



INSTITUTO POLITÉCNICO NACIONAL



UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y
TECNOLOGÍAS AVANZADAS

Ingeniería en Sistemas Automotrices

PROYECTO INTEGRADOR

“PROTOTIPO DE PLATAFORMA PARA
COMPARACIÓN DE DATOS OBTENIDOS
MEDIANTE OBD-II PARA RECOMENDACIONES
DE MANEJO”

Desarrollado por los alumnos

Jiménez Torre Juan Carlos
Ortiz Ruiz Elliot Loyd

Asesores

M.E Carlos Silva Sánchez

Paola Nayeli Cortez Herrera

Ciudad de México, a 7 de enero del 2021



I Agradecimientos y dedicatorias

A mis padres que siempre me han demostrado su apoyo al brindarme la posibilidad de estudiar en una de las escuelas universitarias reconocidas a nivel nacional.

JC

"No que seamos competentes por nosotros mismos para pensar algo como de nosotros mismos, sino que nuestra competencia viene de Dios" 2 Corintios 3:5

A Dios, que por su gracia hoy todos estamos aquí.

A mis segundos padres, Albertina Sánchez Trejo y Carlos Silva Ramírez, que de no ser por ellos no estuviera concluyendo esta etapa de mi vida, a sus consejos, regaños, pero sobre todo por su gran amor incondicional.

A sus hijos, Carlos Silva Sánchez, quien es un gran maestro un gran hermano, pero sobre todo una gran persona, gracias a sus consejos y a su guía. A Adrián Silva Sánchez, Verónica Silva Sánchez porque de igual forma han sido de gran apoyo para mí.

A mis padres que me apoyaron y doy gracias a Dios el aún tenerlos conmigo.

Elliot





Tabla de contenido

I	Agradecimientos y dedicatorias	i
	Tabla de contenido	iii
	Índice de Ilustraciones	vi
	Índice de Tablas.....	viii
II	Resumen	ix
II.1	Resumen del documento	ix
II.2	<i>Abstract</i>	xi
III	Objetivos.....	xiii
III.1	Objetivo general	xiii
III.2	Objetivos particulares	xiii
1	Introducción.....	14
2	Antecedentes.....	17
2.1	Estado del arte	17
2.2	Fundamentos teóricos	18
2.2.1	OBD-II.....	18
2.2.2	Tarjeta de Desarrollo Arduino.....	28
2.2.3	Dispositivos de diagnóstico Automotriz.....	29
2.2.4	Sistema de Geolocalización	29
2.2.5	Arquitectura cliente-servidor.....	30
2.2.6	Cliente	31
2.2.7	Servidor.....	31
2.2.8	Puertos.....	31
2.3	Metodologías.....	32
2.3.1	Extracción de datos.....	34
2.3.2	Envío de paquetes de datos.....	35
2.3.3	Análisis de los datos	36
2.4	Herramientas de trabajo	37
2.4.1	Interfaces de comunicación.....	37



2.4.2	Arduino.....	41
3	Planteamiento del problema	43
3.1	Requerimientos.....	45
3.1.1	Vehículos en México	45
3.2	Propuesta de solución	47
3.2.1	Sistema de monitoreo para recomendaciones finales en la conducción.....	49
3.2.2	Subsistema de transmisión de datos	52
3.2.3	Fuente de alimentación	52
4	Desarrollo de la solución	53
4.1	Subsistema de extracción de datos	54
4.1.2	Módulo de censado.	57
4.1.3	Módulo de almacenamiento.....	58
4.2	Bloque de servicio	58
4.3	Subsistema de recepción.....	59
4.3.1	Módulo de Interfaz hombre-máquina.....	61
5	Validación del sistema	63
5.1	Subsistema de extracción de datos	63
5.1.1	Pruebas de Funcionamiento	63
5.2	Validación de la plataforma.....	66
6	Costos	69
7	Conclusiones.....	71
7.1	CONCLUSIONES	71
7.2	RECOMENDACIONES	72
8	Referencias	73
	Anexo A.....	79
	Anexo B	80
	Anexo C	81
	Apéndice A	85
	Apéndice B.....	107
	Apéndice C	113



Índice de Ilustraciones

Ilustración 1. Componentes del sistema OBD-II	19
Ilustración 2. Aspecto físico de la ECU	20
Ilustración 3. Etapas de funcionamiento ECU	20
Ilustración 4. Funcionamiento del Sensor CKP	21
Ilustración 5. Ubicación de la MIL	22
Ilustración 6. Pines del DLC	23
Ilustración 7. Un mensaje del OBD-II	25
Ilustración 8. Un mensaje CAN OBD-II	25
Ilustración 9. Aspecto físico Arduino.....	28
Ilustración 10. Escáner Automotriz	29
Ilustración 11. Mapa de cobertura móvil [17]	30
Ilustración 12. Estructura de la propuesta de solución	33
Ilustración 13. Formato de Trama de mensaje	34
Ilustración 14. Formato de Trama de mensaje	35
Ilustración 15. Aspecto físico ELM327	38
Ilustración 16. Control de Maestro-Esclavo	39
Ilustración 17. Conexión general Tx-Rx.....	40
Ilustración 18. Aspecto físico SIM900	41
Ilustración 19. Infografía Contaminación del aire, ONU [22]	43
Ilustración 20. Gráfica de vehículos de mayor venta en México	46
Ilustración 21. Estructura de la propuesta de solución	50
Ilustración 22. Proceso de trabajo del subsistema de extracción	51
Ilustración 23. Diagrama Aplicación del proyecto	53
Ilustración 24. Conexión HC-05 y Arduino	54
Ilustración 25. Fragmento del código Comandos AT	55
Ilustración 26. Monitor Serie Arduino	55
Ilustración 27. Captura de pantalla Dispositivos Bluetooth.....	56
Ilustración 28. Fragmento del código Obtención de datos	57
Ilustración 29. Ventana del Puerto Serial	57
Ilustración 30. Comandos para solicitar ver base de datos	58
Ilustración 31. Rutas y Carpetas donde se encuentra guardado el proyecto	59
Ilustración 32. CMD y comandos para activar el entorno virtual	60
Ilustración 33. Editor de texto Sublime Text 2.....	61
Ilustración 34. CMD y comandos para correr el servidor	62
Ilustración 35. Vista principal de la plataforma en la parte administrativa.....	62
Ilustración 36. Conexión de ELM327 a vehículo.....	64



