UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Trabajo de Máster en Ingeniería de Computadores

BOSS: Bluetooth OBDII Simple Simulator

Oscar Alvear Alvear

Tutor: Dr. Carlos Tavares Calafate

Febrero 2014

Universidad Politécnica de Valencia
Departamento de Informática de Sistemas y Computadores
Grupo de Redes de Computadores

A mi esposa y a mis padres por brindarme todo su apoyo en esta travesía, y en especial a Dios por darme las fuerzas y sabiduría para cumplir esta meta.

Agradecimientos

Doy gracias en primer lugar a Dios por ayudarme y guiarme en esta nueva etapa en mi vida; a mi esposa, Ana Gabriela, por estar junto a mi en todo momento; y a mis padres y hermanos por brindarme su apoyo incondicional.

También quiero agradecer a Carlos, mi tutor, por guiarme en la realización del trabajo, a Pietro, por permitirme trabajar dentro del Grupo y a Juan Carlos por la confianza depositada en mi; y a todos mis amigos del GRC, con los cuales he compartido inolvidables momentos.

Gracias...

Índice de figuras

1.1. Dispositivos Móviles 5
1.2. Driving Styles 6
1.3. Aplicaciones Móviles Bluetooth - OBDII 9

2.1. Arquitectura Bluetooth OBDII 11
2.2. Bus CAN 12
2.3. Interfaz OBDII 13
2.4. Interfaz Bluetooth ELM327 14

3.1. Dispositivos Móviles Actuales 17
3.2. Bluetooth Stack 19
3.3. Georeferenciación de Dispositivos Móviles 20
3.4. Sensores de dispositivos móviles. 21

4.1. Dinámica del vehículo. 23
4.2. Potencia de un motor 24
4.3. Modelado de Caja de Cambios. 26
4.4. Fuerzas Aerodimámicas 27
4.5. Fuerza Gravitacional 28
4.6. Fuerzas de Rosamiento 28
4.7. Latitud y Longitud. 30
4.8. Posibles líneas terrestres. 30

5.1. Sistema BOSS. 34
5.2. BOSS-Server 35
5.3. Simulación de movilidad de vehículo en BOSS-Server. 36
5.4. Emulación Bluetooth OBDII y GPS de BOSS. 37
5.5. Parámetros OBDII de BOSS. 38
5.6. GPSEmulator. 39
5.7. OBDIICapture. 40
5.8. Arquitectura del sistema BOSS. 41
5.9. Estructura del sistema BOSS. 42
5.10. ECU Vehicle. 43
5.11. Estructura de la Simulación del Vehículo. 43
5.12. Estructura de GPS Emulator. 47
5.13. UUID Blueooth OBDII. 49
5.14. Mock Location. 50

6.1. Pruebas de Funcionamiento 51
6.2. Conexión con Torque. 52
6.3. Transmisión de datos hacia Torque. 53
6.4. Pruebas de Georeferenciación. 53
6.5. Parámetros Personalizados 54
6.6. Pruebas con DrivingStyles. 54
6.7. Conexión hacia DrivingStyles 55
6.8. Transmisión de datos hacia DrivingStyles 55
6.9. Análisis de paquetes recibidos en vehículo KIA 57
6.10. Tiempos de respuesta para distintos tamaños de mensajes variando el tiempo entre peticiones consecutivas. 57
6.11. Comparación de los tiempos de respuesta de un coche real y del emulador variando el tiempo entre peticiones consecutivas. 58
6.12. Tasa de respuestas por segundo para distintos tamaños de mensajes variando el tiempo entre peticiones consecutivas. 59
6.13. Comparación de la tasa de respuestas por segundo de un coche real y del emulador variando el tiempo entre peticiones consecutivas. . . . 59
6.14. Función densidad de probabilidad para el tiempo de respuesta del OBD-II para distintos tamaños de mensaje. 60
6.15. Función densidad de probabilidad para el tiempo de respuesta del OBD-II compara. 61

Índice de cuadros

2.1. Comandos AT 15
2.2. Modos de operación de OBDII 16
2.3. OBDII PIDs 16

4.1. Coeficientes Aerodinámicos 27
4.2. Coeficientes de Rodadura 29

5.1. Comandos AT configurables en BOSS 47

A.2. OBDII PIDs 71

B.1. Comandos AT Generales 73
B.2. Comandos AT de parámetros programables 74
B.3. Comandos AT OBD 74
B.4. Comandos AT Específicos ISO 75
B.5. Comandos AT Específicos CAN 75
B.6. Comandos AT Específicos J1939 CAN 76

Índice general

Abstract 1

Resumen 3

1. Introducción 5
- 1.1. Introducción . . . 5
- 1.2. Motivación . . . 7
- 1.3. Objetivos . . . 8
- 1.4. Trabajos Relacionados . . . 8

2. Arquitectura Bluetooth OBDII 11
- 2.1. Bus CAN . . . 12
- 2.2. OBDII: On Board Diagnostic . . . 13
- 2.3. ELM327: OBD to RS232 Interpreter . . . 14
  - 2.3.1. AT Commands . . . 14
- 2.4. OBDII PIDs . . . 15

3. Arquitectura Dispositivos Móviles 17
- 3.1. Conexión Bluetooth . . . 18
- 3.2. Conexiones de red . . . 18
- 3.3. Georeferenciación . . . 20
- 3.4. Otros sensores . . . 21

4. Modelos de Simulación 23
- 4.1. Modelo de Vehículo . . . 23
  - 4.1.1. Fuerzas de Tracción . . . 24
  - 4.1.2. Fuerzas Aerodinámicas . . . 27
  - 4.1.3. Fuerza Gravitacional . . . 28
  - 4.1.4. Fuerza de Rozamiento . . . 28
- 4.2. Modelo de Georeferenciación . . . 29
  - 4.2.1. Rhumb Line . . . 31

5. BOSS: Bluetooth OBDII Simple Simulator 33
- 5.1. Sistema BOSS . . . 34
  - 5.1.1. BOSS-Server . . . 34
  - 5.1.2. GPSEmulator . . . 39
  - 5.1.3. OBDIICapture . . . 40

5.2. Arquitectura . . . 41
5.2.1. ECU Engine . . . 42
5.2.2. Vehicle Simulator . . . 43
5.2.3. Geoposition calculator . . . 46
5.2.4. Bluetooth OBDII Emulator . . . 46
5.2.5. GPS Manager . . . 47
5.2.6. GUI . . . 48
5.3. Consideraciones de Desarrollo . . . 48
5.3.1. Lenguajes de programación . . . 48
5.3.2. Librerías . . . 48
5.3.3. Conexión Bluetooth . . . 49
5.3.4. Mock Location . . . 49

6. Pruebas de funcionamiento **51**
6.1. Pruebas Generales . . . 51
6.1.1. Pruebas de establecimiento de enlace . . . 52
6.1.2. Pruebas de transmisión parámetros OBDII . . . 52
6.1.3. Pruebas georeferenciación . . . 53
6.1.4. Pruebas generación de parámetros personalizados . . . 53
6.2. Pruebas con Driving Styles . . . 54
6.2.1. Pruebas de establecimiento de enlace . . . 54
6.2.2. Pruebas de funcionamiento . . . 55
6.3. Análisis de conexión . . . 55
6.3.1. Pruebas multipetición . . . 56
6.3.2. Análisis de la dispersión de retardo . . . 60

7. Conclusiones **63**
7.1. Sumario y Conclusiones . . . 63
7.2. Trabajo Futuro . . . 64
7.3. Reconocimiento . . . 65

A. OBDII PIDs **67**
A.1. Descripción . . . 67
A.2. Estándar . . . 67

B. AT COMMANDS **73**
B.1. Descripción . . . 73
B.2. Comandos . . . 73

Bibliografía **77**

Nomenclatura **79**

Abstract

The majority of vehicles built during the last decade integrate an On Board Diagnostic (OBD II) interface, through which it is possible to monitor and manage multiple operational parameters. In the past few years, Bluetooth OBD II devices have been introduced in the market to facilitate connection to mobile devices. With the increased use of these devices, a lot of applications for control and monitoring of different parameters "in real time" is being developed; so, this infrastructure has opened a broad research area related to "smart driving".

The main problem in the development of these applications is the need to have a vehicle, along with the OBD II connector, for debugging and validation purposes. Furthermore, the challenge of testing different configurations in different scenarios remains.

Our proposal to address this problem is BOSS: Bluetooth OBDII Simple Simulator, which allows testing the correctness and the performance of applications on a regular computer. Another useful functionality of BOSS is the emulation of Geo-positions in Android Systems through a GPS-Emulator. By combining both functionalities, BOSS provides a complete development environment for mobile vehicular applications.

Resumen

La mayoría de los vehículos fabricados en la última década integran una interfaz de diagnóstico llamada OBDII, mediante la cual es posible monitorizar y administrar múltiples parámetros operacionales. En los últimos años, dispositivos Bluetooth OBDII se han introducido en el mercado para facilitar la conexión a dispositivos móviles. Con el incremento del uso de estos dispositivos se están desarrollando muchas aplicaciones para control y monitorización en tiempo real de diferentes parámetros de los vehículos; esta infraestructura ha abierto una amplia área de investigación relacionada con la conducción inteligente.

El mayor problema para desarrollar este tipo de aplicaciones es la necesidad de tener un vehículo, con un conector Bluetooth OBDII, para depurar y validar las aplicaciones. Además, incluso teniendo un vehículo, el problema de probar diferentes configuraciones en diferentes escenario prevalece.

Nuestra propuesta para solventar este problema es BOSS: Bluetooth OBDII Simple Simulator, el cual permite probar la exactitud y el rendimiento de las aplicaciones en un computador regular. Otra funcionalidad de BOSS es la emulación de Geoposicionamiento en Sistemas Android mediante GPS-Emulator. Combinando las dos funcionalidades, BOSS provee un completo ambiente de desarrollo para aplicaciones vehiculares móviles.

1. Introducción

1.1. Introducción

En los últimos años, se ha dado un desarrollo vertiginoso de las Tecnologías de la Información y Comunicación, y en especial de los dispositivos móviles, como smartphones y tabletas, los cuales tienen una gran capacidad de cálculo y almacenamiento, tienen varias interfaces de comunicación como WiFi, Bluetooth, NFC, GSM, WCDMA, LTE, etc., y además varios sensores como giroscopios, acelerómetros, brújulas, sensores de presión, sensores de luz ambiental, GPS, etc., y en una amplia gama de precios y características, lo que les permite acoplarse a las distintas necesidades de los usuarios, lo cual está permitiendo desarrollar una gran variedad de aplicaciones en diferentes ámbitos de la vida cotidiana, facilitando el desarrollo de la computación ubicua y móvil.

Figura 1.1.: Dispositivos Móviles

Por otro lado, los automóviles modernos incorporan una unidad de diagnostico a bordo con una interfaz de comunicación para realizar el análisis de estado de los diferentes parámetros del vehículo, llamado OBDII (On Board Diagnostic); a través de esta interfaz se puede obtener los datos del vehículo en tiempo real, además de los errores registrados en la ECU (Engine Control Unit) del vehículo que se utilizan para mantenimiento. Para recuperar los datos es necesario conectar un intérprete de OBDII mediante el cual se puede conectar a un computador si se tiene una conexión serial, o incluso a un dispositivo móvil si tiene una conexión Bluetooth. El intérprete

de OBDII es “ELM327 OBD to RS232 Interpreter”, en el cual se especifica el diseño de un microchip para la interpretación de los datos enviados por la interfaz OBDII, y brinda una interfaz de salida serial que puede ser un puerto serie o un puerto Bluetooth.

Teniendo en cuenta estas consideraciones, si se aprovecha la capacidad computacional y de comunicación de los dispositivos móviles, además de la capacidad de realizar el diagnóstico de los diferentes parámetros del vehículo en tiempo real, y se conectan a través de un intérprete OBDII Bluetooth, se crea una línea de investigación muy amplia en el ámbito de la "conducción inteligente". Las posibilidades de desarrollo e investigación de aplicaciones móviles que utilicen los datos obtenidos de la interfaz OBDII es muy amplia, desde una simple aplicación de visualización de los datos, como un velocímetro con velocidad, carga del motor, etc., hasta sistemas mucho más sofisticados como sistemas de análisis del estilo de conducción para reducir el consumo de combustible, sistemas asistentes de conducción, sistemas de confort, etc.

Figura 1.2.: Driving Styles

La aplicaciones móviles que utilizan los datos de un vehículo, obtenidos a través de la interfaz Bluetooth OBDII, para ayudar a la “conducción inteligente”, tienen básicamente una arquitectura en la cual (i) utilizan los sensores del dispositivo móvil, como el GPS, para obtener datos del entorno, (ii) obtienen los datos del vehículo a través de la interfaz Bluetooth OBDII, y (iii) aprovechan en poder de procesamiento de los dispositivos para realizar algún tipo de relación o cálculo, o enviar algún tipo de información por sus interfaces de comunicación hacia Internet u otra infraestructura. Uno de los mayores problemas que se puede observar es la necesidad de tener un vehículo con un intérprete Bluetooth OBDII conectado y moviéndose todo el tiempo para realizar las pruebas de funcionamiento, con todos los inconvenientes que conlleva como tiempo, espacio, movilidad, costos, etc., además de los inconvenientes de realizar pruebas en distintos escenarios.

Para solucionar estos problema se requeriría: (i) una forma de simular los datos y la señal generada por la interfaz Bluetooth OBDII, y (ii) una manera de simular la movilidad y sobrescribir en el dispositivo móvil la posición simulada.

Para la simulación de la señal Bluetooth OBDII existen actualmente algunas soluciones, tanto en software como en hardware, pero todas ellas son muy simples y no

tienen la suficiente flexibilidad ni tampoco la facilidad de emular diferentes escenarios; además, son mucho más costosas y diseñadas para realizar pruebas específicas con marcas de vehículos concretos. Por otra parte, para simular una ubicación en los dispositivos móviles, en sistemas Android básicamente, existen algunas aplicaciones; pero son muy sencillas en las que únicamente se introduce una ubicación y se simula la posición indicada, sin la posibilidad de introducir los datos de una fuente externa. En resumen, no existen soluciones lo suficientemente potentes y flexibles para solucionar, de una manera fácil y rápida, los problemas observados con anterioridad de una manera integral.

El Bluetooth OBDII Simple Simulator (BOSS) es nuestra propuesta para solucionar estos problemas, consta de dos partes: (i) una aplicación de escritorio desarrollada en Java (OBDII Simulator),en la cual se realiza la simulación de los datos y se genera dos enlaces de comunicación Bluetooth: una señal para emular un intérprete Bluetooth OBDII y una señal para indicar a "GPS Emulator" la posición a simular en el dispositivo móvil; (ii) una aplicación para dispositivos Android (GPS Emulator), en la cual genera un servicio para recibir a través de una conexión Bluetooth los datos de Geoposicionamiento generados en "OBDII Simulator".

1.2. Motivación

El Grupo de Redes de Computadores de la Universidad Politécnica de Valencia está involucrada desde hace años en líneas de investigación relacionadas con las redes vehiculares en proyectos como "Walkie-Talkie: Sistemas de Comunicaciones Vehiculares para el soporte de una nueva Generación de Sistemas de Transporte Inteligentes, seguros y eco-eficaces" o "ABATIS: Arquitectura de Balanceado Automático de Tráfico mediante Integración de Smartphones con vehículos" en los cuales se utilizan dispositivos móviles para analizar y procesar la información obtenida de los vehículos a través de una interfaz Bluetooth OBDII.

Sin embargo, las posibilidades y tiempo de investigación necesario para el desarrollo de nuevas propuestas y soluciones para la integración de dispositivos móviles y vehículos se ven limitados por la necesidad de tener a mano un vehículo con un intérprete Bluetooth OBDII conectado para realizar las pruebas de funcionamiento; además, las soluciones disponibles en la actualidad no cumplen las expectativas requeridas, ya sean porque son muy básicas para probar diferentes parámetros y escenarios, o porque son costosas y poco flexibles.

Debido a estos inconvenientes es necesario desarrollar un simulador flexible, configurable y fácil de utilizar, en el que se pueda realizar una gran variedad de pruebas de manera fácil y rápida, que sea compatible con dispositivos móviles, en especial sistemas Android, con los que se trabaja actualmente en el Grupo de Redes de Computadores.

1.3. Objetivos

Debido a la necesidad inmediata del Grupo de Redes de Computadores de desarrollar y probar aplicaciones dentro de las líneas de investigación de redes vehiculares y dispositivos móviles, y teniendo presente el problema que conlleva desarrollar este tipo de aplicaciones, el objetivo principal del Trabajo de Máster es analizar y desarrollar una plataforma de simulación de un vehículo con un dispositivo Bluetooth OBDII que sea configurable, intuitivo y fácil de utilizar a través de una interfaz gráfica amigable.

Para cumplir el trabajo de manera satisfactoria es necesario cumplir con los siguientes objetivos específicos:

- Simular la comunicación Bluetooth entre un intérprete Bluetooth OBDII ELM327 y un dispositivo móvil.
- Codificar y reproducir los códigos PIDs del protocolo OBDII a través de la conexión Bluetooth.
- Generar un modelo matemático que simule la variabilidad de los datos de un vehículo en el tiempo y su georeferenciación.
- Crear una interfaz gráfica amigable para la utilización del simulador.
- Realizar la emulación de geoposicionamiento en sistemas Android, acordes con los datos simulados.
- Probar el funcionamiento en aplicaciones Android - Bluetooth OBDII, en especial la aplicación DrivingStyles.

1.4. Trabajos Relacionados

En la actualidad se han creado varias aplicaciones para el control de los datos de OBDII[1], que permiten tener una lectura en tiempo real de los datos del vehículo. Las más populares son DrivingStyles[2], Torque[3], OBD Car Doctor[4], OBDAutoDoctor[5], entre otras. Todas estas aplicaciones permiten tener una lectura de los códigos OBDII[6] del vehículo, y medir su rendimiento en tiempo real. A continuación se analiza algunas de ellas:

- Driving Styles permite analizar los datos capturados para tener un control del estilo de conducción de una persona permitiendo analizar el consumo de combustible en un trayecto.
- Torque, permite leer una gran cantidad de códigos permitiendo escoger cuales mostrar en diferentes paneles, pudiendo configurarlo como un velocímetro.
- OBD Car Doctor, permite abrir una conexión Bluetooth con la interfaz Bluetooth OBDII y solicitar los datos manualmente a través de comandos.

