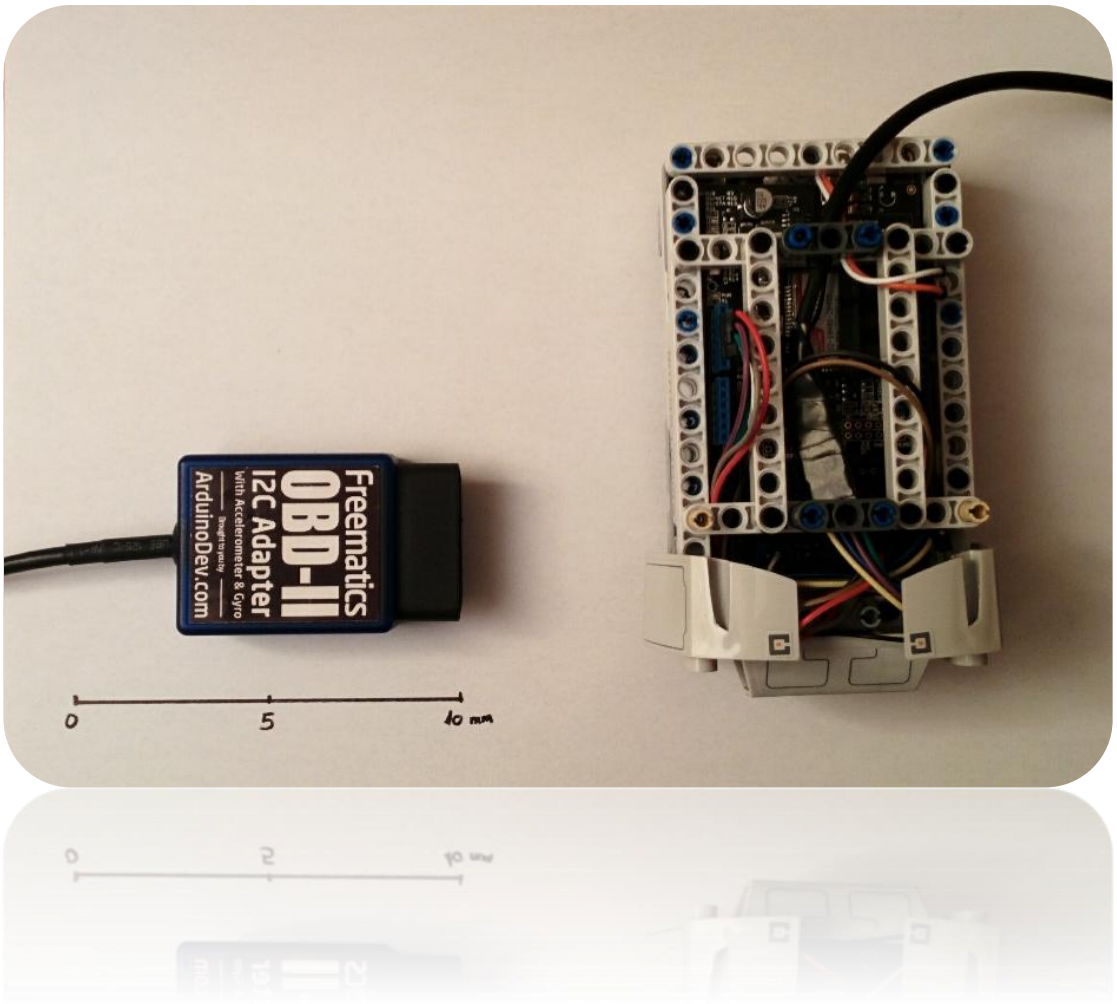

ILUSTRACIÓN 7.1 DISPOSITIVO HARDWARE

7.2. Líneas Futuras

Uno de los principales atractivos de este trabajo es que puede servir como punto de partida para una gran cantidad de trabajos posteriores, por un lado y una vez obtenidos los datos pueden implementarse patrones para que un usuario, pueda planificar un viaje, y que la aplicación realice cálculos aproximados en función de su patrón en cuanto al tiempo de duración del viaje, consumo esperado de combustible.

Además como se comentó previamente se podría mejorar el sistema de volcado de información al servidor enviado los datos por los lotes en lugar de uno como ahora, así el sistema ofrecería mayor robustez frente a los problemas de servicio originados por los retardos y latencias de la red telefónica inalámbrica.