Ilustración 37. Conexión de ELM327 a vehículo.....	64
Ilustración 38. conexión de sistema de transmisión a vehículo	65
Ilustración 39 validación de la plataforma	66
Ilustración 40. mensaje final en la plataforma al momento de terminar la conducción.....	67
Ilustración 41. Comando en CMD para crear el virtualización	81
Ilustración 42. Comprobación de creación exitosa del virtualenv.....	82
Ilustración 43. Activación del entorno virtual.....	83



Índice de Tablas

Tabla 1. Evolución del OBD-II.....	17
Tabla 2. Ejemplo de destellos MIL Chrysler	22
Tabla 3. Descripción del DLC	23
Tabla 4. Tipos de protocolos en OBD-II	24
Tabla 5. Estructura VIN.....	27
Tabla 6. Características Tarjetas Arduino	28
Tabla 7. Especificaciones técnicas del SIM900 GSM/GPRS	41
Tabla 8. Características extendidas de los modelos de Arduino.....	42
Tabla 9. Precio de Vehículos con tecnologías para el ahorro de combustible	44
Tabla 10. Unidades de vehículos vendidos, INEGI [28]	45
Tabla 11. Capacidad volumétrica del motor en los 10 vehículos más vendidos.....	46
Tabla 12. Rendimiento de combustible en los 10 vehículos más vendidos.....	47
Tabla 13. Costos de componentes para la elaboración del proyecto	69
Tabla 14. Métodos o Funciones principales de la librería "elmduino.h" [42]	80

II Resumen

II.1 Resumen del documento

En el presente trabajo se plantea una forma de procesar datos obtenidos mediante el OBD II (por sus siglas en inglés *On Board Diagnostic*) para brindar recomendaciones de manejo al conductor de un vehículo, aprovechando las nuevas tecnologías y gracias a datos obtenidos mediante el sistema OBD II, es posible el almacenamiento y extracción de información en empaquetamiento de datos.

Se emplea el uso de recursos electrónicos: un dispositivo capaz de obtener las magnitudes físicas que cada sensor arroja en tiempo real mediante la conexión DLC (*Data Link Connector*) con la ECU (*Electronic Control Units*), encargada del procesamiento de dichas señales de los sensores. Se utilizan también recursos informáticos como el lenguaje de programación Python y el entorno de trabajo (*Framework*) Django que facilita el desarrollo de una plataforma con una interfaz gráfica de usuario; permitiendo, además, implementar una base de datos donde se almacena la información de un vehículo. Posteriormente se podrán visualizar los resultados a través de un Sitio Web al término de la conducción del vehículo en forma de una calificación puntuada del 5 al 10, con la finalidad de ayudar a cada conductor a mejorar la eficiencia en el consumo de combustible y además evitar la contaminación excesiva de los gases contaminantes emitidos por cada vehículo; lo anterior basado en pruebas de rendimiento realizadas por cada fabricante de vehículos comercializados en México.

Palabras Clave:

OBD II, ECU, Plataforma Web.





II.2 Abstract

This Project explains a way of processing data obtained through OBD II (On Board Diagnostic) is proposed to provide driving recommendations to the driver of a vehicle, taking advantage of new technologies and thanks to data obtained through the OBD II system, it is possible to extract and store information in data packaging.

The use of electronic resources is used: a device capable of obtaining the physical magnitudes that each sensor outputs in real time through the DLC (Data Link Connector) connection with the ECU (Electronic Control Units), in charge of processing these signals from the sensors. It also uses computer resources such as the Python programming language and the Django work environment (Framework) that facilitates the development of a platform with a graphical user interface; allowing, in addition, to implement a database where the information of a vehicle is stored. Subsequently, the results can be viewed through a Website at the end of the vehicle driving in the form of a rating scored from 5 to 10, in order to help each driver improve fuel efficiency and also avoid excessive pollution of polluting gases emitted by each vehicle; the foregoing based on performance tests carried out by each manufacturer of vehicles marketed in Mexico..

Keywords:

OBD II, ECU, Web Platform.



III Objetivos

III.1 Objetivo general

Diseñar una plataforma en una PC local, para procesar datos mediante un algoritmo programado el cual se encargará de comparar los datos obtenidos de los diferentes sensores del automóvil, y al termino, mandará recomendaciones de mejoras para conducir.

III.2 Objetivos particulares

1. Extraer datos del vehículo por medio de un protocolo de comunicación.
2. Analizar datos extraídos y la comprobación de paquetes.
3. Desarrollar un prototipo de una plataforma en una PC local con el lenguaje de programación Python para la comparación de datos extraídos del vehículo.
4. Desarrollar el algoritmo que nos permita hacer una buena comparación de los datos.
5. Comprobar los datos enviados del vehículo con el prototipo de plataforma.
6. Verificar correcto funcionamiento de la plataforma al dar una calificación final al usuario.