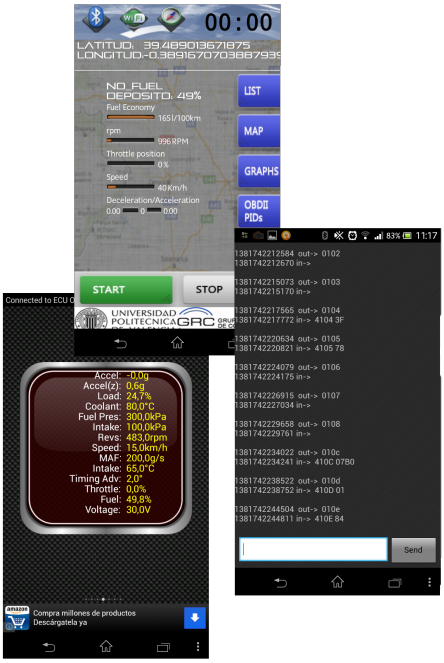

Figura 1.3.: Aplicaciones Móviles Bluetooth - OBDII

El principal obstáculo al que se enfrenta el desarrollador de este tipo de aplicaciones se ve limitado por la necesidad de tener un vehículo con un conector OBDII Bluetooth conectado para realizar las pruebas de funcionamiento.

En el mercado están disponibles algunas opciones que ayudan a simular un intérprete OBDII. Básicamente existen dos tipos de simuladores: (i) basados en hardware, como ECUsim 2000[7]; estos son costosos, ya que se necesita adquirir el hardware y el software para su funcionamiento, y tienen un número limitado de códigos que simulan, al estar orientados al entrenamiento de personal para mantenimiento un tipo de vehículo concreto, y (ii) basados en software, como OBDSim[8], RS232-OBD[9], Automotive OBDII Simulator[10], los cuales son demasiado sencillos y solamente generan unos pocos códigos, además, la mayoría no soporta la emulación de Bluetooth, sino simplemente una conexión serie, habiendo sido desarrollados básicamente para pruebas básicas de funcionamiento de OBDII.

Nuestra propuesta Bluetooth OBDII Simple Simulator, BOSS, es un simulador de un vehículo que emula un intérprete Bluetooth ELM237 (OBDII) por un lado y emula geoposicionamiento en sistemas Android por otro, cumpliendo de esta manera con los requerimientos previamente analizados. A continuación se detalla el trabajo realizado para el desarrollo de BOSS: en el Capítulo 2 se analiza la arquitectura del intérprete Bluetooth OBDII conectado en un vehículo; en el Capítulo 3 se analiza la arquitectura de las aplicaciones móviles para redes vehiculares móviles, sus sensores y la forma de conectarse al vehículo; en el Capítulo 4 se analiza los modelos de simulación, tanto de la dinámica del vehículo como el movimiento de un objeto

dentro del globo terrestre o Loxodromía; en el Capítulo 5 se analiza el sistema BOSS, la arquitectura, y algunos aspectos del desarrollo; en el Capítulo 6 se muestra las pruebas de funcionamiento realizadas de BOSS, con varias aplicaciones existentes en el mercado y un análisis de tiempos de respuesta; y por último en el Capítulo 7 las conclusiones del trabajo realizado y el trabajo futuro que se continuará realizando.

2. Arquitectura Bluetooth OBDII

La unidad de diagnóstico (OBDII[1], OnBoard Diagnostic) es el sistema por el cual los vehículos actuales permiten una comunicación entre sus componentes internos y aplicaciones externas. La unidad OBDII está conectada a través del bus CAN a la unidad de control del vehículo (ECU, Engine Control Unit), sistema de transmisión, sistema ABS, sistema de climatización, sistema de confort, entre otros, brindando una interfaz por la cual consultar los diferentes parámetros de dichos componentes. Para la lectura de datos en un computador o un smartphone es necesario conectar a la unidad OBDII un intérprete OBDII, llamado ELM327[11]. Este brinda una conexión Serie o Bluetooth por la cual se puede leer los datos del vehículo. Para la transmisión de datos es necesario establecer algunas configuraciones iniciales como la velocidad de conexión, el formato de los mensajes, etc.; esto se realiza a través de los comandos AT del intérprete ELM327. Una vez establecida la conexión con el intérprete ELM327 se solicita la información a la unidad OBDII a través de los identificadores de los parámetros llamados OBDII PIDs[6] (On-Board Diagnostics Parameter IDs) los cuales son devueltos en un formato especifico cada uno, por lo que deben ser transformados a las diferentes unidades dependiendo de cada código. El sistema funciona bajo un esquema cliente/servidor, es decir, la aplicación cliente (aplicación externa) solicita la información y el servidor (ELM327) responde a dicha solicitud; no es posible recibir información del vehículo si no se solicita primero de forma expícita .

Figura 2.1.: Arquitectura Bluetooth OBDII

2.1. Bus CAN

En la actualidad los vehículos modernos incorporan una gran cantidad de componentes electrónicos y no es factible usar conexiones punto-a-punto para cada uno de ellos, por lo cual es necesario utilizar otro medio para poder conectarlos. El medio más utilizado en la mayoría de los vehículos es el bus CAN (Controller Area Network), este bus es un enlace que atraviesa el vehículo y los distintos subsistemas se enlazan a dicho enlace permitiendo la intercomunicación entre todos los susbistemas. Por la gran complejidad asociada a los actuales vehículos, suele haber más de una red CAN interconectada, así como otros tipos de buses tales como el bus LIN (Local Interconnect Network), que es un bus mucho más simple y barato, utilizado para las operaciones no críticas del vehículo como los elevadores de ventanillas, por ejemplo. Para las operaciones críticas, como los sistemas ABS o dirección asistida, se está comenzado a utilizar buses mas potentes y seguros como es FlexRay, pero el medio más utilizado para interconectar la mayoría de componentes es el bus CAN.

Controller Area Network, CAN, fue desarrollado por BOSCH a finales de los 80 para interconectar los diferentes componentes electrónicos de los vehículos como sistema de control del motor, suspensiones activas, sistemas ABS, sistemas de control de tracción, sistemas de control de climatización, etc. Es un sistema multimaestro, es decir, cualquiera de los nodos interconectados pueden transmitir o recibir un mensaje; es un sistema sin pérdidas, por lo que no se descartan mensajes, sino que a través del formato de direccionamiento se da prioridad a alguno de los nodos en caso de conflicto; la conexión o desconexión de los nodos es automática, ya que los mensajes no van dirijidos a ningún nodo específico sino que tienen un identificador y llegan a todos los nodos y son utilizados por los nodos que los requieran , siendo descartados por los demás; puede llegar a velocidades de hasta 1Mbps, y el tamaño del mensaje es de hasta 8 bytes.

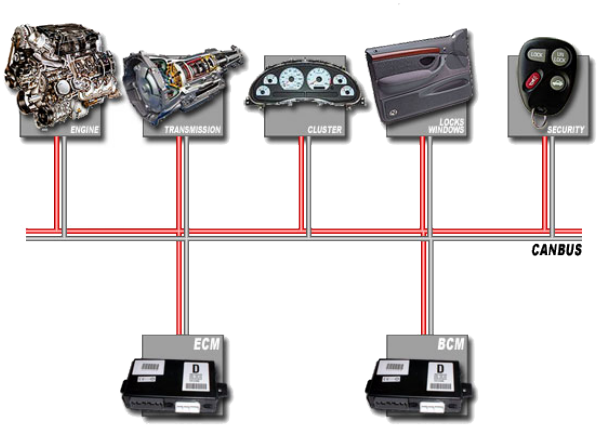

Figura 2.2.: Bus CAN

2.2. OBDII: On Board Diagnostic

La unidad de diagnóstico (OBDII OnBoard Diagnostic versión II) es el sistema por el cual se pueden realizar diagnósticos y reportes del estado de los actuales vehículos. La unidad OBDII permite acceder al estado de los subsistemas internos de los vehículos, como sistema de control del motor, sistema de transmisión, sistema ABS, sistema de climatización, sistema de confort, entre otros, permitiendo realizar varias acciones sobre estas como verificar el estado de funcionamiento, analizar el mal funcionamiento de algún subsistemas, borrar o resetear las alertar almacenadas, etc.

Los sistemas de diagnóstico se introdujeron en vehículos por primera vez en 1969 por Volkswagen. En 1991 se aprobó una ley en California que obliga a todos los vehículos comercializados en este estado a incorporar una unidad de diagnóstico (OBDI) para el control de emisiones de los vehículos. Luego esta ley se extendió a todos los Estados Unidos de Norteamérica y, desde 1996, todos los vehículos comercializados en Estados Unidos deben incorporar una unidad de diagnostico OBDII. Desde 2001, la Unión Europea exije que todos los vehículos incorporen una unidad de diagnóstico EOBD, que es la versión Europea de OBDII.

La interfaz OBDII o EOBD permiten la conexión de cinco protocolos, SAE J1850 PWM[12], SAE J1850 VPW, ISO 9141-2[13], ISO 14230 KWP2000[14] y ISO 15765 CAN[15]. La mayoría de los vehículos solo implementa uno de ellos.

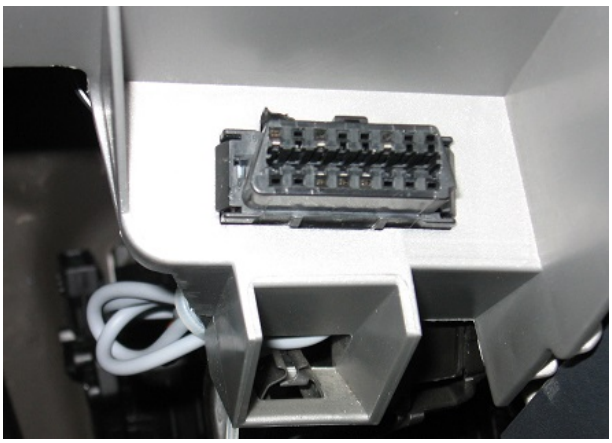

Figura 2.3.: Interfaz OBDII

Existe algunas formas de recuperar la información de OBDII como “Data Logger” que permiten almacenar la información del vehículo para luego analizarlo, pero la solución más difundida y utilizada actualmente es el intérprete ELM327 que permite leer los datos de OBDII en tiempo real.

2.3. ELM327: OBD to RS232 Interpreter

Todos los vehículos actualmente incorporan un unidad de diagnóstico, OBDII, pero no es posible leer los datos directamente de dicha unidad. Para esto es necesario un "puente" entre OBDII y una aplicación externa. ELM327 está diseñado como un enlace entre la unidad OBDII y el estándar RS232. Existen dos tipos de intérpretes: (i) el clásico, que tiene una interfaz RS232 Serie, y es utilizado para conectar a las computadoras normales, y (ii) unos intérpretes nuevos que tienen una interfaz de salida Bluetooth, y que se pueden utilizar para conectar a dispositivos móviles.

El intérprete ELM327 reconoce automáticamente 9 tipos de protocolos de OBDII, pero además se pueden configurar algunos parámetros adicionales mediante los comandos AT.

Figura 2.4.: Interfaz Bluetooth ELM327

2.3.1. AT Commands

En el intérprete ELM327 se pueden configurar algunos parámetros para modificar su comportamiento. La mayoría de los parámetros son autoconfigurables ya que el intérprete verifica el protocolo soportado por el módulo OBDII y ajusta automáticamente la velocidad de transferencia y algunos parámetros para la transmisión de datos, aunque se pueden personalizar algunas configuraciones, por ejemplo para ajustar los tiempos de espera, habilitar o deshabilitar el "echo", resetear o reiniciar al intérprete, entre otros. Todas estas configuraciones se realizan a través de los comandos AT.

El interprete ELM327, al igual que los modems, reconoce a los comandos que inicien con las letras “AT” como un comando de configuración interno, cuando ejecuta correctamente el comando devuelve la respuesta “OK” o el valor solicitado. Se debe tener en cuenta que los valores numéricos debe ser enviados o recibidos en hexadecimal. En la Tabla 2.1 se muestran algunos comandos.

Valor	Descripción
AT Z	Reinicia
AT E0/E1	echo off/on
AT L0/L1	Linefeed off/on
AT S0/S1	Space off/on
AT @1	Descripción del dispositivo
AT @2	Identificador del dispositivo
AT DP	Descripción del Protocolo
AT TP h	Intenta el protocolo h

Cuadro 2.1.: Comandos AT

Todos los comandos AT se pueden encontrar en el Apéndice B

2.4. OBDII PIDs

OBDII PIDs (OnBoard Diagnostics Parameter IDs) son los códigos de identificación de los distintos parámetros que se pueden medir en un vehículo. El estándar OBDII de la SAE J1979 (Society of Automotive Engineers) define algunos parámetros, pero los fabricantes de vehículos pueden e introducen sus propios códigos.

Existen algunos modos de operación en el estándar para acceder a la información de OBDII, los cuales están descritos en la Tabla 2.2.

El estándar SAE J1979 define varios OBDII PIDs en los distintos modos de operación, pero nos centraremos en el Modo 01, porque en él se pueden encontrar los datos actuales del vehículo. Los valores numéricos son enviados y recibidos en hexadecimal. La Tabla 2.3 muestra algunos códigos con su descripción y los parámetros necesarios para interpretarlos.

La columna “PID” indica el identificador del código, la columna “Tam.” indica el número de bytes del código, la columna “Descripción” indica la descripción del código, las columnas “min” y “max” indican los valores mínimos y máximos respectivamente, la columna “unidades” indica las unidades en las que se lee el código, y la columna “Fórmula” indica la fórmula para interpretar el valor recibido.

Una lista de la mayoría de OBDII PIDs, se puede encontrar en el Apéndice A

En la comunicación con el intérprete se solicitan los códigos indicando el modo y el código, por ejemplo para solicitar la velocidad del vehículo el identificador es “010D”

Modo	Descripción
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

Cuadro 2.2.: Modos de operación de OBDII

PID	Tam.	Descripción	min	max	Unidades	Fórmula
00	4	PIDs soportados				
05	1	Engine coolant temperature	-40	215	°C	A-40
0A	1	Fuel pressure	0	765	kPa (gauge)	A*3
0C	2	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
0D	1	Vehicle speed	0	255	km/h	A

Cuadro 2.3.: OBDII PIDs

en donde “01” indica el modo y “0D” el parámetro, y la respuesta sería “410D XX”; en la respuesta se cambia el primer dígito del identificador de “0” a “4”, para indicar que es la respuesta y “XX” es el valor del parámetro solicitado.

En la columna “Fórmula” la letra A representa el primer byte, la letra B el segundo byte, la letra C el tercer byte y así sucesivamente.

El código “00” tiene un tamaño de 4 bytes, es decir 32 bits, e indica los cuales de los próximos 32 códigos son soportados. El último código de este grupo indica los siguientes 32 códigos soportados y así sucesivamente.

3. Arquitectura Dispositivos Móviles

Las Tecnologías de la Información y Comunicación han tenido un desarrollo vertiginoso en los últimos años, en especial los dispositivos móviles, los cuales han llegado a ser un complemento casi imprescindible en las actividades cotidianas. El gran poder de cálculo y de almacenamiento que poseen, junto con la todas las interfaces de comunicación tales como WiFi, Bluetooth, NFC, GSM, WCDMA, LTE, etc., y elementos de sensorización como acelerómetros, giroscopios, brújulas, sensores de proximidad, sensores de luz ambiental, sensores de temperatura, GPS, cámaras, micrófonos, etc. que incorporan, posibilita el desarrollo de toda clase de aplicaciones en el ámbito de la computación ubicua para el soporte a ciudades inteligentes (Smart Cities).

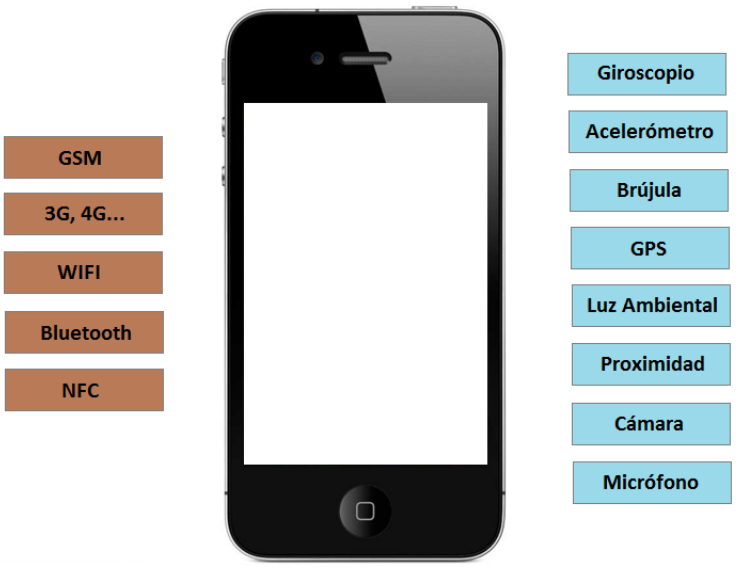

Figura 3.1.: Dispositivos Móviles Actuales

El concepto de ciudad inteligente involucra interaccionar con los elementos del entorno a través de algún dispositivo para facilitar las actividades diarias. Dentro de las ciudades inteligentes se derivan muchas subáreas como hogar inteligente, empresa inteligente, etc. En este contexto, una subárea que está en auge es la "conducción inteligente" con las aplicaciones de redes vehiculares móviles. Las aplicaciones de redes vehiculares móviles se refieren a las aplicaciones donde interactúan los conductores, los vehículos, la infraestructura de carretera y el entorno.

La forma más fácil y barata de interacción entre conductores, vehículos, infraestructura y entorno, son los dispositivos móviles. Éstos pueden interactuar con el conductor, capturar los datos del vehículo a través de una interfaz Bluetooth OBDII, capturar los datos del entorno a través de sus sensores, y comunicarse con la infraestructura de carretera mediante las interfaces de comunicación.

Para una aplicación de redes vehiculares móviles es necesario básicamente la lectura de datos de OBDII mediante un enlace Bluetooth, la conexión del dispositivo hacia el exterior mediante un enlace de red, y el análisis del entorno, como la ubicación por ejemplo, mediante sus sensores.

3.1. Conexión Bluetooth

La forma más fácil de conseguir que los dispositivos móviles comuniquen con un vehículo es a través de un conector OBDII con soporte a Bluetooth. Mediante este enlace inalámbrico entre móvil y vehículo se pueden recuperar datos del vehículo tales como velocidad, consumo, estado general, etc., para luego procesarlos y realizar alguna acción.

La tecnología Bluetooth fue creada con el objetivo de crear redes inalámbricas personales (WPAN, Wireless Personal Area Network), para facilitar la interconexión de dispositivos en cortas distancias en la banda de radiofrecuencia ISM de 2.4 GHz. Actualmente se encuentra en la versión 4 del estándar, y todas las versiones están diseñadas para soportar a las anteriores. Tiene una pila de protocolos en los que se definen el enlace físico hasta la capa de aplicaciones o perfiles. Para interconectar dos dispositivos Bluetooth es necesario implementar alguno de los perfiles definidos.