Para poder realizar un cálculo aproximado de combustible es necesario monitorizar al menos los parámetros Flujo de masa de aire, Velocidad y Carga del motor. Según los métodos que hemos visto para el cálculo del consumo.

Otro línea futura que se podría realizar y pensando en el uso de la plataforma en flotas de vehículos, es la implementación de un sistema de puntuación para los conductores, en función de la información suministrada por los sensores, por ejemplo, atendiendo a consumo. De esta manera la empresa podría ver que conductores hacen un uso más eficiente de los recursos y quiénes no.

8. ANEXOS

8.1. PIDS OBD

Anexo incluido en el cd, en la carpeta Anexos y con nombre PIDS OBD, en este anexo se encuentra una lista de todos las variables recogidas en el Protocolo OBD-II, junto con modos de funcionamiento e información adicional sobre cómo funciona el protocolo OBD-II (En inglés).

8.2. Comandos SIM9000

Anexo incluido en el cd, en la carpeta Anexos y con nombre SIM900_AT Command Manual_V1.03. Este documento es un completo manual de uso del sistema SIM9000, incluye lista de comandos y ejemplos de uso. (En inglés).

8.3. I2C

Anexo incluido en el cd, en la carpeta Anexos y con nombre i2c. Este anexo incluye información técnica detallada de cómo funciona el protocolo I2C. (En inglés).

8.4. Calculo de Consumo

Concesionario en el que se explica cómo realizar un cálculo del consumo de un vehículo diésel haciendo uso del protocolo OBD-II en tiempo real. Anexo incluido en el cd en la carpeta Anexos y con el nombre Vehicule Fuel Consumption. (En inglés).

8.5. Manual de la Aplicación

En primer lugar, hay que instalar las dos aplicaciones web y crear la base de datos que vaya a contener la información. Para ello, se adjunta en el cd el script para la creación de la base de datos con el nombre basedatos.sql (en la carpeta “software/servidor/base de datos”).

Para instalar las aplicaciones web, basta con incluir los ficheros ob.war y Auto_client.war (incluidos en el cd, ob.war en la carpeta “software/servidor/aplicacion” y Auto_client.war en la carpeta “software/cliente”).

En nuestro servidor compatible con JSP, por ejemplo un servidor tomcat local, o bien uno remoto.

Una vez finalizada la instalación procedemos a explicar cómo se utiliza la aplicación web.

Para acceder a nuestros datos basta con acceder a la página principal de la aplicación web Auto_client, como vemos en la ilustración 8.4

Para poder iniciar sesión basta con desplegar el menú deslizable de la izquierda y pulsa en la opción de iniciar sesión, como vemos en la ilustración 8.5