1 Introducción

La sociedad actual se encuentra en constante movimiento usando diferentes tipos de transporte y generando así altos niveles de contaminación [1]. Algunas organizaciones ya han establecido medidas para la regulación de gases contaminantes en fuentes móviles [2].

Un factor importante para disminuir la contaminación emitida por los vehículos automotores es mejorar los hábitos de conducción y a la vez se ahorra en combustible. Un estudio que realizó la empresa automotriz Ford en algunos países europeos señaló que los conductores tienen malos hábitos sin darse cuenta; por ejemplo, el no contar con la presión recomendada sobre los neumáticos, reduce hasta un 4% el rendimiento, otro ejemplo es el arranque ahorrador, evitando las aceleraciones bruscas, se puede ahorrar hasta 11% del consumo de combustible [3].

Las innovaciones tecnológicas aplicadas en los vehículos han permitido mejorar el rendimiento y comodidad. Una de estas mejoras es la implementación del sistema OBD II, gracias a él se puede conocer en tiempo real el estado actual del automóvil para verificar su correcto funcionamiento.

Utilizando el puerto DLC hacia la ECU (Unidad de Control Electrónico), se pueden extraer los datos necesarios, posteriormente almacenarlos en una base de datos y por medio de un algoritmo comparar dichos datos para dar una respuesta al usuario buscando la mejora en la conducción; los datos mostrados al usuario serán en forma de una calificación puntuada del 5 al 10 donde 10 será una calificación excelente, según el rendimiento que proporciona el motor en base a las pruebas de cada fabricante de vehículos.

El presente escrito está dividido en 7 capítulos.

- Capítulo 1: Introducción.
- Capítulo 2: Muestra los proyectos que tienen relación con éste y que han sido desarrollado. Describe los fundamentos del sistema encargado de la gestión de señales de entradas y salidas proporcionados por los sensores y actuadores respectivamente.



- Capítulo 3: Introduce los requerimientos necesarios para llevar a cabo el desarrollo del proyecto, tanto en hardware como en software. Detalla la función de la librería “*elmduino.h*”, indispensable para la obtención de datos, también hay una explicación del *framework* Django para la realización de la plataforma Web.
- Capítulo 4: Cuenta con varias subsecciones detallando cada una de las etapas que conforman el proyecto, así como la puesta en marcha.
- Capítulo 5: Muestra los resultados de todo el proceso que se realizó además de explicar posibles fallas y las soluciones.
- Capítulo 6: Conclusiones sobre el alcance e impacto del trabajo, así como de mejoras en el futuro.
- Capítulo 7: Referencias bibliográficas, anexos y apéndices proporcionando a detalle el código empleado.



2 Antecedentes

2.1 Estado del arte

La historia del desarrollo de un sistema capaz de gestionar los niveles de gases contaminantes ha sido introducida en los vehículos principalmente por agencias americanas, una de las más sobresalientes es la Junta de Recursos del Aire de California (CARB por sus siglas en inglés), en la siguiente tabla se muestra la evolución en este sistema que finalmente se denominó OBD-II, desde su creación hasta la actualidad.

No.	NOMBRE	CARACTERISTICAS	PAÍS	INSTITUTO	TIPO	REFERENCIA
1	Computerized Assembly Line Diagnostic Link (ALDL)	Lee códigos de error a una velocidad de 160 baudios	E.U.	General Motors	Producto (1969)	[4]
2	Computerized Assembly Line Diagnostic Link (ALDL)	La velocidad pasó a 8192 baudios con el receptor / transmisor asíncrono universal (UART)	E.U.	General Motors	Producto (1986)	
3	OBD	El conductor detecta un mal funcionamiento, se impuso la obligación de tener una lámpara que indique los fallos (MIL)	E.U.	California Air Resources Board (CARB)	Proyecto (1989)	[5]
4	OBD II	El conector y los protocolos recomendados de SAE fueron incorporados en la regulación.	E.U.	California Air Resources Board (CARB)	Proyecto (1996)	[4]

Tabla 1. Evolución del OBD-II

La introducción de este sistema de monitoreo de gases contaminantes dio un giro a la industria automotriz, debido a que paulatinamente fue acrecentado el número de componentes electrónicos, principalmente en el motor para la mejora en la mezcla aire-combustible.

2.2 Fundamentos teóricos

La contaminación ambiental es un tema que desde hace algunas décadas la Junta de Recursos del Aire de California determinó que es necesario que todos los automóviles de combustión estén regulados en el control de la emisión de gases producidos por estos al ambiente [6].

El sistema desarrollado fue llamado OBD (*On Board Diagnostic*) y a partir de 1988 todos los vehículos fabricados en Estados Unidos, debían contar con dicho sistema [7]. No obstante, el problema surgió debido a que cada fabricante de automóviles proponía un tipo de conector; al ser usado en un vehículo, no era compatible con otro. La solución fue gracias a la segunda generación del OBD denominado simplemente como OBD-II, que mediante las normas ISO-9141, ISO 14230, SAE-J1850 VPW, SAE-J1850 PWM, e ISO 1725 CAN BUS, proporciona la ubicación concreta de cada pin del conector y la ubicación dentro del habitáculo en el vehículo [6].

2.2.1 OBD-II

El OBD-II se empezó a utilizar a partir del año de 1996 de forma obligatoria en Estados Unidos [2]. Además de regular los gases contaminantes, permite la lectura de los sensores en tiempo real del vehículo, como ejemplo están: las RPM (Revoluciones Por Minuto), velocidad, nivel de combustible.