Los perfiles más conocidos son: OBEX, para la transferencia a archivos; Audio, para enviar la salida de audio de algún dispositivo; y comunicación serial, para crear un enlace entre dos dispositivos.

Los dispositivos Bluetooth pueden conectarse en varias formas, pero la más común es formar una piconet, donde un dispositivo actúa como maestro y los demás se conectan como esclavos. El maestro dirige la comunicación y los tiempos de transferencia de cada esclavo. Un dispositivo Bluetooth puede conectarse con hasta 7 dispositivos activos en simultáneo.

3.2. Conexiones de red

Los dispositivos móviles tienen varias interfaces de comunicación para el intercambio de información entre distintas infraestructuras de red, incluyendo Internet, redes privadas, redes adhoc, etc. Estas interfaces permiten crear distintos tipos de aplicaciones, tales como, sistemas cliente-servidor, sistemas distribuidos, sistemas Cloud, etc. Las interfaces de comunicación más comunes en los dispositivos móviles son:

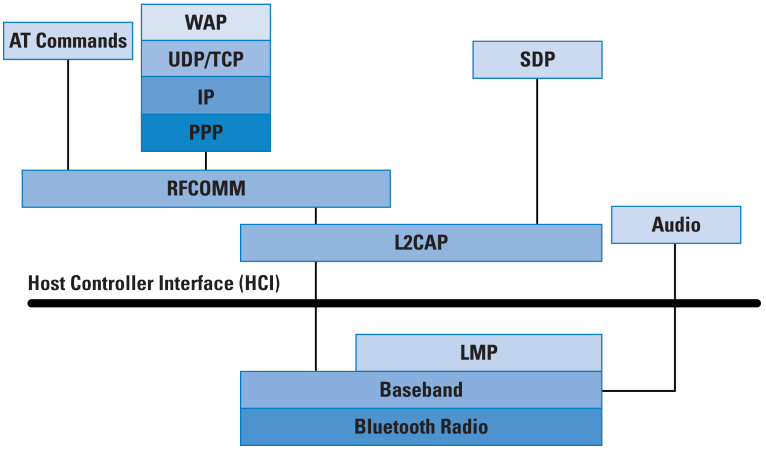

Figura 3.2.: Bluetooth Stack

WiFi (Wireless Fidelity) Es un mecanismo de comunicación inalámbrica en el que los distintos dispositivos se conectan a un punto de acceso que controla la comunicación entre ellos. WiFi es la marca del estándar 802.11. Dicho estándar especifica diferentes variantes del protocolo, donde los más usados son 802.11a, 802.11a, 802.11b, 802.11g y 802.11n. El espectro de comunicación incluye la banda de 2.4 GHz y la banda de 5 GHz pudiendo alcanzar hasta los 100 metros de distancia en campo abierto, y a velocidades de transmisión de hasta 300Mbps en los protocolos más recientes.

Redes Celulares Las Redes Celulares son redes inalámbricas formadas por celdas contiguas, cubriendo una área determinada. Cada celda tiene su transmisor o estación base que recibe la comunicación de varios emisores o dispositivos móviles. Los dispositivos móviles reciben la señal del transmisor más cercano dependiendo de la celda en que se encuentren, y el cambio de celdas o transmisores es transparente. Existen varias tecnologías, siendo las más conocidas: (i) GSM o 2G, que permiten la comunicación celular de voz, (ii) 3G, que permite la comunicación de voz y datos, tiene mayor velocidad que 2G, y es la más usada actualmente, (iii) LTE o 4G, que es una red totalmente basada en IP y permite la comunicación a altas velocidades, pudiendo llegar a 1 Gbps en reposo, y a 100 Mbps en movimiento.

NFC (Near Field Communication) Es el mecanismo de comunicación inalámbrico a corto alcance para el intercambio de datos entre dispositivos. Funciona mediante la inducción de un campo magnético en antenas de espira. Trabaja a una frecuencia de 13,56 MHz y tiene un alcance de unos 10 cm.

3.3. Georeferenciación

En los actuales dispositivos móviles se puede determinar su ubicación por medio de la georeferenciación, es decir, determinar su latitud y su longitud dentro del globo terrestre. Dentro de los dispositivos móviles la georeferenciación se puede realizar por varios métodos:

GPS Los dispositivos móviles incorporan un sensor GPS (Geographic Position System) que permite establecer la ubicación mediante la triangulación con un sistema de satélites de navegación. Estos satélites están ubicados en el espacio, alrededor de la tierra, de tal forma que en cualquier punto de la tierra se puede tener contacto con al menos 4 de ellos en todo momento.

Redes Celulares Otra forma de determinar la ubicación de un dispositivo móvil es mediante la triangulación con las estaciones celulares base más cercanas. Este sistema no es muy exacto, pero permite determinar en qué celda está un dispositivo.

WiFi También se puede determinar la ubicación de un dispositivo móvil mediante las redes WiFi que tiene a su alcance. Una posibilidad es almacenar las direcciones MAC de los puntos de acceso y su ubicación; existen varias empresas que han realizado esta labor, entre ellas Google, permitiendo así determinar la posición del terminal. Otra alternativa es conectarse a una red WiFi y utilizar la dirección IP pública de dicha red para ubicar el terminal.

Figura 3.3.: Georeferenciación de Dispositivos Móviles

La forma más exacta de determinar la posición, y la más usada por los dispositivos móviles, es una combinación de las tres.

3.4. Otros sensores

Los dispositivos móviles tiene varios sensores internos tales como acelerómetros, giroscopios, sensores de luminosidad, etc. Estos permiten capturar los datos del entorno y procesarlos dentro de una aplicación.

Los principales sensores que incorporan la mayoría de dispositivos móviles son:

Acelerómetro Permite medir la aceleración lineal que soporta un dispositivo en los 3 ejes "x", "y" y "z".

Giroscopio Permite medir la orientación y su velocidad de cambio en un dispositivo, es decir la velocidad angular.

Brújula Permite determinar la orientación de un dispositivo.

Luminosidad Permite determinar la cantidad de luz ambiental.

Proximidad Permite determinar si existe un objeto cercano al dispositivo.

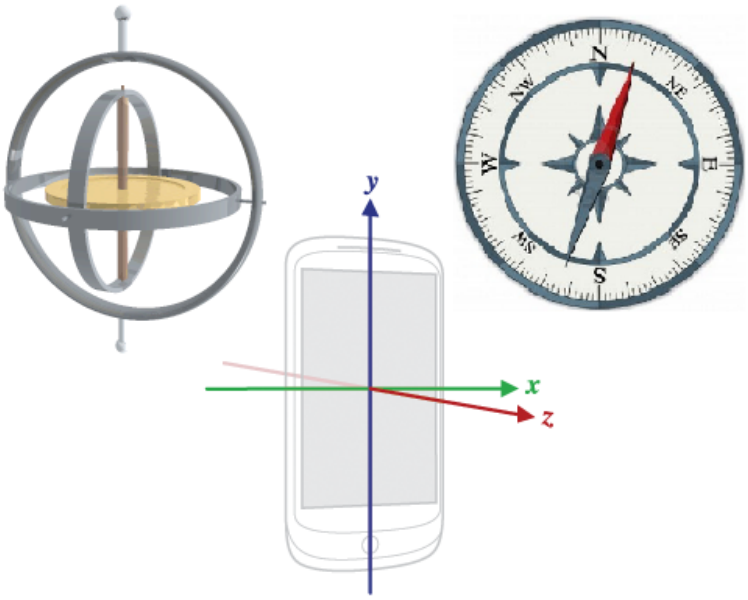

Figura 3.4.: Sensores de dispositivos móviles.

4. Modelos de Simulación

BOSS (Bluetooth OBDII Simple Simulator) es un simulador/emulador de un vehículo con un intérprete Bluetooth OBDII conectado, por lo que es importante conocer: (i) el modelo matemático de la dinámica del vehículo, (ii) el modelo matemático del movimiento de un objeto dentro del globo terrestre, y (iii) la forma de referenciar su posición geográfica.

4.1. Modelo de Vehículo

En el estudio del movimiento de un vehículo[16] se tiene que analizar todos los parámetros que influyen en el mismo. La dinámica de un vehículo de ve afectada por interacción de las fuerzas que le rodean: fuerza de tracción, fuerza aerodinámica, gravedad terrestre y fuerza de rozamiento de la carretera.

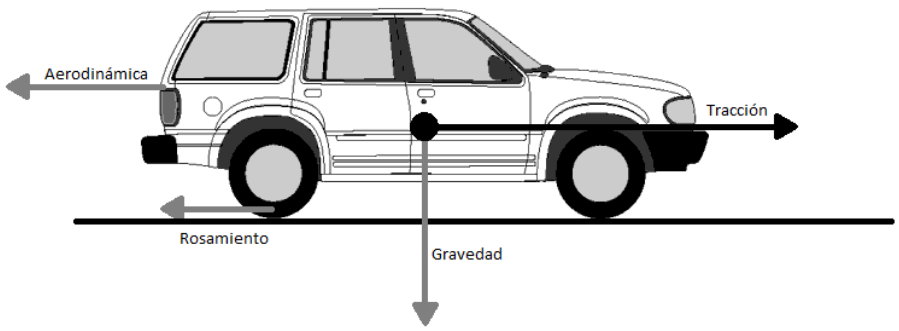

Figura 4.1.: Dinámica del vehículo.

Según las leyes básicas de la física, para mover un objeto es necesario ejercer fuerza sobre este. Dentro de la dinámica de un vehículo esta fuerza (F_t) es el producto de la interacción de otras como la fuerza generada por el motor o fuerza de tracción (F_e), la fuerza de fricción con el aire o fuerza aerodinámica (F_a), la gravedad o fuerza de gravedad (F_g), y las pérdidas por fricción o rozamiento de los neumáticos en la carretera o fuerza de rozamiento (F_r).

$$F_t{=}F_e - F_a - F_g - F_r \tag{4.1}$$

Una vez se conoce la fuerza total que define el movimiento de un vehículo (F_t), y siguiendo la ecuación de la fuerza, tenemos que la aceleración lineal (a_x) para ese vehículo:

$$a_x{=}\frac{F_t}{m} \tag{4.2}$$

Donde m es la masa del vehículo.

4.1.1. Fuerzas de Tracción

Siguiendo el análisis de [16], el movimiento o aceleración de un vehículo depende de (i) la potencia generada por el motor, y (ii) el torque o par motor ejercido sobre el sistema de transmisión, el cual transmite el movimiento a las ruedas.

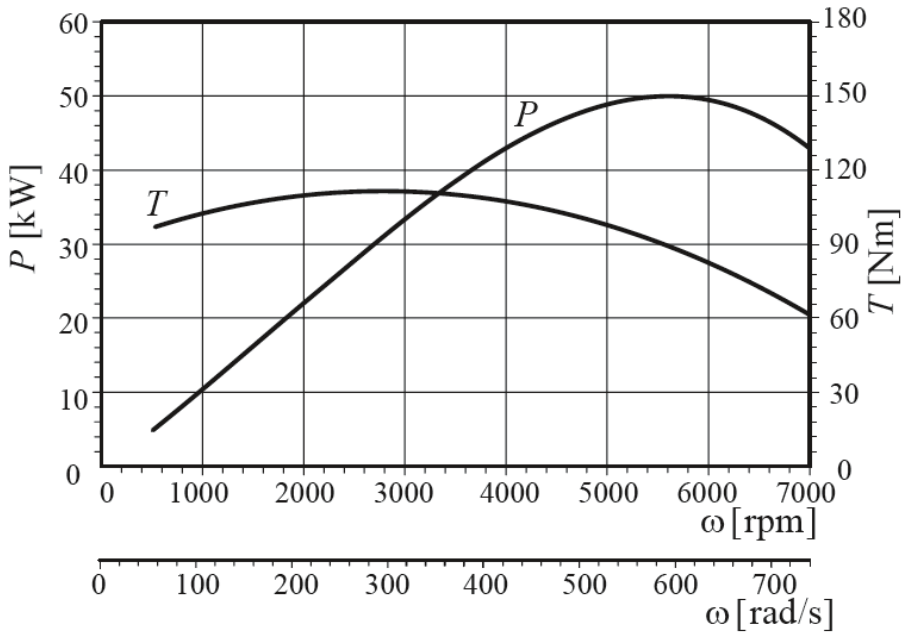

Figura 4.2.: Potencia de un motor

La potencia generada del motor se transfiere hacia los neumáticos a través del torque o par motor que ejerce sobre el sistema de transmisión, pasando por el sistema

de embrague, caja de cambios, sistema diferencial en las cuales este movimiento o velocidad angular es modificada por estos elementos.

$$F_e = \eta_j \cdot \frac{P_e}{V_x} \tag{4.3}$$

Siendo η_j el modificador de torque en un cambio j, P_e la potencia generada por el motor y v_x la velocidad del vehículo.

4.1.1.1. Potencia del Motor

El motor es el que genera la potencia necesaria para producir una aceleración y el movimiento del vehículo. Existen varios tipos de motores, como motores de combustion interna, motores a diesel, motores eléctricos, motores híbridos, etc., los cuales tienen diferentes tipos de comportamiento. Normalmente la ecuación de potencia de un motor es determinada de forma experimental. Sin embargo, en los motores de combustión interna a gasolina, se puede representar por la función P_e=$P_e(W_e)$ que se representa por el polinomio:

$$P_e = \sum_{i=1}^{3} P_i \cdot W_e^i \tag{4.4}$$

En la cual los términos individuales son: :

$$P_1 = \frac{P_m}{W_m} \tag{4.5}$$

$$P_2 = \frac{P_m}{W_m^2} \tag{4.6}$$

$$P_3 = -\frac{P_m}{W_m^3} \tag{4.7}$$

donde W_e es la velocidad angular producida por el motor, W_m la velocidad angular máxima y P_m la potencia máxima que puede producir el motor.

4.1.1.2. Sistema de Transmisión

La potencia generada por el motor ejerce un momento de fuerza llamado *torque* o *par motor* sobre el sistema de transmisión, el cual transfiere el movimiento hacia los neúmaticos, produciendo el desplazamiento del vehículo.

El sistema de transmisión está compuesto por el sistema de embrague, caja de cambios, ejes de transmisión, diferencial, y ruedas. Al transferirse el movimiento por estos elementos se modifica la velocidad angular producida por el motor, en una relación de cambio de η_j veces, que produce una velocidad de movimiento lineal:

$$v_x = \eta_j \cdot R_w \cdot W_e \tag{4.8}$$

En esta ecuación v_x es la velocidad del vehículo, W_e la velocidad angular producida por el motor, η_j la relación de cambio entre la velocidad angular producida por el motor y la transferida hacia las ruedas, y R_w es el radio de los neumáticos.

La relación entre la velocidad angular producida por el motor y la transferida (η_j) hacia las ruedas depende del radio de cada marcha (n_j) y el radio de los componentes del sistemas de transmisión (n_d).

$$\eta_j = \frac{1}{n_d \cdot n_j} \tag{4.9}$$

Los valores n_j y n_d dependen del fabricante.

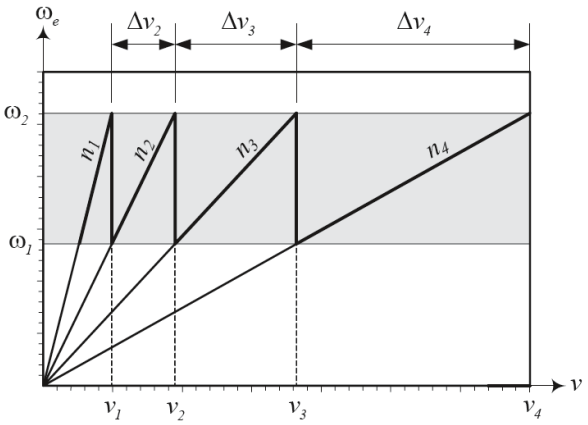

Figura 4.3.: Modelado de Caja de Cambios.

Para el cambio de marchas se especifican dos umbrales de velocidad angular del motor (RPM), W_{min} y W_{max}. Cuando se llega al umbral máximo W_{max} se cambia a una marcha superior, y cuando se llega W_{min} se cambia a una marcha inferior si es posible.

4.1.2. Fuerzas Aerodinámicas

La fuerza aerodinámica es la fuerza de friccion que se genera al penetrar un objeto dentro de un fluido gaseoso. Ésta fuerza depende del tipo de fluido en el cual penetra el objeto. La fuerza de fricción aerodinámica (F_a) que sufre un vehículo al penetrar el aire viene dado por la fórmula:

$$F_a = \frac{1}{2} \cdot C_a \cdot \rho \cdot S \cdot V_x^2 \tag{4.10}$$

Donde, S es la superficie frontal del vehículo, C_a el coeficiente aerodinámico, ρ es la densidad del aire, y V_x la velocidad del viento (o del vehículo si se considera que no existe viento).

El coeficiente aerodinámico C_a representa la resistencia que genera la forma de un objeto al penetrar en el aire. En la mayoría de vehículos está entre 0.28 y 0.45, siendo muy dificil bajar de 0.28. Los vehículos de Fórmula 1 tienen un coeficiente entre 0.7 y 1.1 dependiendo del circuito, y lo utilizan para lograr más aderencia a la pista.

Vehículo	**Coeficiente**
Todo Terreno	0.35-0.45
Común	0.28-0.35
Prototipos	> 0.28
Fórmula 1	0.7-1.1

Cuadro 4.1.: Coeficientes Aerodinámicos

Según la fórmula de la fuerza aerodinámica, se puede observar que la resistencia del aire sube de manera cuadrática con la velocidad del vehículo, experimentando más resistencia a mayor velocidad.

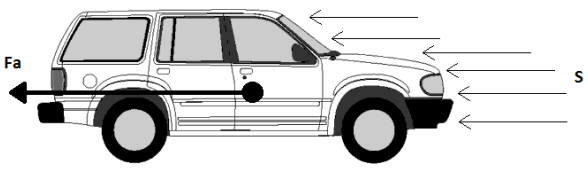

Figura 4.4.: Fuerzas Aerodimámicas

4.1.3. Fuerza Gravitacional

Todos los objetos sobre la tierra soportan una fuerza que los empuja hacia el centro del globo terrestre; esta fuerza es conocida como gravedad. Un vehículo en movimiento también soporta esta fuerza, la cual viene dada por la fórmula:

$$F_g{=}m \cdot g \cdot \sin(\theta) \tag{4.11}$$

Donde m es la masa o peso del vehículo, g la constante de gravedad, y θ el ángulo de inclinación del vehículo.