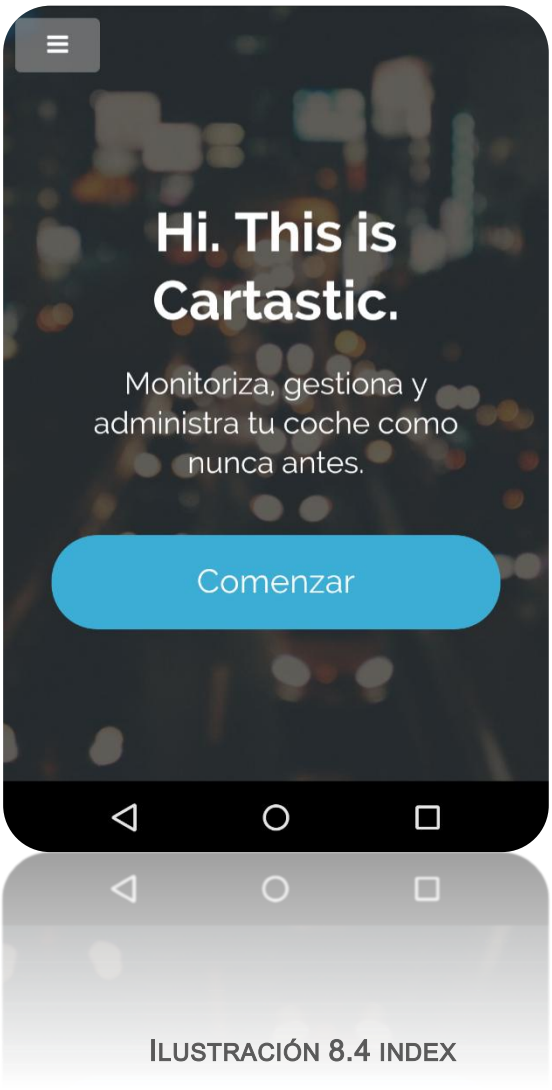

ILUSTRACIÓN 8.4 INDEX

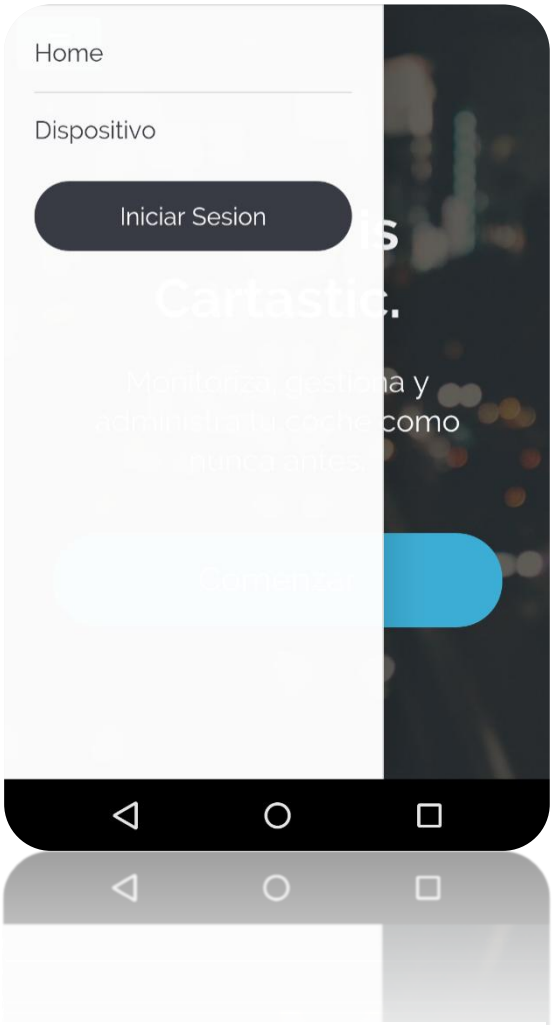

ILUSTRACIÓN 8.5 MENU LATERAL

Para poder iniciar sesión basta con desplegar el menú deslizable de la izquierda y pulsa en la opción de iniciar sesión, como vemos en la ilustración 8.5

Ahora solo necesitamos rellenar el formulario con nuestras credenciales, nombre de usuario y contraseña y pulsar en Iniciar, como vemos en la ilustración 8.6