Una luz de advertencia es mostrada sobre el tablero cuando el sistema OBD-II detecta una anomalía, al mismo tiempo, el o los códigos de falla son almacenados para mostrarlos posteriormente al técnico automotriz y facilitar la corrección del mal funcionamiento [8].

2.2.1.1 Componentes del Sistema OBD-II

A continuación, se muestran los componentes del sistema OBD-II de forma simplificada:

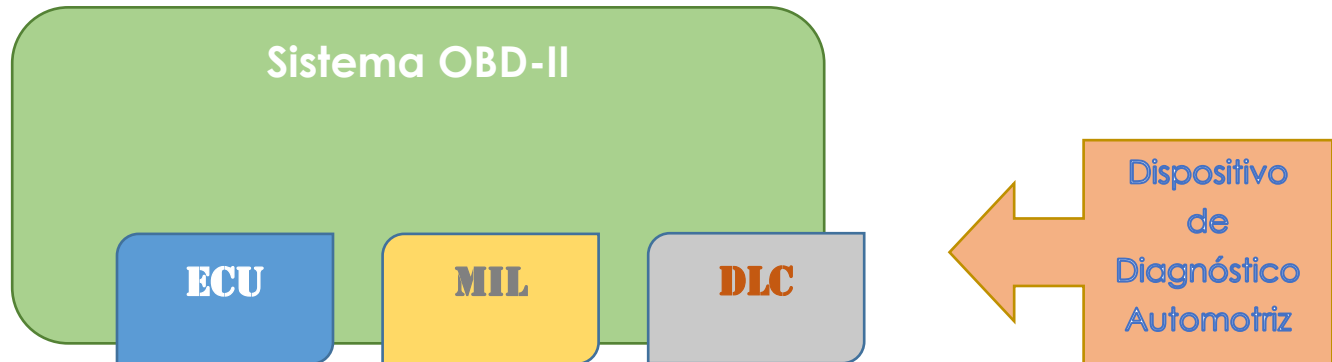


Ilustración 1. Componentes del sistema OBD-II

Un elemento indispensable para alerta y diagnóstico de códigos de fallas es la Luz Indicadora de Malfuncionamiento Mil. En la sección 2.2.1.1.3 se detallarán las características y funciones.

2.2.1.1.1 ECU

El OBD-II realiza el diagnóstico de cada uno de los sensores conectados a la Unidad Electrónica de Control (ECU). La ECU se encarga de comparar los valores establecidos por el fabricante almacenados en su memoria flash, con los valores arrojados por cada sensor, principalmente dentro del monoblock.

La forma física de la ECU es en forma de una caja sellada que generalmente se ubica debajo del tablero de instrumentos en la parte del conductor. Debe soportar el ruido eléctrico y electromagnético que genera el alternador estando en marcha, esta es una característica fundamental para el correcto funcionamiento de la ECU. [9]



Ilustración 2. Aspecto físico de la ECU

El funcionamiento de la ECU se divide en 3 etapas que a continuación se enumeran.

1. Entrada: Recibe señales de tipo analógico y/o digital provenientes de cada sensor. El ADC (Convertidor Analógico-Digital) y acondicionador de señales también conforman esta etapa.
2. Procesamiento: Se encarga de comparar los datos, para esto, utiliza procesadores, unidades de almacenamiento, unidades aritméticas, señales de sincronización.
3. Salida: Los llamados actuadores trabajan gracias a las instrucciones que reciben de la ECU, interviniendo en los tiempos exactos.

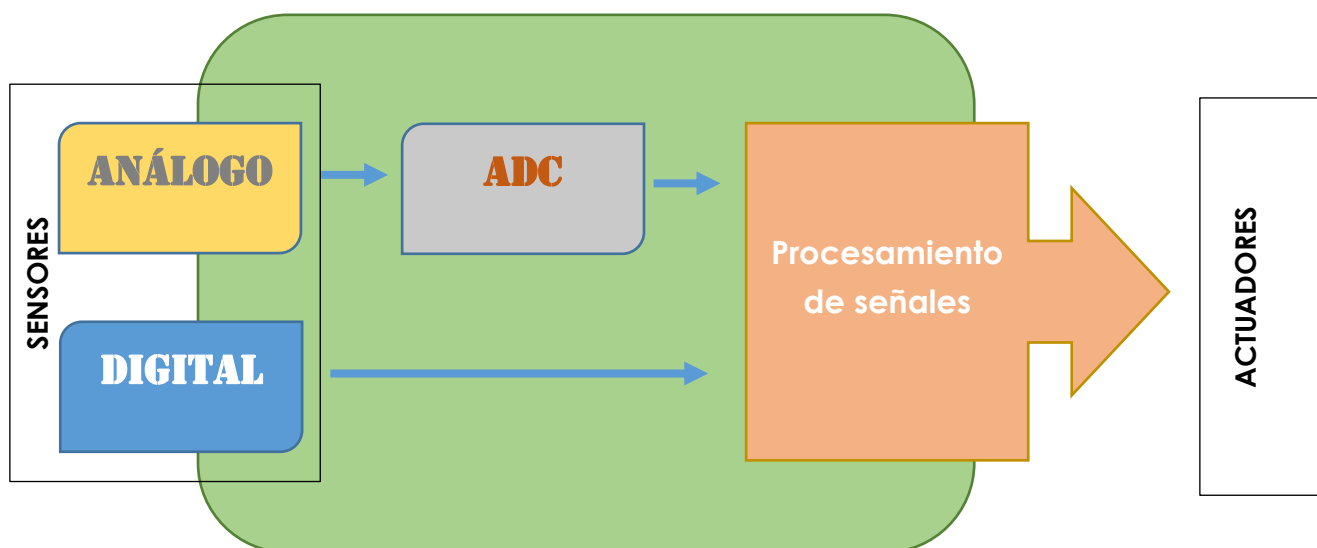


Ilustración 3. Etapas de funcionamiento ECU

Los microprocesadores en general, como el de las PC (*Personal Computer*), son capaces únicamente de procesar señales digitales, por lo que es necesario el uso de convertidores de señal, en el caso de la ECU, para que se puedan interpretar de forma correcta las lecturas captadas por los sensores del vehículo.