La constante de gravedad g varía dependendiendo de la ubicación dentro del globo terrestre, y en el ecuador de la tierra es de $9{,}8m/s^2$.

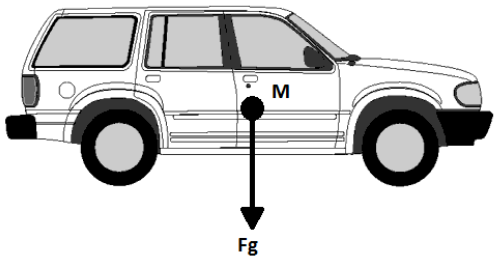

Figura 4.5.: Fuerza Gravitacional

4.1.4. Fuerza de Rozamiento

La fuerza de rozamiento es la fuerza de fricción que se genera por el contacto entre los neumáticos del vehículo con el terreno. Esta fuerza depende de: (i) el tipo de terreno sobre el que se desplaza el vehículo, (ii) el tipo de neumático y (iii) la presión de los neumáticos.

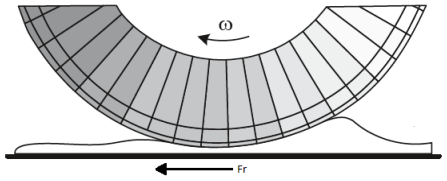

Figura 4.6.: Fuerzas de Rosamiento

Normalmente esta fuerza tiene valores muy pequeños en comparación con las otras fuerzas que dirigen la dinámica del vehículo. La fórmula para el cálculo de la fuerza de rozamiento es:

$$F_r{=}C_r \cdot m \cdot g \cdot \cos(\theta) \tag{4.12}$$

Donde C_r es el coeficente de rodadura, m la masa o peso del vehículo, g la gravedad, y θ el ángulo de inclinación del vehículo.

El coeficiente de rodadura C_r está predefinido según el tipo de terreno o superficie, tal y como se indica en la Tabla 4.2.

Superficie	**Coeficiente**
Asfalto y concreto	0.8-0.9
Asfalto mojado	0.5-0.6
Concreto mojado	0.8
Camino de tierra	0.68
Camino de tierra mojado	0.55
Grava	0.6
Nieve	0.2
Hielo	0.1

Cuadro 4.2.: Coeficientes de Rodadura

4.2. Modelo de Georeferenciación

La georeferenciación es el método por el cual se puede especificar de manera exacta un lugar en el grobo terrestre. Para este propósito se utilizan algunos parámetros: (i) líneas imaginarias que atraviezan el globo terrestre de manera horizontal, llamadas trópicos, en un rango de -90° 0 ’ 0.00” a 90° 0 ’ 0.00 ”, siendo 0° el ecuador, y 90° y -90° los polos norte y sur, respectivamente; (ii) líneas imaginarias que atraviezan el globo terrestre de manera vertical, llamadas meridianos, en un rango de -179° 59’ 59.99” a 180° 0’ 0.00”, siendo el meridiano 0°, o meridiano de Greenwich, la linea que atravieza el observatorio de Greenwich cerca de Londres, los meridianos positivos hacie el Este de Greenwich y los meridianos negativos hacia el Oeste de Greenwich; y (iii) la altitud con respecto al nivel del mar.

Los sistemas de georeferenciación actuales como los sistemas de GPS (Geographic Position System), utilizan algunos parámetros para especificar una ubicación dentro de la superfice de la tierra:

Latitud Especifica el desplazamiento vertical con respecto al ecuador.

Longitud Especifica el delplazamiento horizontal con respecto al meridiano de Greenwich.

Bearing es el ángulo de inclinación con respecto al meridiano que se encuntre.

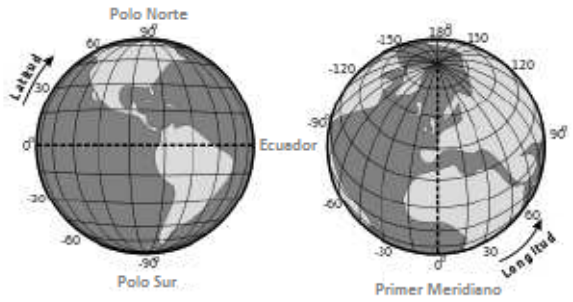

Figura 4.7.: Latitud y Longitud.

Altitud la altitud con respecto al nivel del mar.

Velocidad la velocidad de movimiento.

La superficie terrestre no es un círculo perfecto, sino que es achatada en los polos. Por este motivo, el movimiento o las líneas que pueden trazarse entre dos puntos en la superficie de la tierra son varios: (i) la línea Loxodrómica, (ii) la línea Ortodrómica y (iii) la línea Isoazimutal.

Loxodrómica Es la línea o curva que une dos puntos con un mismo ángulo de inclinación en todos los meridianos que atravieza.

Ortodrómica Es la línea más corta entre dos puntos.

Isoazimutal Es línea cuyo ángulo de inclinación o azimut con respecto al origen es siempre el mismo.

Las líneas o curvas Ortodrómica y Lodrodrómica son practicamente las mismas en distancias pequeñas. Para el análisis del movimiento de un objeto sobre la superficie de la tierra con una dirección o "bearing" específico se utiliza al análisis loxodrómico o "Rhumb Line".

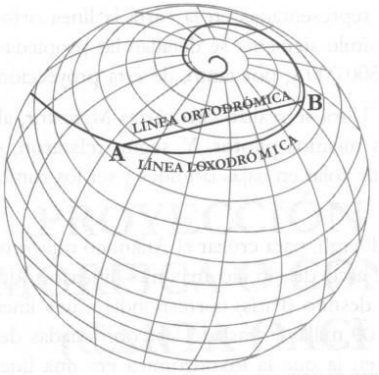

Figura 4.8.: Posibles líneas terrestres.

4.2.1. Rhumb Line

El analisis de la ruta entre dos puntos se llama “Rhumb Line”[17], y es analizada a través de trigonometría esférica y elisoidal, habiendo varias propuestas como “Vicent Formule” o “Haversin Formule” para calcular la distancia entre dos puntos.

El método para calcular un punto final (φ_2, λ_2), teniendo como entrada el punto inicial (φ_1, λ_1), la inclinación o bearing (θ) y la distancia de desplazamiento (d) con R como radio de la tierra, es:

Algoritmo 4.1 Cálculo de un punto destino.

$$\alpha = d/R$$
$$\varphi_2 = \varphi_1 + \alpha . \cos(\theta)$$
$$\Delta\varphi' = \ln\left(\tan\left(\frac{\pi}{4} + \frac{\varphi_2}{2}\right) / \tan\left(\frac{\pi}{4} + \frac{\varphi_1}{2}\right)\right)$$
$$q = \begin{cases} \cos(\varphi_1) & \overrightarrow{EW} \\ \frac{\Delta\varphi}{\Delta\varphi'} & \overrightarrow{WE} \end{cases}$$
$$\Delta\lambda = \alpha . \sin(\theta)/q$$
$$\lambda_2 = (\lambda_1 + \Delta\lambda + \pi)\, mod(2\pi) - \pi$$

5. BOSS: Bluetooth OBDII Simple Simulator

Bluetooth OBDII Simple Simulator (BOSS) es la solución desarrollada para simular un vehículo con una interfaz Bluetooth OBDII. BOSS permite generar los parámetros de un vehículo en movimiento, controlar el vehículo dentro de un mapa a través de un joystick, configurar nuevos parámetros y su método de cálculo, así como controlar la comunicación Bluetooth.

La plataforma desarrollada está formada por tres aplicaciones:

BOSS-Server Es la aplicación principal desarrollada en Java. En esta aplicación se simula los parámetros de un vehículo y su localización, y se emula la señal Bluetooth de la interfaz OBDII.

GPSEmulator Es la aplicación que sirve para emular una localización en sistemas Android. En esta aplicación se crea un servicio donde se capturan los datos de geoposicionamiento simulado en BOSS a través de una conexión Bluetooth y se emula la localización capturada.

OBDIICapture Es una aplicación Android en la que se capturan los datos de una interfaz Bluetooth OBDII. Esta aplicación se utilizó para analizar la conexión Bluetooth hacia una interfaz OBDII real, y luego para realizar las pruebas de conexión de OBDII entre el Emulator y el Simulator.

Combinando las tres aplicaciones anteriores disponemos de un sistema completo de pruebas para aplicaciones móviles vehiculares.

BOSS tiene las siguientes características:

- Multiplataforma: Está desarrollado en el lenguaje de programación Java junto con la librería Bluecove[18] para la conexión Bluetooth, y la librería JMapViewer[19] para el visor de mapas, lo que lo convierte en un sistema multiplataforma que se ha probado tanto en Windows como en Linux.
- Interfaz gráfica amigable: Tiene una interfaz gráfica amigable e intuitiva, desarrollada con la biblioteca gráfica Swing de Java donde simplemente se "enciende" el vehículo y se controla su ruta a través de un joystick en un mapa de OpenStreetMap[20].
- Flexible: Se puede agregar o eliminar los diferentes parámetros a simular, así como especificar la fórmula para el cálculo de su valor, referenciando en dicha fórmula a otros parámetros ingresados.

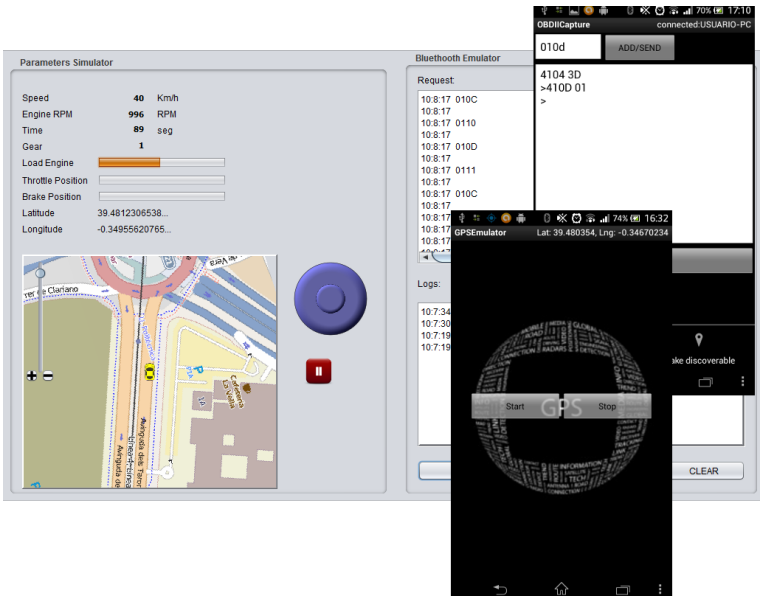

Figura 5.1.: Sistema BOSS.

- Simulación de Georeferenciación: Mediante una aplicación adicional que ha sido desarrollada, GPSEmulator, se puede emular el geoposicionamiento en cualquier sistema Android en base a los datos generados por BOSS, en tiempo real.
- Control de la comunicación Bluetooth - OBDII: se puede observar en tiempo real toda la comunicación entre el simulador y el dispositivo móvil, y todos los eventos generados en la misma. Esto es útil para analizar el correcto funcionamiento de las aplicaciones.

5.1. Sistema BOSS

El sistema BOSS está compuesto por un conjunto de aplicaciones (i) BOSS-Server, (ii) GPSEmulator (iii) OBDIICapture, que forman un sistema completo para pruebas de funcionamiento de aplicaciones móviles vehiculares. Cada una de estas aplicaciones tienen su función dentro del sistema para brindar una solución integral.

5.1.1. BOSS-Server

BOSS-Server es la aplicación principal de sistema, y se encarga de generar los datos del vehículo y transmitirlos mediante una interfaz Bluetooth OBDII emulada en el computador. Está desarrollada en el lenguaje de programación Java de manera que es multiplataforma y fácil de instalar en diferentes sistemas operativos como Windows o Linux.

Esta aplicación consta, principalmente, de dos componentes: (i) el componente encargado de la simulación de los parámetros de un vehículo, y (ii) el componente de conexión Bluetooth y emulación de una señal OBDII. Además de estos dos componentes, está también el componente encargado de la configuración de parámetros adicionales a simular.

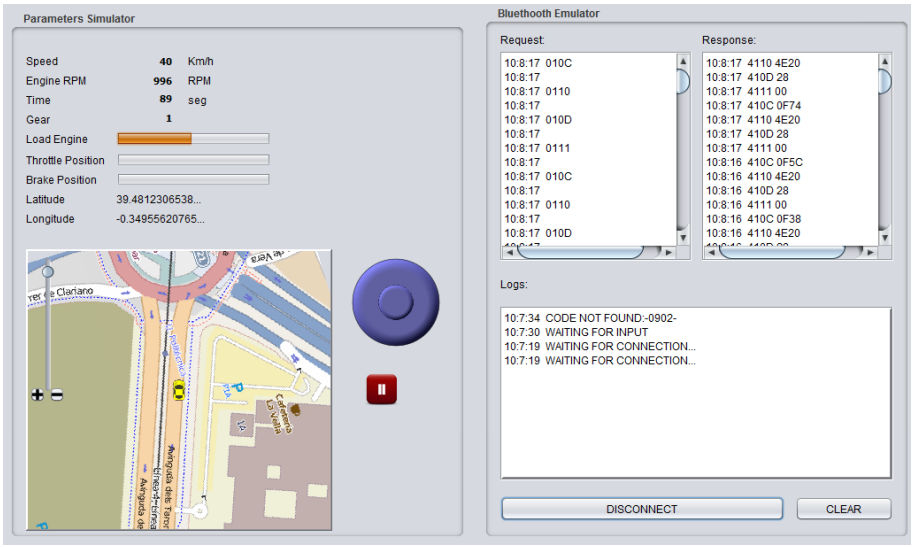

Figura 5.2.: BOSS-Server

5.1.1.1. Simulador de Vehículo

El Simulador de Vehículo es el componente que se encarga de controlar la dinámica del vehículo y visualizar los parámetros básicos simulados.

Los parámetros básicos de simulación son:

- Speed.- Indica la velocidad del vehículo.
- Engine RPM.- Indica la velocidad del motor en Revoluciones por minuto.
- Time.- Indica el tiempo desde que se inició la simulación.
- Gear.- Indica la marcha en la que se está desplazando el vehículo.
- Load Engine.- Indica el porcentaje de carga del motor.
- Throttle Position.- Indica la posición del acelerador.
- Brake Position.- Indica la posición del freno.
- Latitude.- Indica la latitud donde se encuentra el vehículo.
- Longitude.- Indica la longitud donde se encuentra el vehículo.
- Bearing.- Indica la dirección del vehículo.

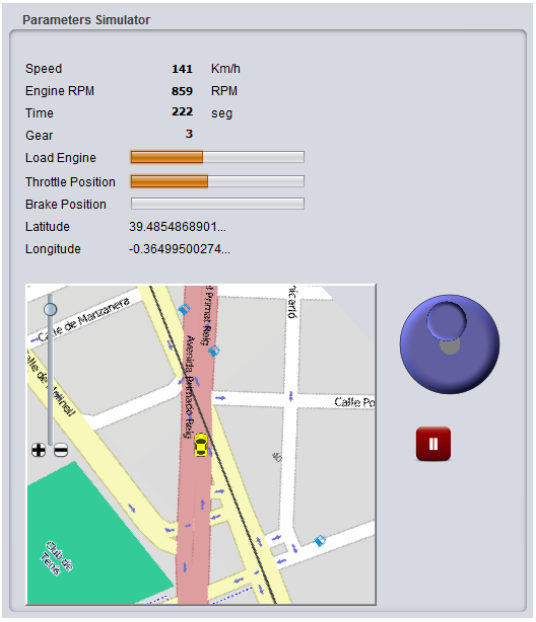

Figura 5.3.: Simulación de movilidad de vehículo en BOSS-Server.

Este componente es el encargado de simular la dinámica de un vehículo y para ejecutar una simulación se requiere realizar 4 pasos:

1. A través del mapa se establece la ubicación inicial del vehículo.
2. Se arranca el vehículo mediante el botón de encendido.
3. Se controla el vehículo libremente en el mapa mediante el joystick desde el lugar establecido en el paso 1.
4. Para finalizar la simulación se apaga el vehículo en el botón correspondiente.

El control la velocidad del vehículo se realiza a través del joystick digital incorporado en el componente, para acelerar se debe mover el joystick hacia adelante y para frenar se debe mover hacia atrás. La dirección del vehículo también se controla mediante el joystick, con los movimientos hacia izquierda y derecha.

5.1.1.2. Emulador Bluetooth

Esta sección es la encargada de controlar y monitorizar la emulación OBDII Bluetooth y la conexión Bluetooth para la peticiones de georeferenciación. Aquí se crea una conexión Bluetooth hacia las aplicaciones móviles que requieran los parámetros OBDII y una conexión Bluetooth hacia GPSEmulator para la emulación de GPS. Se pueden monitorizar todos los datos que se envían a través de los canales Bluetooth creados.

Esta sección está dividida en tres subsecciones:

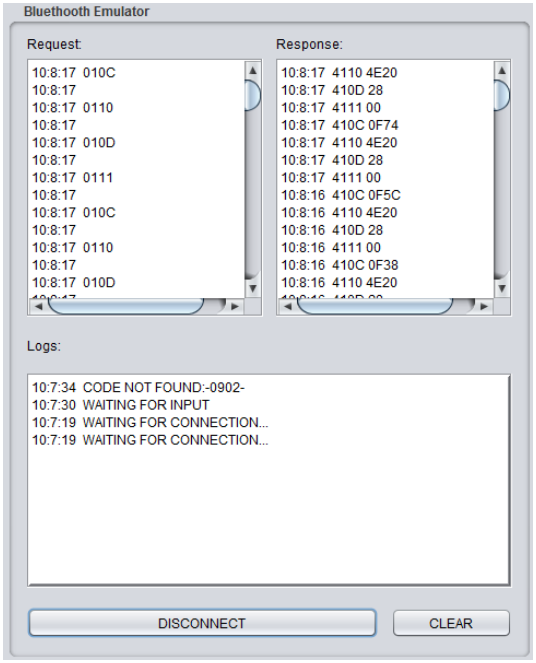

Figura 5.4.: Emulación Bluetooth OBDII y GPS de BOSS.

- Request.- Muestra las peticiones que llegan al sistema mediante los canales Bluetooth.
- Response.- Muestra las respuestas del sistema, ya sean estas OBDII o GPS.
- Logs.- Muestra los eventos importantes de las conexiones, tales como el estado de la conexión, los códigos no encontrados, etc.