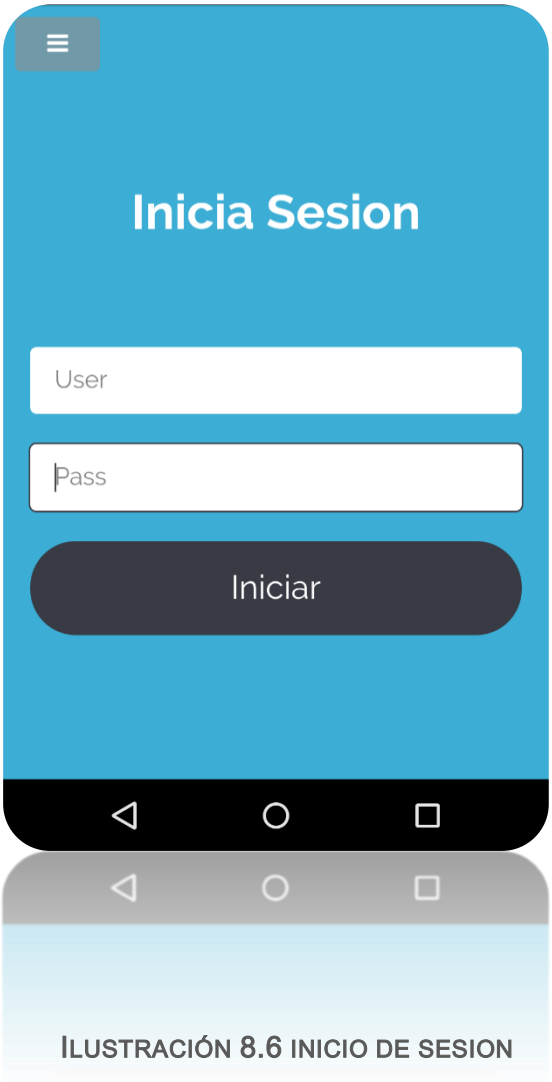

ILUSTRACIÓN 8.6 INICIO DE SESION

Al autenticarnos accedemos a un menú desplegable desde el que elegiremos el viaje que deseemos ver, ilustración 8.7.Una vez seleccionado pulsamos aceptar y se nos abrirá el mapa.

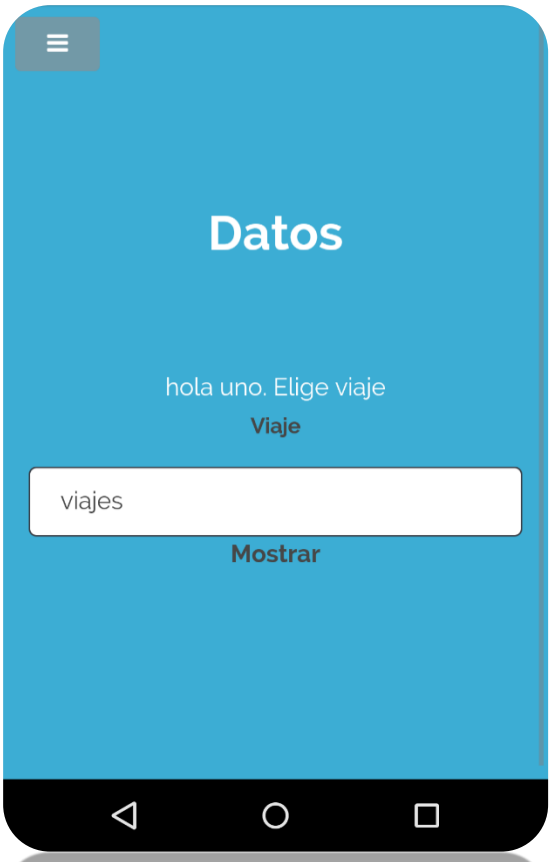

ILUSTRACIÓN 8.7 VIAJES

Aquí vemos todas las marcas que tenemos disponibles para ese viaje, cada marca es un envío de información por parte del dispositivo hardware. Si pulsamos sobre alguna de ellas veremos en detalle el valor de la mediccion de los parámetros en ese momento concreto.

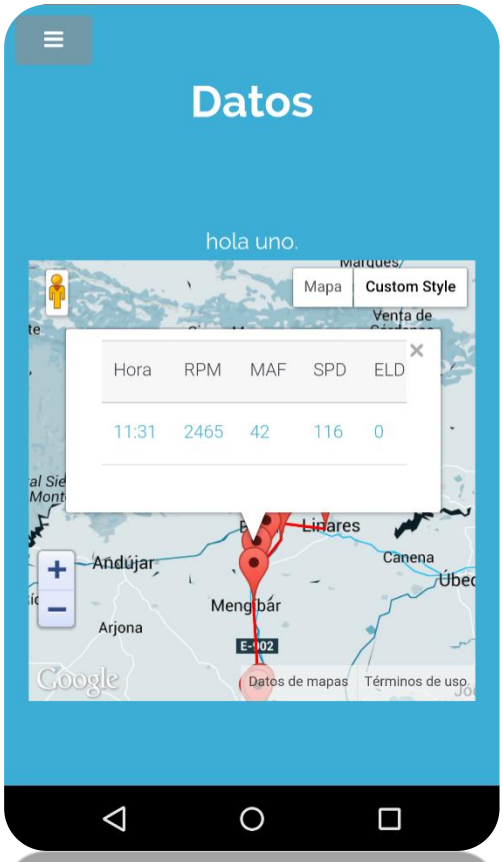

ILUSTRACIÓN 8.8 MAPA

Para ver el comportamiento de una variable a lo largo del viaje completo, basta con pulsar en cualquier marca en el nombre de la variable que deseemos ver, y se nos abrirá una gráfica en la que veremos cómo ha ido variando.