2.2.1.1.2 Sensores

Los sensores son dispositivos que captan algún fenómeno físico y lo transforman en una señal eléctrica. Tome como ejemplo el sensor CKP (*Crankshaft Position*) que mide las RPM del cigüeñal.

Cuando en un conductor eléctrico circula una corriente, aparece un campo magnético y también puede ser, al contrario; en presencia de un campo magnético existe la aparición de un campo eléctrico por separación de cargas en el interior de un conductor eléctrico por el que circula una corriente perpendicular al movimiento de las cargas [10].

Todo vehículo cuenta con una rueda dentada en la parte inferior del monoblock. Allí, exactamente, al lado de esta va alojado el sensor: un imán y una bobina. Cada vez que la rueda dentada gira, la bobina capta el campo magnético del imán y emite una señal de voltaje [11].

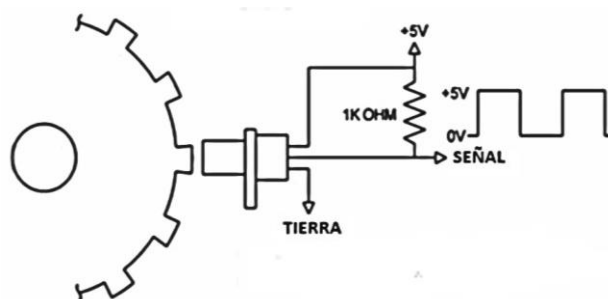


Ilustración 4. Funcionamiento del Sensor CKP

En el caso del sensor CKP, envía una señal digital, pues solo varía de 0 a 5 V en sus extremos, sin pasar por valores intermedios; por el contrario, un sensor ECT (*Engine Coolant Temperature*) es un ejemplo de sensor análogo.

2.2.1.1.3 Luz Indicadora de Fallas MIL

La función principal de la Luz Indicadora de Mal funcionamiento es para alertar al conductor de una posible falla en el motor. La ubicación de la MIL se encuentra en el tablero de instrumentos. Posee un color ámbar.



Ilustración 5. Ubicación de la MIL

Cada vez que el sistema detecta una falla en algún sensor, enciende la MIL y la mantiene encendida para que el propietario del vehículo lleve al servicio lo más pronto posible.

Para entrar en el modo de diagnóstico se debe abrir y cerrar el *switch* de encendido 3 veces en menos de 5 segundos finalizando en la posición ON. La Luz empezará a destellar, primero de forma lenta, hará una pausa y en seguida destellará rápidamente.

Destellos largos	Destellos cortos	Descripción
1	1	Falla en señal de referencia.
2	4	Falla en el circuito del TPS.
5	5	Fin del mensaje.

Tabla 2. Ejemplo de destellos MIL Chrysler

Cada fabricante de vehículos cuenta con su propia descripción de códigos de falla en la Luz Indicadora de Malfuncionamiento.

2.2.1.1.4 Conector de Enlace de Datos DLC

El conector de enlace de datos es el puerto de conexión con 16 pines múltiples; se utiliza para conectar equipos de diagnóstico. Está basado en el estándar SAE J1962, se ubica debajo del tablero de instrumentos.

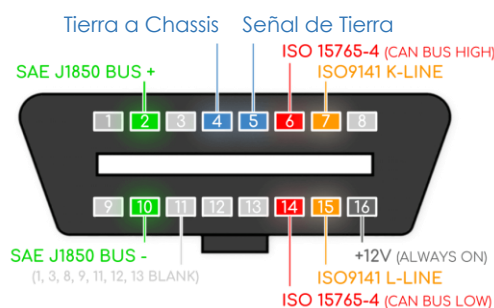


Ilustración 6. Pines del DLC

PIN	Descripción [12]
1	Uso del fabricante
2	Bus (+) J1850 VPM (Variable Pulse Modulation) y PWM (Pulse Width Modulation)
3	Uso del fabricante
4	Tierra (chasis)
5	Señal de tierra
6	Bus de datos CAN alto (J-2284)
7	Línea K ISO 9141-2
8	Uso del fabricante
9	Uso del fabricante
10	Bus (-) J1850
11	Uso del fabricante
12	Uso del fabricante
13	Uso del fabricante
14	Bus de datos CAN (Controller Area Network) bajo (J-2284)
15	Línea L ISO 9141-2 16
16	Voltaje de batería

Tabla 3. Descripción del DLC

Aunque la forma del conector es la misma para todos vehículos, cada fabricante decide qué pines utilizar de acuerdo con el protocolo de comunicación que elija.

2.2.1.2 Protocolos de comunicación

Los protocolos de comunicación del OBD-II permiten establecer la comunicación e intercambio de mensajes en forma bidireccional entre el dispositivo de diagnóstico y la ECU.

La tabla siguiente muestra la comparativa entre los cinco protocolos de comunicación soportados por el OBD-II [12], [13].