Para controlar las conexiones Bluetooth se tiene dos botones:

- CONNECT/DISCONENCT.- mediante este botón se inician o terminan las conexiones Bluetooth.
- CLEAR.- Elimina los datos y logs que se muestran de las diferentes subsecciones (Request, Response y Logs).

5.1.1.3. Parámetros

En la ventana de Parámetros OBDII PIDs se puede manipular los parámetros adicionales que se simulan en BOSS.

Se tiene un listado de todos los parámetros que se están simulando, y se puede modificar en tiempo de ejecución los valores y fórmulas de cálculo de cada uno de ellos.

Cada parámetro tiene varios atributos que se pueden modificar:

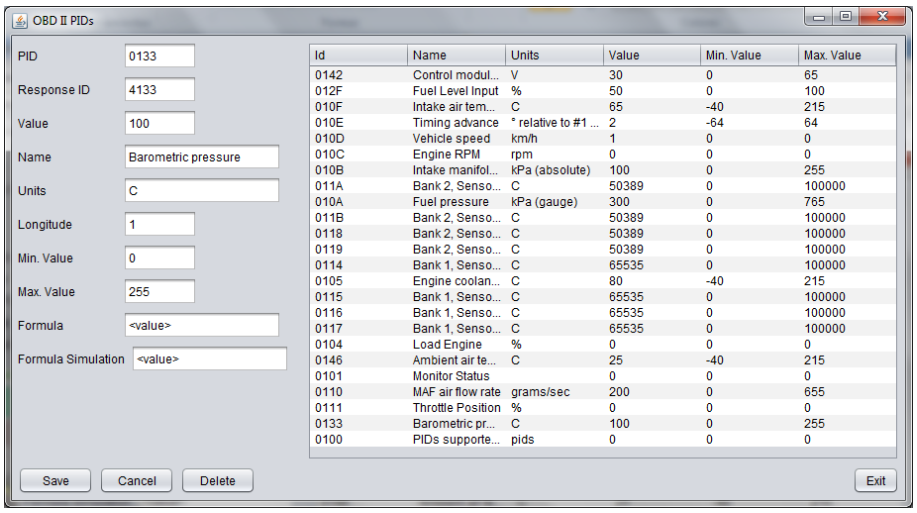

Figura 5.5.: Parámetros OBDII de BOSS.

- PID.- Es el identificador del parámetro OBDII PID; tiene una longitud de 4 dígitos hexadecimales, donde los dos primeros dígitos indican el modo, y los dos siguientes el identificador del PID.
- Response ID.- Indica el identificador con el cual se envía la respuesta de un PID. Normalmente es el mismo PID, cambiado el primer dígito “0” por “4”.
- Value.- Indica el valor que tiene el PID en ese momento. Cuando se introduce un nuevo parámetro este atributo indica el valor por defecto con el cual se inicializa.
- Name.- Indica el nombre o descripción del PID.
- Units.- Indica las unidades del valor del PID.
- Longitude.- Indica la longitud (número de bytes) que tiene el PID.
- Min. Value.- Indica el valor mínimo que puede tener el PID.
- Max. Value.- Indica el valor máximo que puede tener el PID.
- Formula.- Indica la fórmula con la cual se transforma el valor del PID al valor que se envía a través del enlace Bluetooth.
- Formula Simulation.- Indica la fórmula para el cálculo del valor de PID. En este campo se puede hacer referencia a otros PID para calcular el valor.

En las fórmulas, el valor propio del PID es <value>, y para hacer referencia a otros parámetros se usa <PID>. Por ejemplo, para un parámetro con id 01FF y que calcula su valor mediante su valor actual más el radio entre la velocidad (010D) y las rpm (010C) la fórmula sería: <value> + <010D> / <010C>

5.1.2. GPSEmulator

Es una aplicación para aplicaciones Android, en la cual se crea un enlace Bluetooth hacia BOSS-Server para recuperar los datos de localización (latitud, longitud) y se establece dicha posición en el sistema Android para emular la georeferenciación de un GPS. Para funcionar de manera correcta, GPSEmulator cancela los métodos de recuperación de la posición del sistema Android (como GPS, WiFi, triangulación GSM, etc.) y sobrescribe la información con los datos obtenidos previamente. El proceso de recuperación de la posición y de emulación de la georeferenciación se realiza de manera periódica como un servicio del sistema, es decir, el proceso se ejecuta incluso cuando se sale de la aplicación.

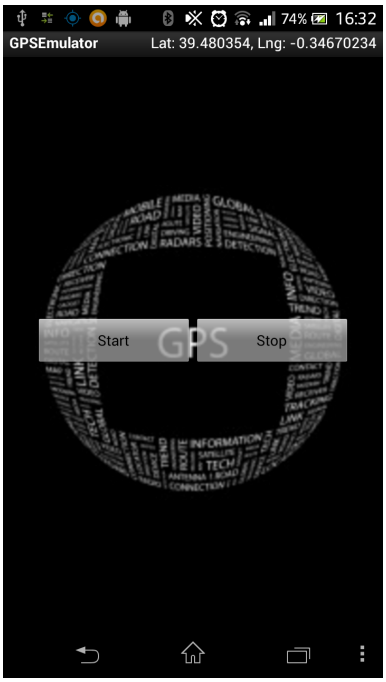

Figura 5.6.: GPSEmulator.

El interfaz de la aplicación es bastante sencillo ya que simplemente existen dos botones: (i) Start, que inicia el servicio de emulación, y (ii) Stop, que detiene el servicio de emulación.

Cuando se inicia el servicio se escoge el dispositivo Bluetooth en el cual se está ejecutando la aplicación BOSS, del cual se recupera la información; cuando se detiene el servicio se cierra la conexión Bluetooth hacia BOSS-Server, finalizando así la emulación de georeferenciación.

5.1.3. OBDIICapture

OBDIICapture es una aplicación desarrollada en Android de lectura de parámetros OBDII que crea un enlace de comunicación Bluetooth entre la aplicación y una interfaz OBDII Bluetooth (ELM327). Mediante este enlace se recupera la información de los parámetros del vehículo y se analiza el formato y los tiempos de respuesta de las peticiones.

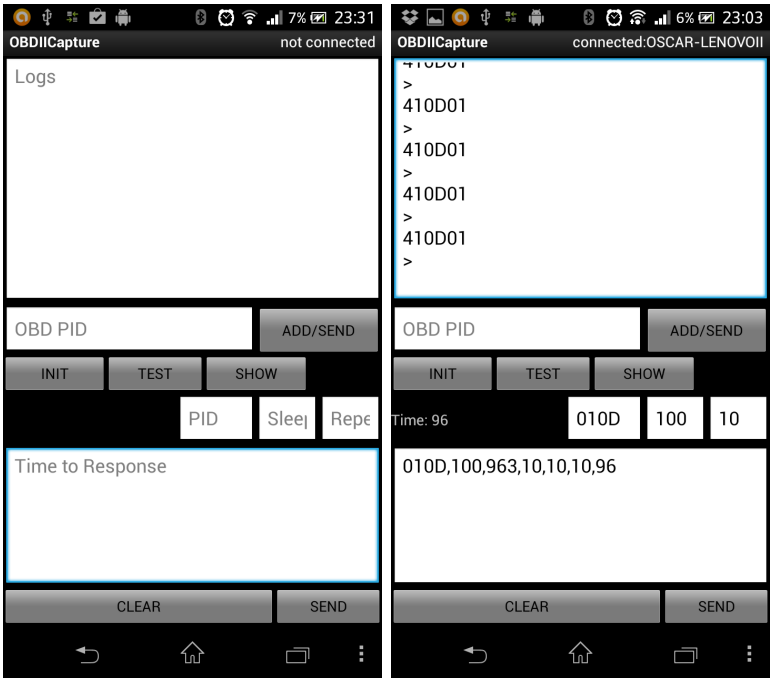

Figura 5.7.: OBDIICapture.

Esta aplicación se desarrolló con el fin de analizar de manera exacta el formato de los mensajes y los datos que se envían y se reciben mediante una interfaz OBDII Bluetooth y sus tiempos de respuesta, para a partir de esos datos desarrollar el emulador correctamente.

La aplicación tiene dos secciones:

1. Análisis de comandos. En esta sección se puede enviar cualquier comando hacia la interfaz OBDII, ya sean comandos AT o PIDs OBDII, para analizar el formato de la respuesta y el tiempo que tarda en recuperar al información. El comando se ingresa en la entrada “OBD PID” y se envía mediante el botón “ADD/SEND y los resultados se muestran en cuadro de “Logs”.

2. Análisis de tiempos de respuesta. En esta sección sirve para analizar las repuestas cuando se sobrecarga el sistema, es decir, se envían “n” peticiones de un comando “c”con un intervalo indicado “i”.
   Estos parámetros se ingresan en la aplicación de la siguiente manera: “n” se

especifica en PID, “c” en Repeat e “i” en Sleep. El botón Test ejecuta la prueba y el botón Show muestra los resultados en el cuadro “Response Time”

5.2. Arquitectura

BOSS está construido bajo una arquitectura en capas, en las cuales se independizan las diferentes funcionalidades del sistema, lo que permite trabajar de manera independiente en cada una de ellas facilitando así el desarrollo y mantenimiento del sistema.

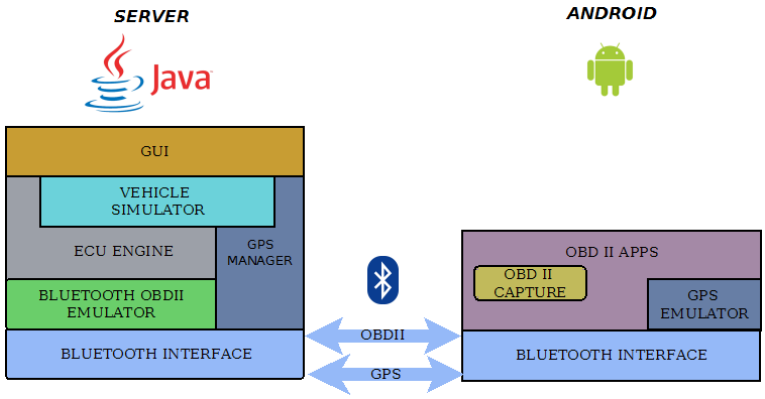

Figura 5.8.: Arquitectura del sistema BOSS.

Se debe tener en cuanta que BOSS es un conjunto de aplicaciones que funcionan en distintos sistemas en una estructura cliente/servidor formando una solución completa para realizar pruebas de funcionamiento en aplicaciones vehiculares móviles. Por un lado se tiene la parte del servidor que funciona en cualquier sistema operativo bajo la maquina virtual de Java y que genera los datos; y por otro lado se tiene la parte del cliente que funciona bajo Android, y el la cual se utilizan los datos generados en el servidor.

Dentro del servidor, las capas de datos y las que sirven de enlace a las demás capas es “ECU Engine” y “GPS Manager”. En estas capas se crean una estructura de datos en la cual se almacena los distintos parámetros del vehículo y de georeferenciación. Sobre estas capas está la capa “Vehicle Simulation”, donde se simula la dinámica del vehículo y se generan los datos de geoposicionamiento; por encima de las capas “Vehicle Simulation”, “ECU Engine” y “GPS Manager” está la capa “GUI” donde se muestran los datos del sistema en la parte del servidor y se capturan los acciones del usuario; por debajo de la capa “ECU Engine” está “Bluetooth OBDII Emulator”, donde se generan los mensajes Bluetooth OBDII en base a los datos almacenados en “ECU Engine”, y que se envían a través de la conexión Bluetooth. Además la capa “GPS Manager” es la encargada de gestionar las solicitudes de georeferenciación.

En la parte de los clientes, los datos son obtenidos mediante una conexión Bluetooth. Todas las aplicaciones Bluetooth OBDII, incluida OBDIICapture, obtienen los datos de los parámetros del vehículo mediante una conexión Bluetooth, y los datos de georeferenciación mediante los sensores del sistema (GPS principalmente). GPSEmulator corre como un servicio del sistema dentro de Android, y su función es recuperar los datos de georeferenciación mediante una conexión Bluetooth. Dicha conexión es independiente de la conexión Bluetooth para los parámetros OBDII del servidor BOSS, y permite emular la posición recuperada en el sistema.

El servidor BOSS es un sistema multihilo, es decir, se crean hilos de ejecución independientes para (i) la simulación de datos, (ii) el manejo del entorno gráfico, y (iii) para la comunicación Bluetooth. La comunicación entre hilos es mediante el paradigma de variable compartida, donde la estructura de datos "ECU Engine" es compartida por todos los hilos de procesamiento, y un cambio realizado en un hilo se ve reflejado en los otros automáticamente, comportándose como un sistema real.

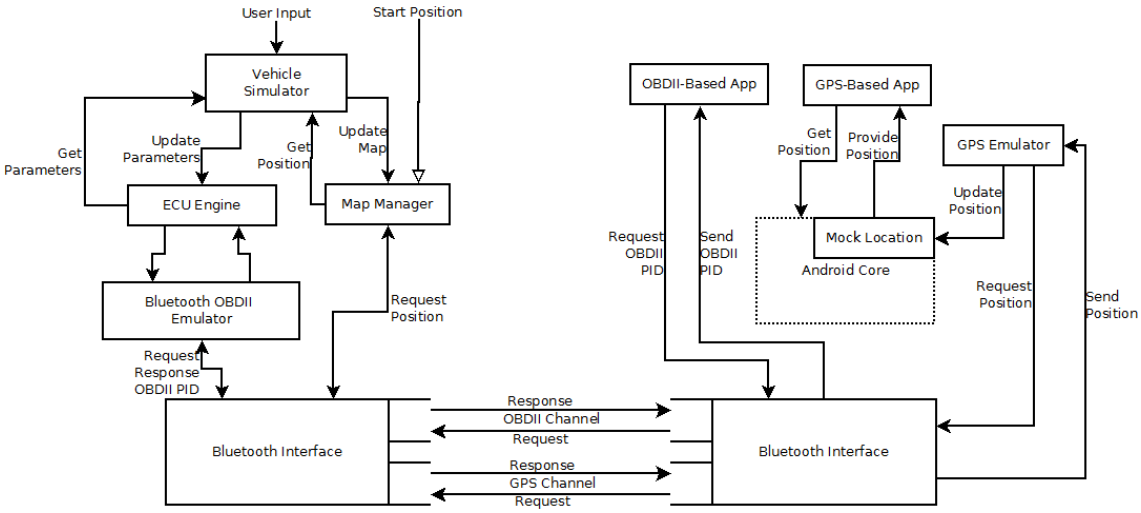

Figura 5.9.: Estructura del sistema BOSS.

5.2.1. ECU Engine

La capa "ECU Engine" simula la ECU de un vehículo. En esta capa se crea la estructura principal de datos , llamada "ECU Engine", donde se almacenan los parámetros simulados del vehículo, los parámetros de configuración para la simulación y los parámetros de la conexión Bluetooth OBDII. Esta es la capa central de los parámetros del vehículo, y las demás capas escriben o leen los datos sobre esta capa.

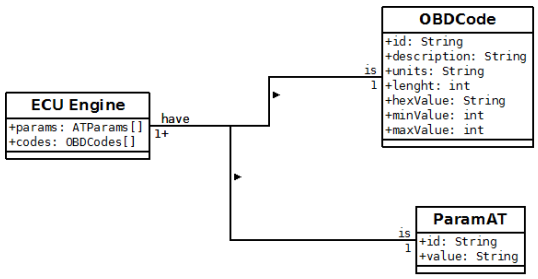

Figura 5.10.: ECU Vehicle.

5.2.2. Vehicle Simulator

Esta capa está diseñada para simular la dinámica de un vehículo en varios pasos: (i) recibe los datos de entrada (posición del pedal de aceleración y dirección) de la capa "GUI" (joystick), (ii) genera los parámetros del vehículo y los parámetros de georeferenciación, y (iii) almacena los datos generados en la estructura "ECU Engine" y "GPS Manager" respectivamente.

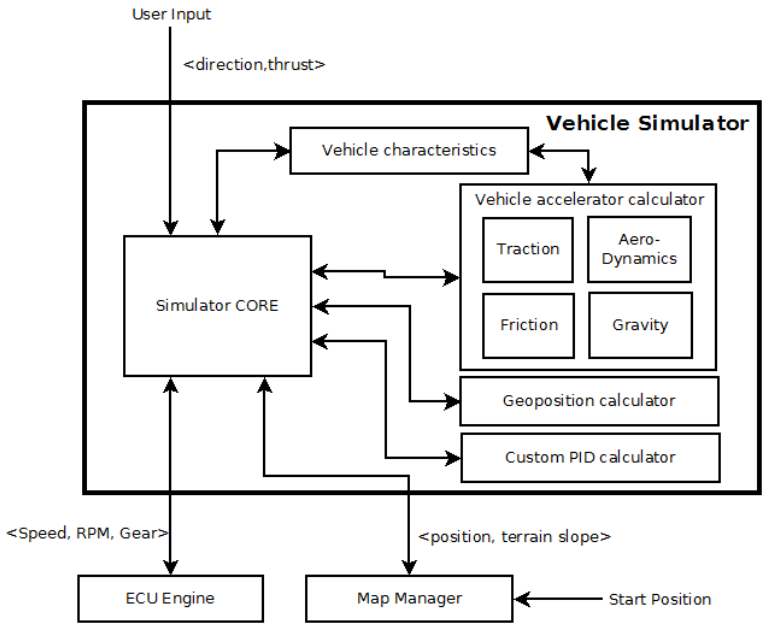

Figura 5.11.: Estructura de la Simulación del Vehículo.

Para realizar la simulación utiliza como datos de entrada: (i) la posición del pedal de aceleración, que es un valor entre -100 y 100 que indica el nivel de aceleración o frenado de vehículo y (ii) la dirección, que indica el ángulo de giro del vehículo. Junto con los datos actuales almacenados en "ECU Engine" y los parámetros internos de configuración, se realiza la simulación de los datos mediante los modelos matemáticos de la Sección 4.1, y se almacenan los datos generados en la capa "ECU Engine" para

que se emule la señal Bluetooth OBDII en la capa “Bluetooth OBDII Emulator” y se muestren los datos del vehículo en la capa “GUI”.