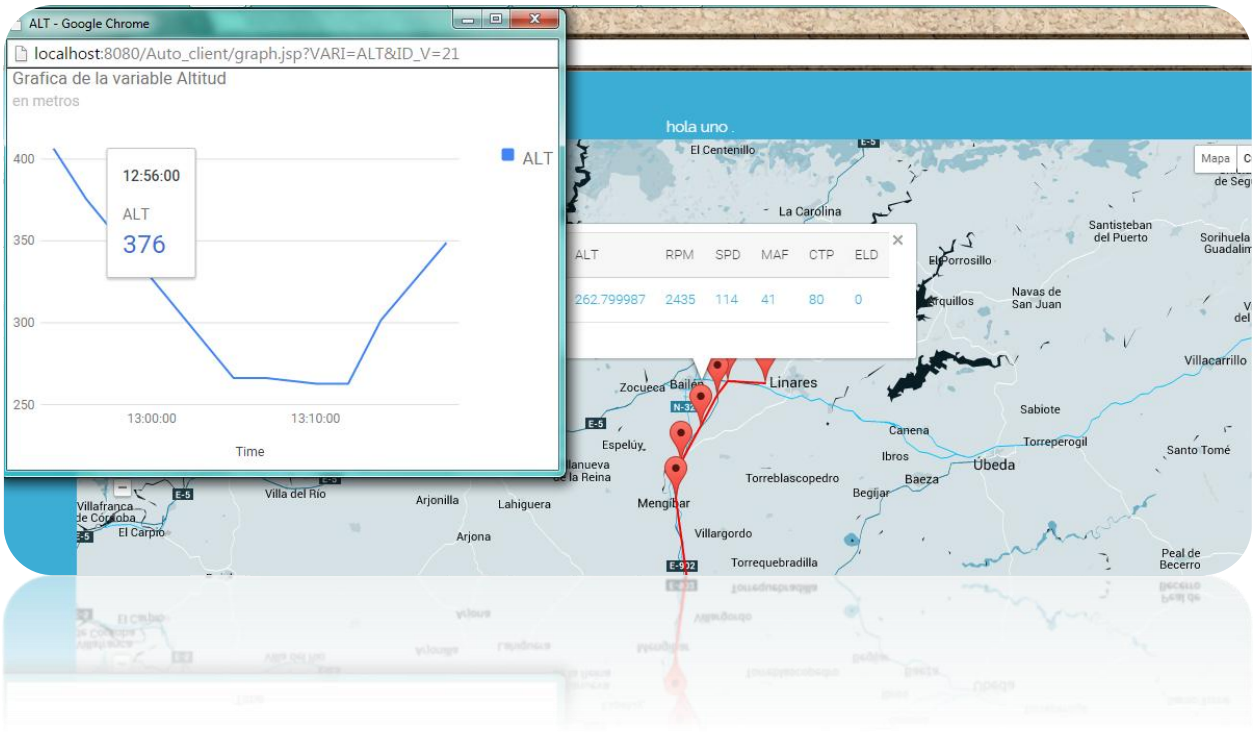

ILUSTRACIÓN 8.9 MAPA CON GRAFICA

Para concluir vemos que la aplicación web se adapta perfectamente a dispositivos móviles y es capaz de adecuar la información a la pantalla desde la que estemos consultando.

8.6. Manual del dispositivo Arduino

Para utilizar el dispositivo solo necesitamos un vehículo y conocer donde se encuentra el puerto OBD-II, una vez localizado, y con el coche apagado, aunque si ya está en marcha no pasa nada, enchufamos el cable en el puerto.

ILUSTRACIÓN 8.5 DISPOSITIVO EN FUNCIONAMIENTO

Al conectarlo se deben encender unas luces azules en la clavija OBD-II y encenderse algunos led en la placa Arduino. Cuando el canal OBD-II está disponible, se encenderá un LED de estado en un lateral del dispositivo, (un led que va cambiando de color), mientras que el led este encendido significa que el dispositivo está en primer lugar, cogiendo la información del GPS, y una vez que la posición está disponible, el dispositivo monitoriza las variables del motor y procederá a establecer una conexión en la red GPRS y enviar los datos al servidor web. Una vez mandados los datos el dispositivo apagara el led e iniciara un tiempo de espera de 1minuto antes de volver a encender el led y repetir el proceso.