PROTOCOLO	FABRICANTE	COMUNICACION	VELOCIDAD TRANSMISION	VOLTAJE
SAE J1850 PWM	Ford, Lincoln y Mercury	Modulación de ancho de pulso (PWM)	41.6 Kbaud/s	0 — 5 V en modo diferencial
SAE J1850 VPN	General Motors	Modulación de ancho de pulso variable (VPN)	10.4 — 41.6 Kbaud/s	2.2 V — OL 8 V — 1L
ISO 9141-2	Fabricantes europeos, asiáticos, Chrysler, Jeep y Dodge	Comunicación similar al estándar RS-232	10.4 Kbaud/s	0 — 12 V (se ajustan al voltaje de la batería)
ISO 14230 KWP (Key Word Protocol)	Fabricantes europeos y asiáticos	Comunicación similar al estándar RS-232	1.2 — 10.4 Kbaud/s	0 — 12 V (se ajustan al voltaje de la batería)
ISO 15765 CAN	Compañía Bosch	Red de Área del controlador	250 — 500 Kbps	2.5 — 5 V (CANH) 2.5 — 0 v (CANL)

Tabla 4. Tipos de protocolos en OBD-II

Cada equipo de diagnóstico automotriz se encarga de enlazarse con cualquier protocolo de comunicación para brindar el servicio de detección de códigos de falla y entre otros parámetros que se mostrarán con detalle en las siguientes subsecciones.

2.2.1.3 Mensajes del OBD-II

Los protocolos J1850 e ISO tienen el mismo formato en la trama del mensaje: 3 bytes de cabecera, 7 bytes máximo de datos y 2 bytes de comprobación de errores.

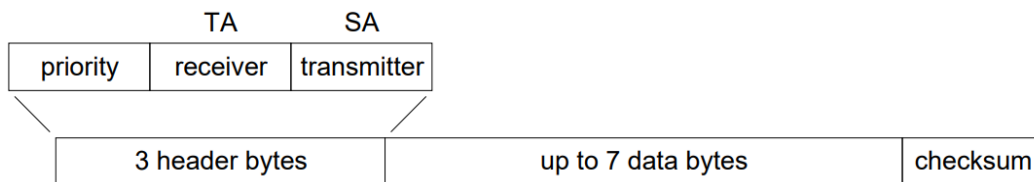


Ilustración 7. Un mensaje del OBD-II

Los parámetros que conforman un mensaje son [14]:

- PRIORIDAD: Indica el nivel de importancia para ejecutar determinada acción.
- (TA) *TARGET ADDRESS*: Dirección destino del mensaje.
- (SA) *SOURCE ADDRESS*: Dirección de procedencia o de origen.
- DATOS: Pueden ser un parámetro o una petición.
- *CHECKSUM*: Comprueba si hubo errores durante la transmisión, en caso de error, pide el reenvío de este.

El protocolo CAN (*Control Area Network*) es parecido a los anteriores, la diferencia está en el tamaño de los bytes de cabecera o de identificación ID, 11 o 29 bits; el primero se conoce como CAN estándar y el segundo como CAN extendido.

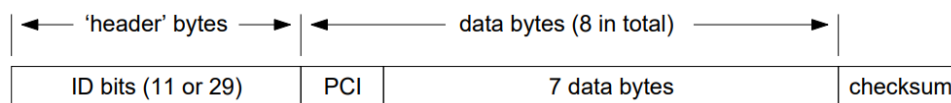


Ilustración 8. Un mensaje CAN OBD-II

Los parámetros que conforman un mensaje CAN son [14]:

- ID bits: Indican el tipo de mensaje, su prioridad, y dirección origen y destino.
- PCI: El Protocolo de Control de Información proporciona el tamaño de la trama de datos que por lo regular es de 7 bytes.
- Datos y *CHECKSUM*: Cumplen la misma función que en los protocolos antes mencionado.

2.2.1.4 Modos de medición

Para obtener datos de la ECU, el sistema OBD-II cuenta con nueve modos diferentes, cada modo permite acceder a ciertos parámetros del sistema. Se requiere de un parámetro de identificación PID que es mandado por el dispositivo de diagnóstico automotriz. A modo de ejemplo, si el usuario quiere saber las RPM (Rotaciones por Minuto) del cigüeñal, primero debe indicar en el dispositivo de diagnóstico que quiere datos en tiempo real, pertenecen al modo 1, después presionará en el botón de RPM, de esta forma estará enviando el PID 0x0C y obtendrá como respuesta el valor numérico de las RPM actuales. Existe una lista bastante extensa de los PID, esta lista se encuentra en el [Apéndice A](#).

A continuación, se explicarán brevemente los 9 modos de medición; solo se puede acceder a ellos con un dispositivo de diagnóstico profesional [15].

- Modo 1: Datos en tiempo real

Es posible acceder a todos los sensores del vehículo, obteniendo las medidas físicas reales en el momento exacto que se solicitan.

- Modo 2: Datos congelados

Cuando ocurre una falla, los datos de los sensores son almacenados; es similar al modo 1, no obstante, son datos pasados, no en tiempo real.

- Modo 3: Códigos de falla

Como se ha explicado anteriormente, cuando la MIL enciende, el código de falla DTC (*Diagnostic Trouble Codes*) es almacenado en la memoria para ser mostrado al técnico automotriz. En caso de no encontrarse códigos de falla, recibe una respuesta nula.