Para la simulación de la dinámica del vehículo se calcula el primer lugar la sumatoria de las fuerzas que actúan sobre este:

Algoritmo 5.1 Modelo de Fuerzas que actúan sobre un vehículo en movimiento.

```
void simulate(double acceleration, double direction) {
    double Vx = ECuVehicle.speed;
    double time = (new Date).toTime();
    double timeRun = time - lastTime;
    double Fa = calculateAerodynamicForce(Vx);
    double Fg = calculateGravityForce(0);
    double Fr = calculateFrictionForce(0);
    double Fe = calculateTractionForce(acceleration, Vx);
    double Ft = Fe - Fa - Fr - Fg;
    double ax = F / mass;
    Vx = ax * timeRun;
    ECUVehicle.speed = Vx;
    lastTime = time;
}
```

Fuerza Aerodinámica Mediante la fórmula Subsección 4.1.2, se calcula el valor de la fuerza aerodinámica en base al área frontal del vehículo, S; el coeficiente aerodinámico, Ca; la densidad del aire, p; y la velocidad del vehículo, Vx.

Algoritmo 5.2 Modelo de la fuerza Aerodinámica

```
double calculateAerodynamicForce(double Vx) {
    return (0.5) * S * Ca * p * Math.pow(Vx,2);
}
```

Fuerza de Gravedad Mediante la fórmula Subsección 4.1.3 se calcula el valor de la fuerza de la gravedad dependiendo de la masa del vehículo, mass; la fuerza de gravedad, g; y la inclinación del vehículo, anc. La inclinación se considera 0 para esta versión del simulador.:

Algoritmo 5.3 Modelo de la fuerza de Gravedad

```
double calculateGravityForce(double anc) {
    return mass * g * Math.sin(anc);
}
```

Fuerza de Rozamiento Mediante la fórmula Subsección 4.1.4 se calcula el valor de la fuerza de fricción del vehículo con la calzada dependiendo del coeficiente

de rozamiento, Cr; la masa del vehículo, mass; la fuerza de gravedad, g; y la inclinación del vehículo, anc. La inclinación se considera 0 para esta versión del simulador.:

Algoritmo 5.4 Modelo de la fuerza de Rozamiento

```
double calculateFrictionForce(double anc) {
    double Cr = u0 + u1 * Math.pow(Vx,2);
    return Cr * mass * g * Math.cos(anc);
}
```

Fuerza de Motor Mediante la fórmula Subsección 4.1.1 se calcula el valor de la fuerza que produce el vehículo a través del motor dependiendo de la posición del acelerador, acceleration; la velocidad inicial del vehículo, Vx; la marcha en que se encuentra, gear; la potencia máxima, Pm; la revoluciones máximas, Wm; y los ratios del sistema de transmisión dependiendo de la marcha en que se encuentra:

Algoritmo 5.5 Modelo de la fuerza de Motor

```
double calculateTractionForce(double acceleration, double Vx) {
    int gear = ECUVehicle.gear;
    double W0 = ECUVehicle.rpm;
    int rpmMin = MIN_RPM;
    int rpmMax = MAX_RPM;
    double cAcc = acceleration / 100;
    double Wm = RPMm * 2 * Math.PI / 60;
    double ni = nGear[gear];
    double rpm = ni * acc * f(Vx);
    if (rpm > rpmMax) {
        rpm = rpmMin;
        gear ++;
    } else if (rpm < rpmMin) {
        rpm = rpmMax;
        gear --;
    }
    double We = rpm;
    double P1 = Pm / Wm;
    double P2 = Pm / Math.pow(Wm,2);
    double P3 = Pm / Math.pow(Wm, 3);
    double Pe = P1 * We + P2 * Math.pow(We, 2) - P3 * Math.pow(We, 3);
    double Fe = Pe * ni / Vx;
    ECUVehicle.rpm = We;
    return Fe;
}
```

Para el calcular la marcha y el ratio del sistema de transmisión se basa en la ecuación Subsubsección 4.1.1.2 donde, dependiendo de la velocidad lineal del vehículo, se calcula la marcha por los límites mínimo y máximo que se establecen.

5.2.3. Geoposition calculator

El cálculo de la geoposición del vehículo es una subparte del simulador del vehículo. En esta sección se generan los datos de geoposicionamiento utilizados por GPS Manager, basado en los modelos estudiados en la sección Subsección 4.2.1.

Para la simulación de la posición se debe establecer la posición inicial del vehículo dentro del mapa de la capa "GUI" y una vez iniciado el proceso de simulación utilizará esta posición para determinar la ubicación del vehículo en movimiento. La posición se actualiza en la capa "GPS Manager" y esta enviará los datos a las peticiones de goereferenciación.

Para el cálculo de la nueva posición se toman como referencia: (i) la posición actual del vehículo, (ii) la orientación, y (iii) la distancia, que se obtiene mediante la velocidad del vehículo y el tiempo entre muestreos.

Algoritmo 5.6 Modelo de Georeferenciacion

```
public static GeoPoint getPosition(double lat, double lng, double bearing, double
distance) {
    double dist = distance / EARTH_RADIUS;
    double lon1 = Math.toRadians(lng);
    double lat1 = Math.toRadians(lat);
    double brng = Math.toRadians(bearing);
    double lat2 = Math.asin(Math.sin(lat1) * Math.cos(dist) + Math.cos(lat1) *
Math.sin(dist) * Math.cos(brng));
    double lon2 = lon1 + Math.atan2(Math.sin(brng) * Math.sin(dist) *
Math.cos(lat1), Math.cos(dist) - Math.sin(lat1) * Math.sin(lat2));
    lon2 = (lon2 + 3 * Math.PI) % (2 * Math.PI) - Math.PI;
    GeoPoint resp = new GeoPoint(Math.toDegrees(lat2), Math.toDegrees(lon2));
    resp.setBearing(((bearing + 180) % 360));
    return resp;
}
```

5.2.4. Bluetooth OBDII Emulator

La capa "Bluetooth OBDII Emulator" es la encargada de gestionar la comunicación Bluetooth OBDII con las aplicaciones vehiculares móviles y configurar el formato de las respuestas. En esta capa se procesan las peticiones entrantes de las conexiones Bluetooth y se responden con los datos almacenados en "ECU Engine".

Se puede configurar los parámetros del emulador mediante los comandos AT, y esta capa es la encargada de gestionar estas solicitudes y realizar las configiraciones respectivas en el simulador. Las configuraciones que se pueden realizar están descritas en la Tabla 5.1

Comando	Descripción
AT D	Resetea todas las configiraciones
AT Z	Resetea todas las configiraciones
AT E0, E1	Habilita o deshabilita el echo.
AT L0, L1	Habilita o deshabilita un salto de linea luego de CR.
AT S0, S1	Habilita o deshabilita los espacios entre
AT SP h	Establece el protocolo h. Esto cambia los tiepos de respuesta

Cuadro 5.1.: Comandos AT configurables en BOSS

Además, en esta capa se gestiona la conexiones Bluetooth para las peticiones de parámetros OBDII de aplicaciones móviles vehiculares. Solo se permite una conexión entrante para OBDII y una conexión entrante para GPS al mismo tiempo.

5.2.5. GPS Manager

La capa "GPS Manager" es la encargada de manejar la comunicación Bluetooth GPS y almacenar los datos de georeferenciación. En esta capa se procesan las peticiones entrantes de las conexiones Bluetooth y se responden con los datos almacenados.

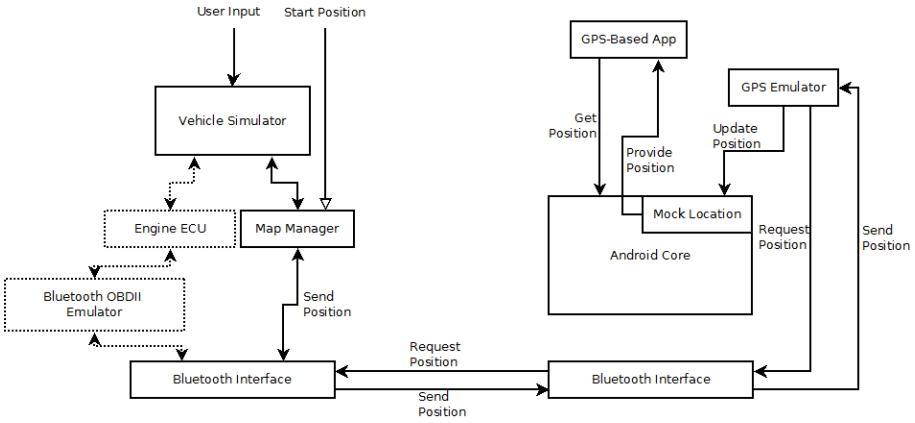

Figura 5.12.: Estructura de GPS Emulator.

La posición inicial se ingresa mediante el mapa de la capa "GUI", y los datos son actualizados mediante la capa "Vehicle Simulation". Cuando una aplicación móvil requiere los datos, esta capa genera la respuesta con los datos almacenados en ese momento y gestiona la conexión Bluetooth.

5.2.6. GUI

La capa gráfica de BOSS se encarga de (i) controlar la simulación, (ii) mostrar los datos de la simulación del vehículo, (iii) mostrar los datos de la emulación Bluetooth OBDII y Bluetooth GPS.

Para controlar la simulación de desarrolló un joystick con el cual se puede establecer el nivel de aceleración, nivel de frenada y dirección del vehículo en un mapa.

Los datos del vehículo y su posición se muestran en un entorno gráfico de ventanas, además se muestra en un mapa la ubicación y dirección del vehículo.Se muestran los datos básicos del vehículo y la georeferenciación, y la comunicación Bluetooth de la emulación.

Los datos de emulación se muestran en tres secciones, una para las peticiones "Request", otra para las respuestas "Response" y una para los logs generales tales como avisos del estado de la conexión o PIDs no encontrados "Logs".

5.3. Consideraciones de Desarrollo

En el momento de desarrollo se tuvieron en cuenta algunos aspectos, como el lenguaje de programación, las librerías externas a utilizar, etc. A continuación se describen los más importantes.

5.3.1. Lenguajes de programación

Para el desarrollo del sistema BOSS se ha seleccionado Java para la aplicación servidor, y la plataforma Android para las aplicaciones cliente.

Java — Es uno de los lenguaje de programación más difundidos actualmente; se trata de un lenguaje multiplataforma que se ejecuta sobre un máquina virtual (JVM, Java Virtual Machine) y que no depende del sistema operativo, existiendo una gran variedad de librerías externas libres para el desarrollo.

Android — Es uno de los sistemas operativos para móviles más utilizado en la actualidad ya que es bastante robusto y ofrece muchas funcionalidades, además de ser de código abierto.

5.3.2. Librerías

Para el desarrollo del servidor BOSS en Java se han utilizado varias librerías externas y bibliotecas gráficas.

Swing Es una biblioteca gráfica de Java que incluye widgets (elementos gráficos) para la interfaz de usuario como botones, cajas de texto, menús, tablas, etc.

Bluecove Es una librería externa de Java para la comunicación serial Bluetooth, la cual tiene soporte para sistemas operativos Windows, Linux y MacOS. Para sistemas Linux es necesario, además, instalar los módulos del sistema BlueZ, e incluir las librerías java BlueCove-GPL que permiten utilizar los módulos BlueZ.

JMapViewer Es una librería externa de Java que permite manipular mapas de OpenStreetMap.

5.3.3. Conexión Bluetooth

Para la transmisión de datos del servidor BOSS a cualquier aplicación móvil vehicular que requiera los datos de OBDII se necesita emular la señal Bluetooth que generan los intérpretes Bluetooth OBDII. Los intérpretes OBDII crean una comunicación Bluetooth con un perfil de comunicación serie (COM) y un UUID específico.

```
local = LocalDevice.getLocalDevice();
local.setDiscoverable(DiscoveryAgent.GIAC);
UUID uuid = new UUID("0000110100001000800000805F9B34FB", false);
String url = "btspp://localhost:" + uuid.toString()
        + ";name=" + name;
notifier = (StreamConnectionNotifier) Connector.open(url);
```

Figura 5.13.: UUID Blueooth OBDII.

El UUID “0000110100001000800000805F9B34FB” identifica al servicio como un intérprete Bluetooth OBDII, y es necesario desarrollar la aplicación para que emita la señal Bluetooth con este UUID específico.

5.3.4. Mock Location

Los sistemas Android permiten establecer localizaciones geográficas de prueba para realizar pruebas de movilidad. Para establecer las localizaciones de pruebas o “Mock Locations” es necesario: (i) habilitar en la configuración del sistema la opción “Ubicaciones simuladas”, y (ii) mediante programación establecer la posición simulada mediante la función setTestProviderLocation del administrador de ubicaciones “LocationManager”. Una vez establecida, la ubicación de pruebas es transparente para las aplicaciones, ya sea simulada o real.

Los pasos para establecer una posición simulada son:

```
location.setLatitude(latitude);
location.setLongitude(longitude);
location.setTime(System.currentTimeMillis());
locationManager.setTestProviderLocation(
        GPSEmulatorService.nameGPSService, location);
```

Figura 5.14.: Mock Location.

- Habilitar la opción de "Ubicaciones simuladas" en las opciones de desarrollador del sistema.
- Dar los permisos de "ACCESS_MOCK_LOCATION" a la aplicación en el manifiesto.
- Habilitar el modo MOCK:
  mLocationClinet.setMockMode(true).
- Especificar el proveedor de ubicaciones de prueba:
  locationManager.setTestProviderLocation("proveedorPrueba",testLocation)
- Establecer la ubicación simulada:
  mLocationClient.setMockLocation(testLocation).

6. Pruebas de funcionamiento

De cara a validar la plataforma desarrollada, se han realizado la pruebas de funcionamiento en varias aplicaciones móviles vehiculares como Torque, OBD Car, DrivingStyles, etc. En cada una de ellas se ha probado la secuencia de inicio de conexión, la transmisión de datos Bluetooth OBDII y la simulación de localizaciones de prueba.

En las pruebas de funcionamiento se comprobó que los parámetros OBD-II generados por el simulador y los parámetros OBD-II recibidos en las aplicaciones móviles se correspondan, y que la utilización del simulador en vez de un vehículo real sea transparente para dichas aplicaciones. Por otro lado, se comprobó que mientras se ejecutara el servicio de GPSEmulator la ubicación reconocida por el sistema sea la generada en la simulación.

6.1. Pruebas Generales

Para realizar las pruebas generales de funcionamiento se han utilizado las aplicaciones móviles OBD-II más populares en el mercado, como Torque, OBD Car y Car Gauge Lite. En estas aplicaciones se probó: (i) el proceso de conexión y establecimiento de los parámetros del enlace de comunicación, (ii) la lectura de los parámetros OBDII y (iii) la emulación de las posiciones geográficas en el sistema.

Figura 6.1.: Pruebas de Funcionamiento

6.1.1. Pruebas de establecimiento de enlace

En el momento de conexión todas las aplicaciones tienen una secuencia más o menos similar. Concretamente,a secuencia en Torque es:

1. Se resetea el dispositivo con el comando ATZ, se establece todas las configuraciones por defecto con ATD, se elimina el echo con ATE0, etc.
2. Luego se buscan los protocolos que soporta el dispositivo con los comandos ATSP0, ATSP1, ATSP2, etc.
3. Por último, se verifica los parámetros que soporta el dispositivo con el comando 0100.

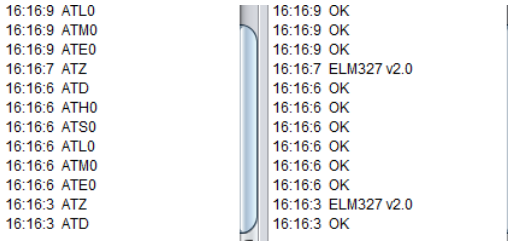

Figura 6.2.: Conexión con Torque.

Se verificó que nuestra aplicación responde correctamente a todas las solicitudes, y que se establece el enlace de comunicación sin problemas.

6.1.2. Pruebas de transmisión parámetros OBDII

Para probar el correcto funcionamiento de la transmisión de parámetros OBDII por parte de BOSS hacia las aplicaciones, se han realizado varias pruebas con cada una de las aplicaciones, donde se comprobó que:

1. Las solicitudes son correctamente recibidas a través de la conexión Bluetooth en BOSS.
2. Las respuestas se formatean por parte de BOSS siguiendo el formato indicado por el estándar OBD-II, y son enviadas por el enlace Bluetooth.
3. Los datos son correctamente interpretados por las aplicaciones móviles.

Además, como ejemplo del potencial de la plataform desarrollada, se realizó una prueba de ingeniería inversa para determinar los parámetros requeridos por Torque para calcular el consumo de combustible cuando no es posible recuperar el parámetro “0110: MAF air flow rate”, comúnmente utilizado para este propósito. Nuestra aplicación muestra que Torque calcula dicho consumo en base a:

- 010B: Intake manifold absolute pressure
- 010C: Engine RPM

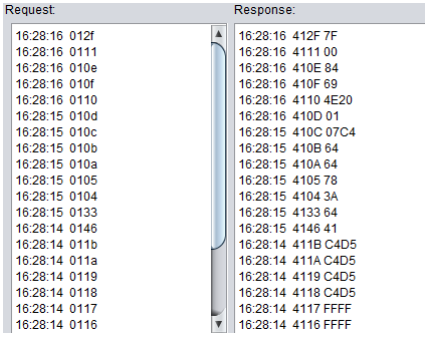

Figura 6.3.: Transmisión de datos hacia Torque.

- 010D: Vehicle speed
- 010F: Intake air temperature

6.1.3. Pruebas georeferenciación

Se han realizado varias pruebas para la simulación de ubicaciones, y ha comprobado su funcionamiento en aplicaciones que utilizan georeferenciación, como GoogleMaps, verificándose que dichas aplicaciones reconocen las ubicaciones simuladas de manera transparente.

Figura 6.4.: Pruebas de Georeferenciación.

6.1.4. Pruebas generación de parámetros personalizados

Se verificó que se pueda agregar y modificar nuevos parámetros OBD-II en tiempo de ejecución, y que el cálculo de los valores sea acorde con las fórmulas especificadas.

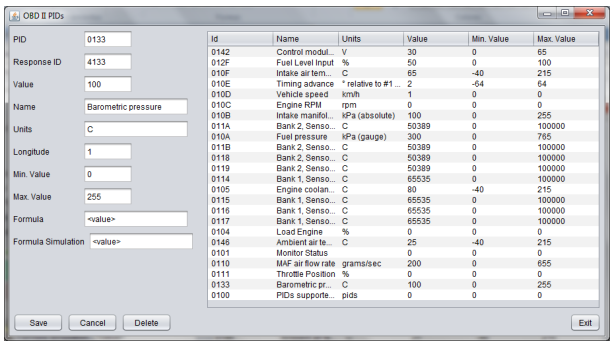

Figura 6.5.: Parámetros Personalizados

6.2. Pruebas con Driving Styles

La mayor cantidad de pruebas de funcionamiento se han hecho con la aplicación DrivingStyles, ya que es una aplicación desarrollada en el Grupo de Redes de Computadores y que tiene la estructura de las aplicaciones que se desarrollan en el grupo. En dichas pruebas ha quedado patente el potencial y la utilidad del emulador desarrollado para acelerar el desarrollo de este tipo de aplicaciones.