El dispositivo Hardware es sencillo de usar porque no dispone de ningún botón ya que no requiere de ningún tipo de configuración por parte del usuario. Aunque el dispositivo debe de incluir un identificador de dispositivo que será necesario para poder validar los datos enviados al servidor.

9. REFERENCIAS BIBLIOGRÁFICAS

Apartado 4:

- **Arduino:**

Información general del proyecto Arduino. Disponible en la dirección: http://www.arduino.cc

- **GPS:**

Mauricio Gende Ivana Molina, "Trilateración" Facultad de Ciencias Astronómicas y Geofísicas .6 de junio de 2011. Disponible en la dirección: http://catedras.fcaglp.unlp.edu.ar/geofisica/referenciacion-en-geofisica/teoria/instrumental-y-tecnicas-topograficas/trilateracion

Portal oficial del sistema de posicionamiento GPS. Disponible en la página: http://www.gps.gov/spanish.php

Portal oficial del sistema de posicionamiento Europeo, Galileo. Disponible en:

http://www.satellite-navigation.eu/

Portal oficial del sistema de posicionamiento Chino, Beidou. Disponible en: http://en.beidou.gov.cn/

- **OBD:**

Lista completa de códigos genéricos de OBD-II. Disponible en la dirección: http://www.totalcardiagnostics.com/support/Knowledgebase/Article/View/21/0/genericmanufacturer-obd2-codes-and-their-meanings.

Directive 98/69/EC of the European Parliament and of the Council of 13 October 1998.

Disponible en la dirección:

http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1998L0069:19981228:EN:PDF

- **GPRS:**

Historia e introducción a las redes móviles de segunda generación http://www.dcs.gla.ac.uk/~lewis/teaching/Tik-111.htm

Eberspächer, J. et al. “GSM Switching, Services and Protocols” Ed. Wiley 1999

- **Java:**

Historia del lenguaje de programacion Java. Disponible en la dirección: http://www.cad.com.mx/historia_del_lenguaje_java.htm

- **JSP:**

Muñoz Expósito, J.E., P. Prado, R., García Galán, S. Aplicaciones Telemáticas. Ed. Bubok, 2011.

- **Google API:**

Página del portal para desarrolladores de Google Charts. Disponible en: https://developers.google.com/chart/?hl=es

Página principal del portal de soporte de Google para desarrolladores de Google Maps. Disponible en: https://developers.google.com/maps/?hl=es

Pliego de Condiciones

1. ESPECIFICACIONES TÉCNICAS DE LOS MATERIALES

En este apartado se describirán las características básicas y técnicas de los elementos necesarios que componen el trabajo.

Están divididos en dos grupos atendiendo a sus naturalezas, por un lados se describirán los elementos físicos (hardware) y por el otro los recursos informáticos necesarios (software).

1.1. Hardware

1.1.1. Placa de desarrollo basa en microcontrolador

Sera necesaria la utilización de un sistema basado en microcontrolador con las siguientes características.

- Microcontrolador: compatible con ATmega2560
- Tensión de alimentación: 5V
- Pines digitales: 54 (14 con PWM)
- Entradas analógicas: 16
- Corriente máxima por pin: 40 mA
- Corriente máxima para el pin 3.3V: 50 mA
- Memoria flash: 256 KB
- SRAM: 8 KB
- EEPROM: 4 KB
- 3 Puertos UART
- 1 Puerto I2C
- Velocidad de reloj: 16 MHz

ILUSTRACIÓN 1.1.1. ARDUINO MEGA

Como por ejemplo el Arduino Mega 2560

1.1.2. Modulo GPS

Sera necesario un módulo de conexión con GPS capaz de proveer al sistema de acceso al servicio GPS.

- <1 sec de tiempo Time To First Fix for Hot and Aided Starts
- -160dBm
- Alta inmunidad a las interferencias
- 4 Hz radio de actualizacion
- 2.0mm tamaño de los pines, Compatible con zócalos xBee
- UART, USB, DDC and SPI interfaces

Las características son similares a las del GPS Bee kit de Digi.