- **Modo 4: Borrar**
Al entrar en este modo, de alguna forma se puede escribir sobre la ECU, ya que nos permite el borrado de códigos de falla, apagando la MIL del tablero de instrumentos.
- **Modo 5: Sensores de oxígeno**
Permite la lectura de los resultados obtenidos por los sensores de oxígeno para determinar la eficiencia del convertidor catalítico.
- **Modo 6: Resultados de pruebas**
Permite el acceso del resultado de monitoreo continuo y no continuo. Está programado por cada fabricante automotriz y varía de una marca a otra.
- **Modo 7: DTC**
Modo lectura de los DTC pendientes almacenados en la ECU.
- **Modo 8: Prueba de actuadores**
El técnico tiene la opción de activar/desactivar los actuadores tal como bombas de combustible, válvula de ralentí, entre otros.
- **Modo 9: Información del automóvil**
La información que se solicita es el VIN, que consiste en 17 caracteres alfanuméricos (excepto I, O y Q).

CARACTER	DESCRIPCION
1	- País de fabricación
2	- Fabricante
3	- Tipo de automóvil o división de fabrica
4 al 8	- Características del automóvil
9	- Dígito de control
10	- Año del modelo
11	- Planta de ensamblaje
12 al 17	- Secuencia de producción

Tabla 5. Estructura VIN

La razón del por qué se pueden acceder a todos los modos de medición solo con equipos profesionales se explicará en la sección 2.2.3 Dispositivos de diagnóstico automotriz.

2.2.2 Tarjeta de Desarrollo Arduino

Arduino es una empresa de software y hardware libre, lo que significa que cada persona puede modificar el código escrito por otros usuarios y utilizarlo para su conveniencia; se basa en la arquitectura AVR de los microprocesadores ATMEL, empezaron a comercializarse en el año 2012 [16].

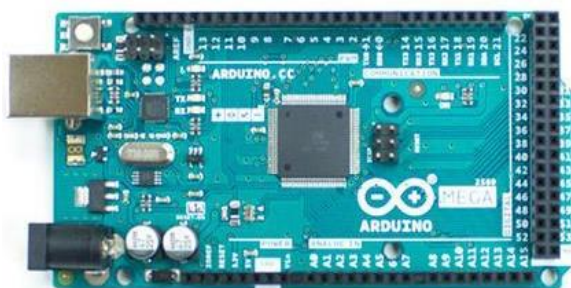


Ilustración 9. Aspecto físico Arduino

A diferencia de otras tarjetas de desarrollo como la Raspberry Pi que cuenta con un único modelo, el Arduino incorpora diversos microprocesadores con características que van de acuerdo con cada necesidad. En la tabla siguiente se muestran sus características principales.

Características	ARDUINO UNO	ARDUINO LEONARDO	ARDUINO MEGA 2560
Microprocesador	ATmega328	ATmega32u4	ATmega2560
Pines digitales I/O	14	20	54
Pines analógicos de entrada	6	12	16
Memoria Flash	32KB	32KB	256KB

Tabla 6. Características Tarjetas Arduino

Gracias a la aceptación por la mayoría de sus usuarios y a las grandes ventas que ha tenido, las tarjetas Arduino son fáciles de adquirir y a un costo accesible.

2.2.3 Dispositivos de diagnóstico Automotriz

En el mercado existen distintos tipos de dispositivos para el diagnóstico automotriz, la diferencia se basa en el alcance que se lograr con éstos, es decir, algunos escáneres permiten la lectura y escritura de la información establecida en la ECU por los fabricantes como es el caso del VIN (Número de Identificación Vehicular). Además, permiten leer los códigos genéricos y específicos de cada marca automotriz para su posterior reparación. La mayoría de ellos cuentan con una pantalla para poder visualizar toda esa información.



Ilustración 10. Escáner Automotriz

Comúnmente llamado escáner automotriz, cuenta con varias versiones dependiendo del uso desinado para ello. Existen para marcas específicas dividiéndose en asiáticas, europeas o americanas, y otro tipo son los multimarca, que abarca un número mayor de modelos existentes en el parque vehicular.

2.2.4 Sistema de Geolocalización

La Red GPRS significa *General Packet Radio Service* (servicio general de paquetes vía radio), proporcionada por los servicios de telefonía móvil, esta es la sucesora del GSM

(Global System for Mobile communications, sistema global para las comunicaciones móviles).