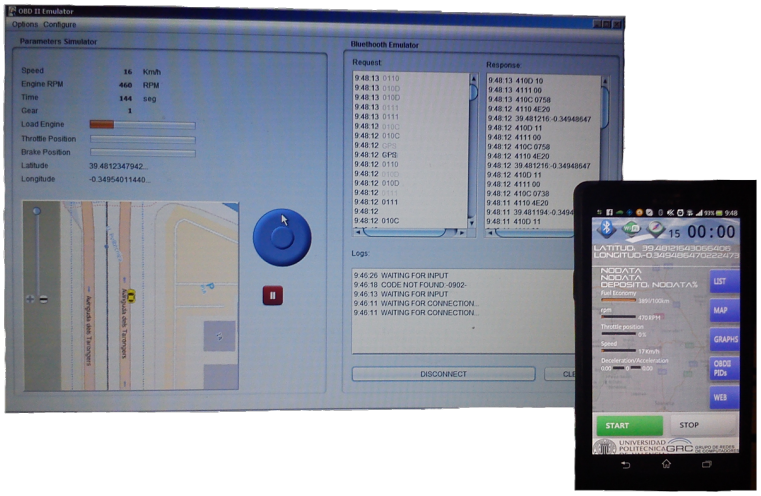

Figura 6.6.: Pruebas con DrivingStyles.

Los resultados obtenidos fueron satisfactorios, ya que BOSS ha emulado correctamente tanto la georeferenciación como los datos OBD-II.

6.2.1. Pruebas de establecimiento de enlace

En DrivingStyles el establecimiento de la conexión es similar al de otras aplicaciones siguiendo los tres pasos iniciales: (i) resetear los parámetros del dispositivo, (ii) buscar el protocolo que soporta, y (iii) buscar los parámetros disponibles. Al igual que

en otras aplicaciones, la comunicación se realiza sin problemas, y para la aplicación el uso de un emulador es totalmente transparente.

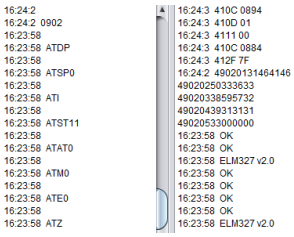

Figura 6.7.: Conexión hacia DrivingStyles

6.2.2. Pruebas de funcionamiento

La solicitud de los parámetros OBD-II con DrivingStyles sigue un patrón similar al de otras aplicaciones, y todas las pruebas realizadas no tuvieron ningún inconveniente salvo que en cada solicitud de PID, DrivingStyles agregaba un salto de línea '\n' junto con el CR '\r', el cual el simulador tomaba como otro comando en blanco. Se adecuó BOSS para eliminar los caracteres adicionales y analizar solo los comandos válidos.

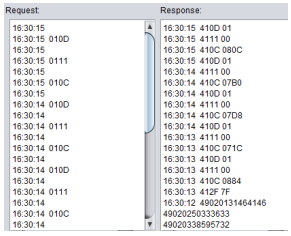

Figura 6.8.: Transmisión de datos hacia DrivingStyles

Además, sehan ejecutado varias pruebas de simulación en las que se circulaba alrededor de la ciudad de Valencia, España, verificándose que DrivingStyles reconocía de manera transparente las posiciones simuladas.

6.3. Análisis de conexión

Dado que en algunos tipos de aplicaciones vehiculares móviles el tiempo de respuesta y la exactitud de los datos es muy importante para su correcto funcionamiento, como expone Zaldivar en su trabajo de detección de accidentes con smartphones[21], se analizó el tiempo de respuesta de las comunicaciones basadas en OBD-II mediante un adaptador Bluetooth.

El tiempo de respuesta de es un aspecto muy importante de tener en cuenta para el desarrollo de las aplicaciones vehiculares móviles, ya que depende de este para poder establecer el número de peticiones que se pueden solicitar por segundo a un vehículo y conocer la exactitud de los datos y la cantidad de datos con la que se puede trabajar.

Para analizar el tiempo de respuesta de un vehículo real, así como de la plataforma BOSS, se han realizado dos tipos de pruebas:

1. Pruebas de estrés (multipetición), donde se envían "ráfagas" de solicitudes con intervalos de tiempo pequeños entre cada solicitud, y se analizan las respuestas y el tiempo necesario para ejecutar toda la prueba.

2. Pruebas de peticiones individuales, donde se solicita un solo parámetro de cada vez, y se analiza el tiempo necesario para responder dicha solicitud.

Para cada una de estas pruebas se han probado tres tipos de PIDs:

1. 010D. (Velocidad) PID de un byte de longitud.

2. 010C. (RPM) PID de dos bytes de longitud.

3. 0134. (O2S1) PID de cuatro bytes de longitud.

6.3.1. Pruebas multipetición

Se han realizado pruebas de estrés en un vehículo Kia Sportage (2011) enviando un conjunto de solicitudes de PIDS, cada una con un tiempo entre peticiones variable en el rango de 10 a 100 ms.

A continuación se presentan los resultados obtenidos.

6.3.1.1. Pérdidas de respuestas

En la figura Figura 6.9 se muestran los resultados obtenidos cuando variamos el tiempo entre peticiones OBD-II consecutivas. Se verifica que las pérdidas de paquetes se estabilizan en valores prácticamente nulos a partir de un intervalo entre peticiones de aproximadamente 55 ms, lo que corresponde a aproximadamente a 18 peticiones por segundo. Cuando enviamos peticiones con tiempos entre peticiones consecutivas menores que 55 ms existen pérdidas de respuestas o respuestas erróneas, y dicho comportamiento se agrava a medida que reducimos dicho tiempo. .

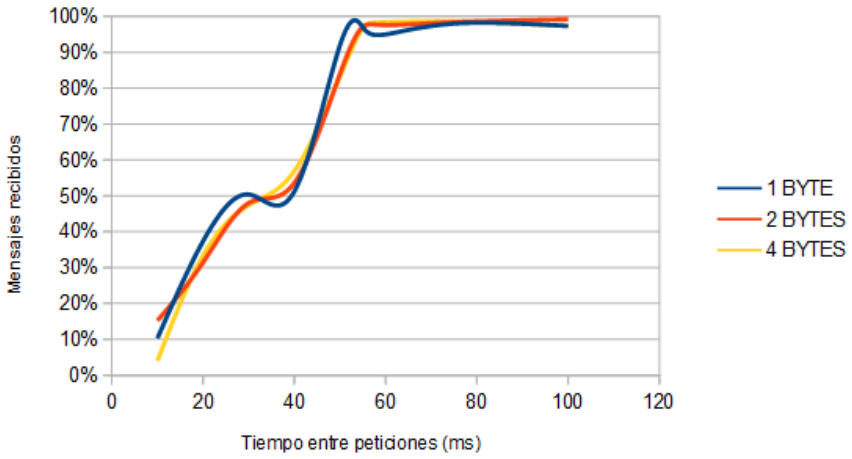

Figura 6.9.: Análisis de paquetes recibidos en vehículo KIA

6.3.1.2. Tiempos de respuesta

En la figura Figura 6.10 se puede observar el tiempo de respuesta del sistema a medida que variamos el tiempo entre peticiones. Podemos observar que, cuando existen respuestas incorrectas, el tiempo de respuesta medio es muy superior a los tiempos de respuesta cuando el intervalo entre peticiones es mayor y no existen pérdidas. Esto se debe a que, al haber pérdidas, no se responden a todas las solicitudes. Por esa razón no es recomendable realizar peticiones con intervalos menores al tiempo de estabilización, en este caso 55 ms, que es tiempo con el que mejor tasa se alcanza.

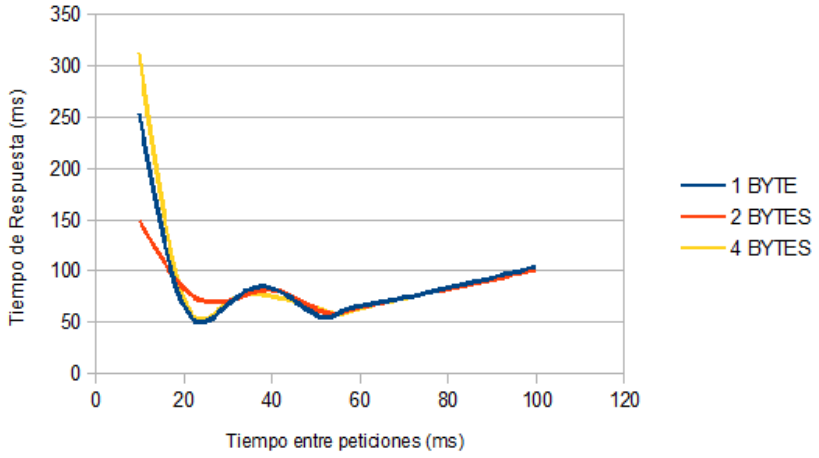

Figura 6.10.: Tiempos de respuesta para distintos tamaños de mensajes variando el tiempo entre peticiones consecutivas.

Realizando las mismas pruebas con BOSS (ver figura Figura 6.11) se puede observar que no se tienen pérdidas, ya que BOSS cuando tiene muchas peticiones simplemente retrasa su respuesta pero no descarta ninguna solicitud. A partir del tiempo de estabilización, el comportamiento es prácticamente el mismo que el de un vehículo real, tal y como se pretendía.

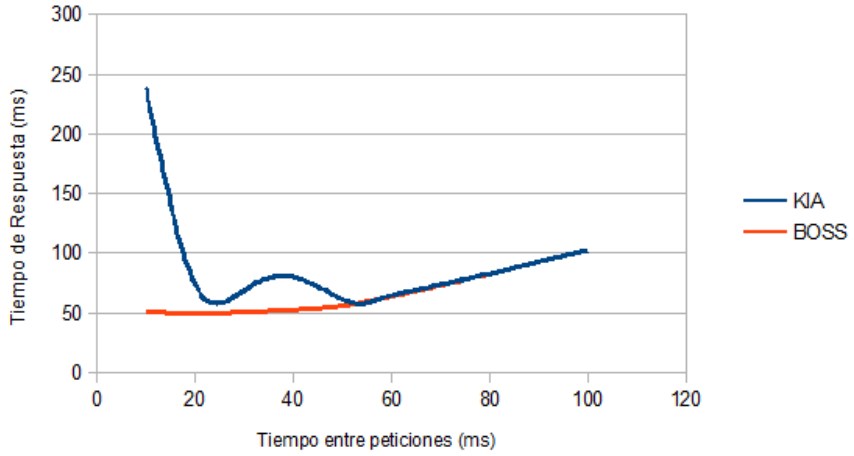

Figura 6.11.: Comparación de los tiempos de respuesta de un coche real y del emulador variando el tiempo entre peticiones consecutivas.

6.3.1.3. Tasa de respuestas correctas

La tasa de respuestas correctas tiene una variación más o menos aleatoria mientras hay pérdidas de paquetes, pero cuando se envían solicitudes con diferencias superiores al tiempo de estabilización, la tasa de respuestas viene dada por el tiempo entre solicitudes. Además, se puede observar que las pérdidas no dependen del tamaño de los mensajes, siendo prácticamente las mismas para los tres PIDs utilizados.

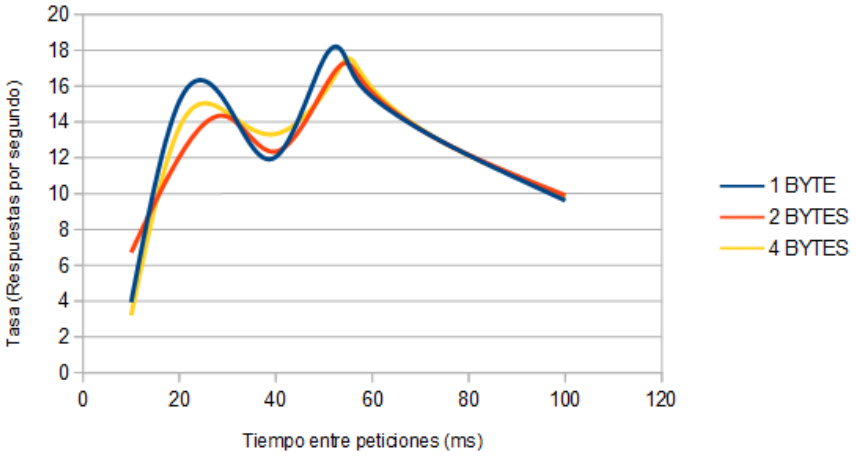

Figura 6.12.: Tasa de respuestas por segundo para distintos tamaños de mensajes variando el tiempo entre peticiones consecutivas.

Como en BOSS no existen pérdidas de mensajes, se puede observar que la tasa de respuestas en un vehículo real y en BOSS es distinta cuando se tienen pérdidas de mensajes (ver figura Figura 6.13), pero es prácticamente la misma a partir del tiempo de estabilización, que es cuando se deja de perder mensajes.

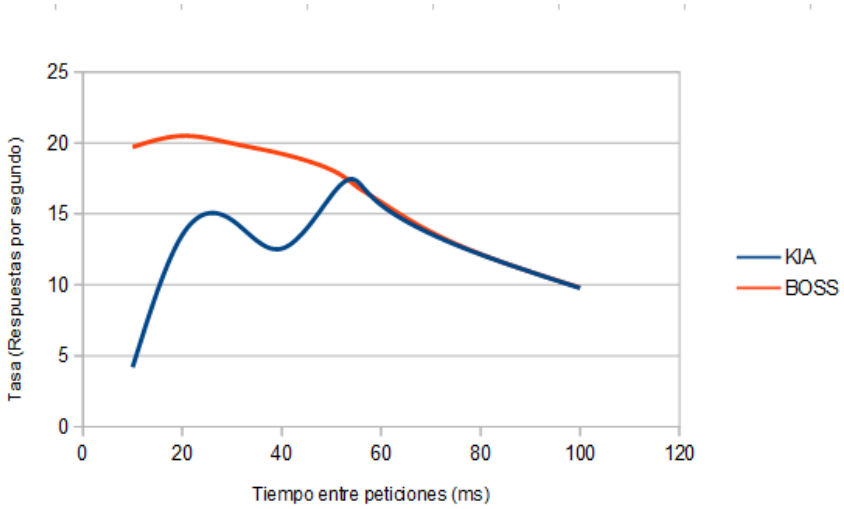

Figura 6.13.: Comparación de la tasa de respuestas por segundo de un coche real y del emulador variando el tiempo entre peticiones consecutivas.

6.3.2. Análisis de la dispersión de retardo

Otro aspecto importante a analizar es el tiempo que se demora en responder una sola petición, sin congestión del canal, y analizar la variabilidad de los resultados. Para el análisis de la variabilidad de los tiempos de respuesta se graficó la función densidad de probabilidad de los datos obtenidos.

En la figura Figura 6.14 se puede observar que en general la distribución de estos tiempos de respuesta está mayoritariamente entre los 80 y 110 ms, habiendo pequeñas diferencia entre los distintos tamaños de mensaje. Como se puede verificar, no hay una correlación directa entre el tiempo de respuesta y el tamaño del mensaje, debiéndose por eso asociar dicho tiempo a las especificidades de cada PID.

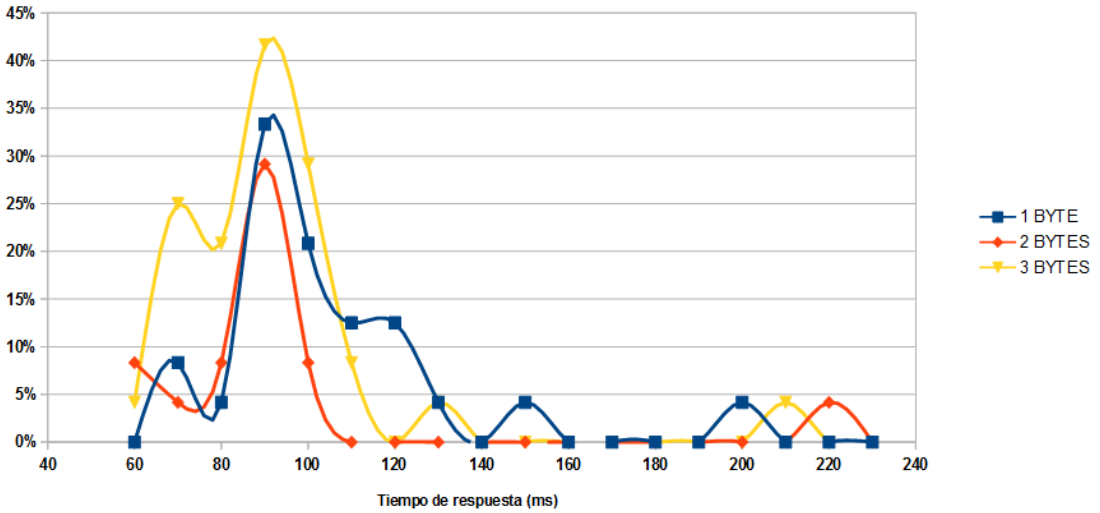

Figura 6.14.: Función densidad de probabilidad para el tiempo de respuesta del OBD-II para distintos tamaños de mensaje.

Haciendo un análisis entre los datos medios del vehículo Kia y los datos obtenidos de la plataforma BOSS (ver figura Figura 6.15) se puede observar que la mayor cantidad de datos se encuentran, en ambos casos, entre 80 y 110 ms y que la media es prácticamente la misma, siendo la dispersión de los datos ligeramente superior en el vehículo real.