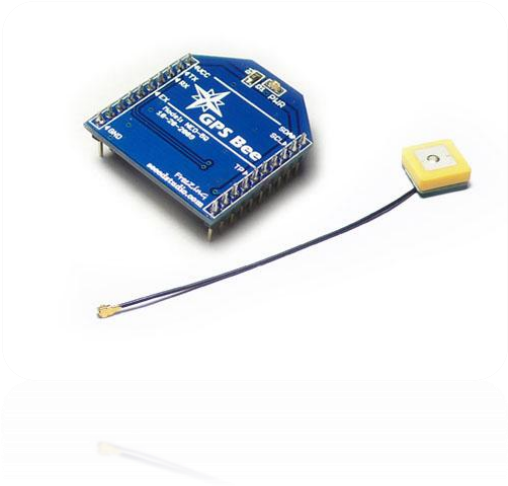

ILUSTRACIÓN 1.1.2. XBEE GPS KIT

1.1.3. Adaptor xBEE

Este adaptador compatibiliza el kit GPS Bee a los sistemas que utilizan pines de tamaño 2.54mm. Debe ser un zócalo compatible con los sistemas xBee

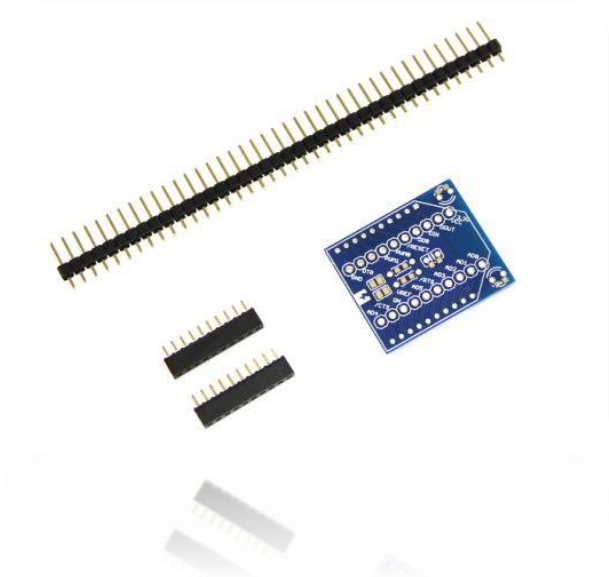

ILUSTRACIÓN 1.1.3. XBEE ADAPTER

Como por ejemplo el Xbee Breakout Kit de Digi.

1.1.4. Mochila GPRS

Sera necesario también un módulo que permita al sistema basado en microcontrolador conectarse a la red GRPS/GPS, debe de reunir las siguientes características.

- Compatibilidad con Arduino o Compatibles
- Posibilidad de elegir entre UART o Software serial
- Soporte para cuatribanda 850/900/1800/1900MHz
- Soporte para commandos AT Standard - GSM 07.07 & 07.05 and Enhanced - SIMCOM AT Commands
- Soporte para los protocolos TCP/UDP, FTP/HTTP, FOTA, MMS, AT
- Alimentacion 5v via 5V pin, 6.5~12v via Vin pin

Estas características son similares a las del GPRS Shield V2.0 de seeedstudio.

ILUSTRACIÓN 1.1.4. GPRS SHIELD V2

1.1.5. Adaptador OBD-II

También será necesario un cable que provea a nuestro Sistema de una interfaz de comunicaciones con el protocolo OBD-II

- Conector OBD-II compatible directamente con el Puerto OBD-II
- Compatible para el bus CAN J1939, J1850, ISO 9141 etc.
- Compatible con la libreria de Arduino

Como por ejemplo el cable de Freematics OBD-II i2c Adapter

ILUSTRACIÓN 1.1.5. FREEMATICS OBD-II I2C ADAPTER

1.2. Software

1.2.1. Servidor

Necesitaremos un servidor de hosting que nos provea al menos de los siguientes servicios:

- JSP
- Servidor SQL
- 300 MB de Almacenamiento

Como por ejemplo el servicio que nos ofrece la empresa de hosting hospedaxes

1.2.2. Entorno de desarrollo Arduino

Sera necesario un software que permita compilar y transferir el programa a la placa Arduino

ILUSTRACIÓN 1.2.2. IDE ARDUINO

Como por ejemplo el Arduino Software programa gratuito y descargable de la propia página de Arduino.