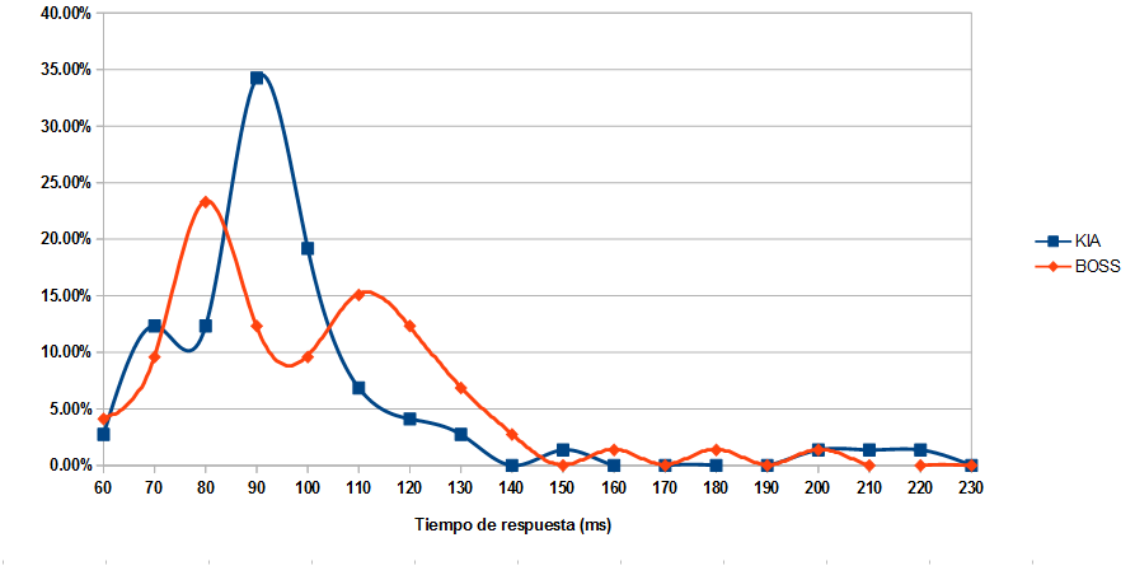

Figura 6.15.: Función densidad de probabilidad para el tiempo de respuesta del OBD-II compara.

Globalmente, los resultados experimentales muestran que la plataforma BOSS permite emular con un elevado grado de similitud el comportamiento de un vehículo real en términos de comunicación con una ECU mediante interfaz OBD-II.

7. Conclusiones

7.1. Sumario y Conclusiones

Teniendo en cuenta los importantes desarrollos realizados en el área de las Tecnologías de la Información y Comunicación, y en especial en los dispositivos móviles, así como la posibilidad de recopilar información de la Unidad de Diagnóstico a Bordo (OBDII) de los vehículos en tiempo real a través del intérprete ELM327, se abre una línea de investigación muy amplia en el ámbito del "Smart Driving". Esta línea de investigación se ve limitada por la necesidad de tener un vehículo con un interprete Bluetooth OBDII para realizar las pruebas de funcionamiento. La solución más factible para solucionar este problema seria un simulador en el que se generen los datos de un vehículo y se transmitan a través de una conexión Bluetooth, emulando un intérprete OBDII Bluetooth.

Un estudio de las soluciones existentes en la actualidad permitió verificar que en el mercado existen soluciones tanto en hardware como en software para este problema, pero ninguna de ellas satisface todas las necesidades existentes. Las soluciones en hardware son costosas y tienen un número limitado de códigos que simulan, al estar orientados al entrenamiento de personal para mantenimiento un tipo de vehículo concreto; aquellos basados en software son demasiado sencillos y la mayoría no soporta la conexión Bluetooth, sino simplemente una conexión serie, habiendo sido desarrollados básicamente para pruebas sencillas.

Nuestra propuesta para solucionar el problema es "BOSS: Bluetooth OBDII Simple Simulator" y entre sus principales características cabe destacar:

- Multiplataforma: Está desarrollado en Java y usa la librería bluecove para la conexión Bluetooth, lo que lo convierte en un sistema multiplataforma probado tanto en Windows como en Linux.
- Interfaz gráfica amigable: Su utilización es muy amigable e intuitiva, simplemente se "enciende" el vehículo y se controla su ruta a través de un mapa.
- Flexible: Se puede agregar o eliminar los diferentes parámetros a simular, así como especificar la fórmula para el cálculo de su valor, referenciando en dicha fórmula a otros parámetros ingresados.
- Simulación de Georeferenciación: Mediante una aplicación adicional que ha sido desarrollada, GPS Emulator, se puede emular geoposicionamiento en cualquier sistema Android, con los datos generados por BOSS, en tiempo real.

- Control de la comunicación Bluetooth - OBDII: se puede observar en tiempo real toda la comunicación entre el simulador y el dispositivo móvil, y todos los eventos generados en la misma. Esto puede ser útil para analizar el correcto funcionamiento de las aplicaciones.

Se han realizado pruebas de funcionamiento usando varias aplicaciones, como Torque, OBD Doctor, OBD Car Doctor, y Driving Styles, tomando mayor énfasis este último por ser una aplicación desarrollada por el Grupo de Redes de Computadores, y sabiendo que es esquema que se una aplicación que se seguirá desarrollando dentro del grupo. Además, se realizó un estudio comparativo de tiempos de respuesta a solicitudes de parámetros entre un vehículo real y BOSS para verificar el funcionamiento, donde se comprobó que existe un elevado grado de similitud entre ambos.

Con BOSS se ha creado una infraestructura de pruebas para las aplicaciones móviles orientadas a redes vehiculares facilitando su desarrollo, ya que se puede, de una manera muy fácil y barata, realizar una gran cantidad de pruebas en múltiples escenarios, sin la necesidad de tener un vehículo real.

7.2. Trabajo Futuro

El trabajo futuro se puede clasificar en 3 áreas: (i) refinamiento del simulador, (ii) integración con otro simuladores o generadores de tráfico, y (iii) análisis y definición de una arquitectura estándar para las aplicaciones móviles vehiculares.

- En cuanto al refinamiento del simulador, se podría: (i) parametrizar algunas variables del internas al simulador, como velocidades de transferencia, activación/desactivación de algunos parámetros de ELM327, la manipulación de la comunicación multilínea; (ii) parametrizar algunas variables del modelo de simulación; (iii) crear un módulo de reproducción de trazas tomadas en vehículos reales, etc.
- Se puede también, extender OBDCapture para que capture una traza completa de un recorrido y crear otro módulo en el servidor BOSS para la reproducción de la traza capturada.
- Además, se podría integrar BOSS en otros simuladores como OMNET++ y SUMO, para simular tráfico y tener un entorno más realista teniendo la interacción de nuestro vehículo simulado en BOSS, con otros vehículos simulados por los otros simuladores.
- Por último, se podría analizar y diseñar una arquitectura para el desarrollo de aplicaciones móviles vehiculares, donde se establezcan las diferentes capas de la aplicación y donde se independicen la parte de la comunicación, la sensorización y el análisis de datos, disminuyendo la complejidad y tiempo de desarrollo de las aplicaciones.

7.3. Reconocimiento

Este trabajo se ha realizado dentro del Grupo de Redes de Computadores de la Universidad Politécnica de Valencia, con el financiamiento de estudios de la Universidad de Cuenca - Ecuador y el Programa de Becas de la Convocatoria Abierta 2012 Segunda Fase de la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación del Ecuador.

A. OBDII PIDs

A.1. Descripción

OBDII PIDs (OnBoard Diagnostics Parameter IDs) son los códigos de identificación de los distintos parámetros que se pueden medir en un vehículo. El estándar OBDII de la SAE J1979 (Society of Automotive Engineers) define algunos parámetros, pero los fabricantes de vehículos pueden e introducen sus propios códigos.

A.2. Estándar

El estándar define un conjunto de parámetros que deben proveer todos los fabricantes. Eston brindan una bgran cantidad de datos del funcionamiento del vehículo. A continuación se muestra un cuadro con algunos de los parametros:

id	sz	description	min	max	units	formula
46	1	Ambient air temperature	-40	215	°C	A-40
51	1	Fuel Type				
52	1	Ethanol fuel %	0	100	%	A*100/255
00	4	PIDs soportados [0 -20]				
01	4	Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL) status and number of DTCs.)				
02	2	Freeze DTC				
03	2	Fuel system status				

id	sz	description	min	max	units	formula
46	1	Ambient air temperature	-40	215	°C	A-40
51	1	Fuel Type				
52	1	Ethanol fuel %	0	100	%	A*100/255
04	1	Calculated engine load value	0	100	%	A*100/255
05	1	Engine coolant temperature	-40	215	°C	A-40
06	1	Short term fuel % trim—Bank 1	-100 Sub-tracting Fuel (Rich Condi-tion)	99.22 Adding Fuel (Lean Condi-tion)	%	(A-128) * 100/128
0A	1	Fuel pressure	0	765	kPa (gau-ge)	A*3
0B	1	Intake manifold absolute pressure	0	255	kPa (ab-solu-te)	A
0C	2	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
0D	1	Vehicle speed	0	255	km/h	A
0E	1	Timing advance	-64	63.5	° re-lative to #1 cylin-der	A/2 - 64
0F	1	Intake air temperature	-40	215	°C	A-40
10	2	MAF air flow rate	0	655.35	grams/sec	((A*256)+B) / 100
11	1	Throttle position	0	100	%	A*100/255
12	1	Commanded secondary air status				

id	sz	description	min	max	units	formula
46	1	Ambient air temperature	-40	215	°C	A-40
51	1	Fuel Type				
52	1	Ethanol fuel %	0	100	%	A*100/255
13	1	Oxygen sensors present				[A0..A3] == Bank 1, Sensors 1-4. [A4..A7] == Bank 2...
14	2	Bank 1, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 - 100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
1C	1	OBD standards this vehicle conforms to				
1D	1	Oxygen sensors present				Similar to PID 13, but [A0..A7] == [B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2]
1E	1	Auxiliary input status				A0 == Power Take Off (PTO) status (1 == active) [A1..A7] not used
1F	2	Run time since engine start	0	65,535	seconds	(A*256)+B
20	4	PIDs supported [21 - 40]				
21	2	Distance traveled with malfunction indicator lamp (MIL) on	0	65,535	km	(A*256)+B

id	sz	description	min	max	units	formula
46	1	Ambient air temperature	-40	215	°C	A-40
51	1	Fuel Type				
52	1	Ethanol fuel %	0	100	%	A*100/255
22	2	Fuel Rail Pressure (relative to manifold vacuum)	0	5177.265	kPa	((A*256)+B) * 0.079
23	2	Fuel Rail Pressure (diesel, or gasoline direct inject)	0	655,350	kPa (gauge)	((A*256)+B) * 10
24	4	O2S1_WR_lambda(1): Equivalence Ratio Voltage	0			
2C	1	Commanded EGR	0	100	%	A*100/255
2D	1	EGR Error	-100	99.22	%	(A-128) * 100/128
2E	1	Commanded evaporative purge	0	100	%	A*100/255
2F	1	Fuel Level Input	0	100	%	A*100/255
30	1	# of warm-ups since codes cleared	0	255	N/A	A
31	2	Distance traveled since codes cleared	0	65,535	km	(A*256)+B
32	2	Evap. System Vapor Pressure	-8,192	8,192	Pa	((A*256)+B)/4
33	1	Barometric pressure	0	255	kPa	(Absolute) A
34	4	O2S1_WR_ lambda(1): Equivalence Ratio Current				

id	sz	description	min	max	units	formula
46	1	Ambient air temperature	-40	215	°C	A-40
51	1	Fuel Type				
52	1	Ethanol fuel %	0	100	%	A*100/255
3C	2	Catalyst Temperature Bank 1, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 - 40
45	1	Relative throttle position	0	100	%	A*100/255

Cuadro A.2.: OBDII PIDs

B. AT COMMANDS

B.1. Descripción

El intérprete OBDII ELM327 puede ser configurado mediante los comandos AT. Como la familia de modems, ELM327 reconoce cualquier comando que comienze con AT como un comando de configuración, y la respuesta es el valor solicitado o la palabra “OK” cuando se ejecutó satisfactoriamente el comando.

B.2. Comandos

Los comandos se clasifican en vario tipos:

command	description
<CR>	repeat the last command
BRD hh	try Baud Rate Divisor hh
BRT hh	set Baud Rate Timeout
D	set all to Default
E0,E1	Echo off, or on
FE	Forget Events
I	print the version ID
L0,L1	Linefeeds off, or on
LP	go to Low Power mode
M0,M1	Memory off, or on
RD	Read the stored Data
SD hh	Save Data byte hh
WS	Warm Start (quick software reset)
Z	reset all
@1	display the device description
@2	display the device indentifier
@3ccccccccccc	store the @2 identifier

Cuadro B.1.: Comandos AT Generales

command	description
PP xx OFF	disable Prog Parameter xx
PP FF OFF	all Prog Parameter disabled
PP xx ON	enable Prog Parameter xx
PP FF ON	all Prog Parameters enabled
PP xx SV yy	for PP xx, Set the Value to yy
PPS	print a PP Summary

Cuadro B.2.: Comandos AT de parámetros programables

command	description
AL	Allow Long (>7 bytes) message
ACM	display Activity Monitor Count
AMT hh	set the Activity Mon Timeout to hh
AR	Automatically Receive
AT0,1,2	Adaptative Timeout off, auto 1, auto 2
BD	perform a Buffer Dump
BI	Bypass the initialization sequence
DP	Describe the current Protocol
DPN	Describe the Protocol by Number
H0,H1	Headers off, or on
MA	Monitor All
MR hh	Monitor for Receiver = hh
MT hh	Monitor for Transmitter = hh
NL	Normal Length messages
PC	Protocol Close
R0,R1	Response off, or on
RA hh	set the Receive Address to hh
S0,S1	print of Space off, or on
SH xyz	Set Header to xyz
SH xxyyzz	Set header to xxyyzz
SP h	Set protocol to h and save it
SP Ah	Set protocol to Auto, h and save it
SP 00	Erase stored protocol
SR hh	Set the Receive address to hh
SS	use Standard Search order (J1978)
ST hh	Set Timeout to hh x 4 msec
TA hh	set Tester Address to hh
TP h	Try protocol h
TP Ah	Try protocol h with Auto search

Cuadro B.3.: Comandos AT OBD

command	description
FI	perform a fast Initiation
IB 10	set the Baud rate to 10400
IB 48	set the Baud rate to 4800
IB 96	set the Baud rate to 9600
IIA hh	set ISO (slow) Init Address to hh
KW	display the Key Words
KW0,KW1	Key Word checking off, or on
SI	perform a Slow (5 baud) Initiation
SW hh	Set Wakeup interval to hh x 20 msec
WM	set the Wakeup messages

Cuadro B.4.: Comandos AT Específicos ISO

command	description
CEA	turn off CAN Extended Addressing
CEA hh	use CAN Extended Address hh
CAF0,CAF1	Automatic Formatting off, or on
CF hhh	set the ID Filter to hhh
CF hhhhhhhh	set the ID Filter to hhhhhhhh
CFC0,CFC1	Flow Controls off, or on
CM hh	set the ID Mask to hhh
CM hhhhhhhh	set the ID Mask to hhhhhhhh
CP hh	set CAN Priority to hh (29bit)
CRA	reset the Receive Address filter
CRA hhh	set CAN Receive Address to hhh
CRA hhhhhhhh	set the Rx Address to hhhhhhhh
CS	show the CAN Status counts
CSM0,CSM1	Silent Monitoring off, or on
D0,D1	display of the DLC off, or on
FC SM h	Flow Control, Set the mode to h
FC SM hhh	FC, Set the Header to hhh
FC SH hhhhhhhh	Set the Header to hhhhhhhh
FC SD	FC, Set Data to
PB xx yy	Protocol B options and baud rate
RTR	send a RTR message
V0,V1	use of Variable DLC off, or on

Cuadro B.5.: Comandos AT Específicos CAN

command	description
DM1	monitor for DM1 message
JE	use J1939 Elm data format
JHF0,JHF1	Header Formatting off, or on
JS	use J1939 SAE data format
JTM1	set Timer Multiplier to 1
JTM5	set Timer Multiplier to 5
MP hhhh	Monitor for PGN 0hhhh
MP hhhhhh	Monitor for PGN hhhhhh

Cuadro B.6.: Comandos AT Específicos J1939 CAN

Bibliografía

[1] International Organization for Standardization, "ISO 15765: Road vehicles, Diagnostics on Controller Area Networks (CAN)," 2004.

[2] J. Meseguer, C. Calafate, J.-C. Cano, and P. Manzoni, "DrivingStyles: a smartphone application to assess driver behavior," in *IEEE Symposium on Computers and Communications*, 2013.

[3] Ian Hawkins, "Torque: OBD2 Performance and Diagnostics for your Vehicle. Available: http://torque-bhp.com/," 2014.

[4] "Obd car doctor, available: http://www.incardoc.com/."

[5] "Obd auto doctor, available: http://www.obdautodoctor.com/android/."

[6] International Organization for Standardization, "Obd-ii pids, available: http://en.wikipedia.org/wiki/obd-iipids."

[7] "Ecusim, available: http://www.scantool.net/ecusim-2000.html."

[8] G. Briggs, "Obdsim - simulate an elm327 device, available: http://icculus.org/obdgpslogger/obdsim.html."

[9] "Rs232-obd-sim, available: https://code.google.com/p/rs232-obd-sim/."

[10] A. Shaw, "Automotive obdii simulator," June 2011.

[11] E. E. . C. for the Hobbyist, "Obd to rs232 interpreter," 2013.

[12] SAE International - Vehicle Architecture For Data Communications Standards, "Class B Data Communications Network Interface," 2006.

[13] International Organization for Standardization, "ISO 9141-2:1994/Amd 1:1996," 1996.

[14] International Organization for Standardization, "ISO 14230-1:1999: Road vehicles, Diagnostic systems, Keyword Protocol 2000," 1999.

[15] International Organization for Standardization, "ISO 15031-3:2004: Road vehicles, Communication between vehicle and external equipment for emissions-related diagnostics," 2004.

[16] R. Jazar, *Vehicle Dynamics: Theory and Application*. Springer, 2008.

[17] "Rhumb Line, Available: http://en.wikipedia.org/wiki/Rhumbline/," 2014.

[18] Bluecove Team, "Bluecove. Available at: http://bluecove.org/," 2008.

[19] OpenStreetMap Team, "The JMapViewer Project. Available at: http://wiki.openstreetmap.org/wiki/JMapViewer/," 2014.

[20] OpenStreetMap Team, "The OpenStreetMap Project. Available at: http://www.openstreetmap.org/," 2014.

[21] J. Zaldivar, C. Calafate, J.-C. Cano, and P. Manzoni, "Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pp. 813–819, 2011.

Nomenclatura

η_j	modificador de radio torque en marcha j
ρ	densidad del aire
θ	angulo de pendiente
a_x	aceleración lineal
C_a	coeficiente aerodinámico
C_r	coeficiente de rodadura
F_a	fuerza aerodinámica
F_e	fuerza de motor
F_g	fuerza de gravedad
F_r	fuerza de rozamiento
F_t	fuerza total
g	gravedad
m	masa
n_d	radio de cambio de componentes del sistema de transmisión
n_j	radio de cambio de marcha j
P_e	potencia generada por el motor
P_m	potencia máxima del motor
R_w	radio de los neumáticos
S	superficie frontal del vehículo
v_x	velocidad lineal
W_e	velocidad angular producida por el motor

W_m	velocidad angular máxima
$W_m in$	velocidad angular mínima
BOSS	Blutooth OBDII Simple Simulator
RPM	revoluciones por minuto