1.2.3. Entorno de desarrollo web

Necesitaremos un entorno de programación para poder desarrollar las aplicaciones web necesarias, debe ser compatible con:

- J2EE
- HTML
- JAVASCRIPT

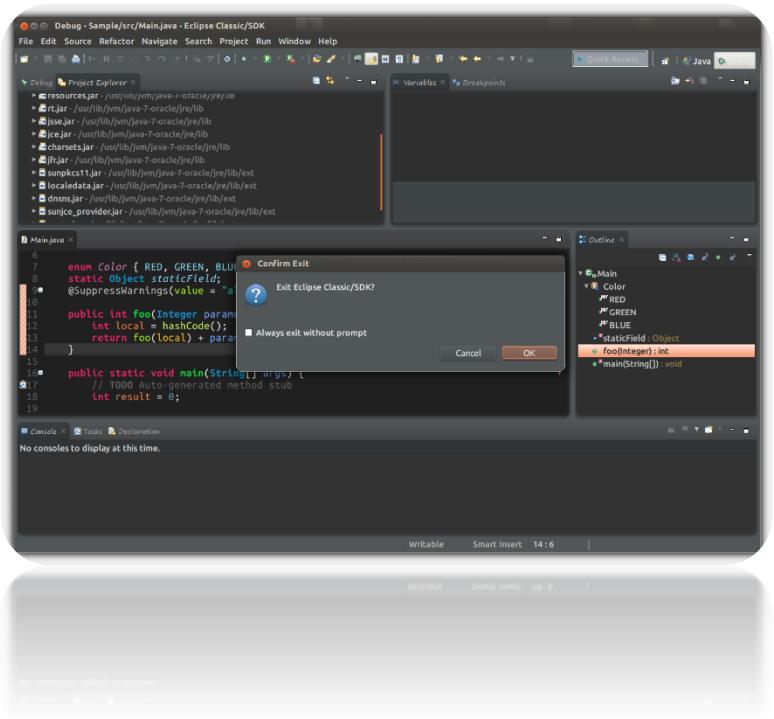

ILUSTRACIÓN 1.2.3. IDE ECLIPSE

Características similares al IDE Eclipse, de la Eclipse Foundation.

Estudio Económico

El estudio de económico se dividirá en partidas. Las partidas agrupas costes según su naturaleza. Por un lado tenemos los gastos del dispositivo, los del servidor, el software requerido y por último la mano de obra.

Al final se incluye el resumen donde se le aplica el IVA y demás costes indirectos.

1. COSTES MATERIALES

1.1. Partida 1. Dispositivo

Concepto	*Cantidad*	*Unitario*	*Subtotal*
Arduino Mega	1	39.00 €	39.00 €
GPRS Shield	1	44.07 €	44.07 €
GPS Bee	1	23.17 €	23.17 €
Adaptador OBD-II	1	43.90 €	43.90 €
xBee Adaptador	1	3.67 €	3.67 €
Cable 1 pin (pack)	1	3.67 €	3.67 €
Conexión GPRS	1	0.83 € / mes	9.92 € / año
Total			157.48 € 9.92 € / año

1.2. Partida 2. Servidor

Concepto	*Cantidad*	*Unitario*	*Subtotal*
Hosting	1	5 € / mes	60 € / año
Total			60 € / año

1.3. Partida 3. Software

Concepto	*Cantidad*	*Unitario*	*Subtotal*
IDE Arduino	1	0.00 €	0.00 €
IDE Web	1	0.00 €	0.00 €
Total			0.00 €

1.4. Partida 3. Mano de obra

Concepto	*Cantidad*	*Unitario*	*Subtotal*
Programador	300h	25.00 €	7,500.00 €
Total			7,500.00 €

2. RESUMEN DEL PRESUPUESTO

Partida	Resumen	Importe
1	Dispositivo	157.48 € mas 9.92 € /año
2	Software	0.00 €
3	Servidor	60.00 € / año
4	Mano de obra	7,500.00 €

Total sin IVA (€uros)	7,657.48 € y 69.92 € / año
21% de IVA (€uros)	1,608.07 € y 14.68 € / año
16% Costes indirectos (€uros)	1,225.20 €
Presupuesto total	10,490.75 € y 84.60 € / año

El importe total asciende a la cantidad de:

DIEZ MIL CUATROCIENTOS NOVENTA EUROS CON SETENTA Y CINCO CENTIMOS, junto con un coste anual de **OCHENTA Y CUATRO EUROS CON SESENTA CENTIMOS**.