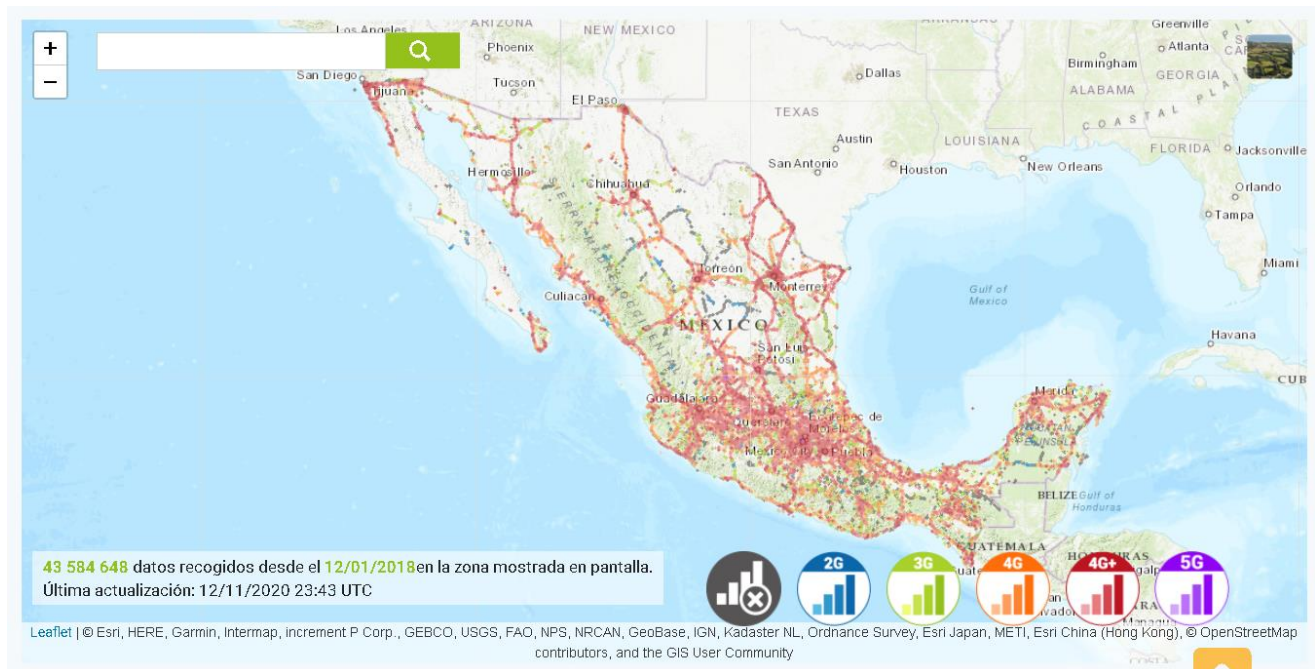


Ilustración 11. Mapa de cobertura móvil [17]

Esta tecnología de comunicaciones móviles se encuentra disponible en más del 82% de las terminales mundiales [18], por lo tanto, se tendrá mayor cobertura para el envío de los datos extraídos del vehículo. Otro factor importante para la implementación de esta Red es que permite el envío de SMS (Short Message Service) a una “velocidad de transferencia de entre 56 a 114 kbps (kilobits por segundo), lo que significa enviar 30 SMS por minuto, mientras que con GSM podemos mandar entre 6 y 10” [18].

2.2.5 Arquitectura cliente-servidor

Para esta parte del proyecto, debemos tener en cuenta lo que se trata la parte de cliente-servidor.

La arquitectura cliente-servidor se trata de un modelo de aplicación distribuida donde las tareas se reparten entre los proveedores de servicios (servidor) y los demandantes (clientes).



A la red de comunicaciones e la cual varios clientes se conectan a un servidor se le conoce como red cliente-servidor, donde los servicios se encuentran a disposición d ellos.

2.2.6 Cliente

El cliente es un elemento activo que envía una solicitud al servidor mediante su dirección IP y un número de puerto asociado al servicio por el que pide. Sus características con:

- ° Enviar solicitudes al servidor.
- ° Esperar y recibir respuestas del servidor.
- ° Conectarse a varios servidores a la vez.
- ° Interactuar directamente con los usuarios finales.

2.2.7 Servidor

El servidor es un elemento pasivo que se encarga de atender los requerimientos de múltiples clientes. Sus características son las siguientes.

- ° Esperar a que lleguen solicitudes de los clientes.
- ° Receptar la solicitud, procesarla y luego enviar una respuesta al cliente.
- ° Aceptar conexiones de un gran número de clientes.
- ° No interactuar directamente con los usuarios finales.

2.2.8 Puertos

Un puerto es una interfaz que permite comunicar una aplicación cliente con una aplicación servidor sobre una red. El puerto identifica a la aplicación que lo usa, y se representa con 16 bits.

2.3 Metodologías

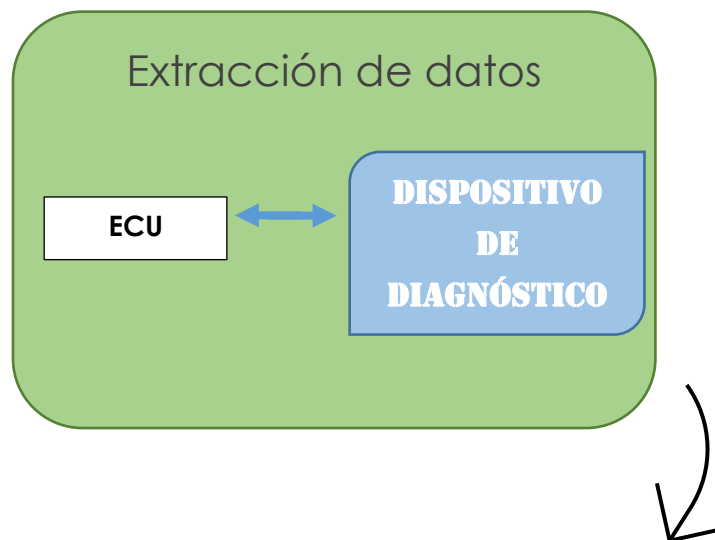
Las etapas para el desarrollo de este proyecto se basan en 3 subsistemas: extracción de datos, transmisor y receptor.

La extracción de datos se obtendrá mediante dispositivos electrónicos con un precio muy accesible para cualquier persona.

La transmisión se llevará a cabo mediante la utilización una tarjeta SIM de la compañía TELCEL, aunque podría utilizarse la de cualquier otra compañía.

La recepción, junto con la plataforma Online trabajarán en conjunto para el análisis de los paquetes de datos obtenidos. El usuario registrado en la plataforma Online contará con un ID por cada vehículo que cuente con el sistema de recomendaciones de manejo instalado. Al contar con acceso a Internet, se podrá observar de manera visual la gráfica de calificaciones en el manejo de acuerdo con los datos almacenados previamente en una base de datos.

Cada subsistema cumple con una tarea específica que a continuación se explica.



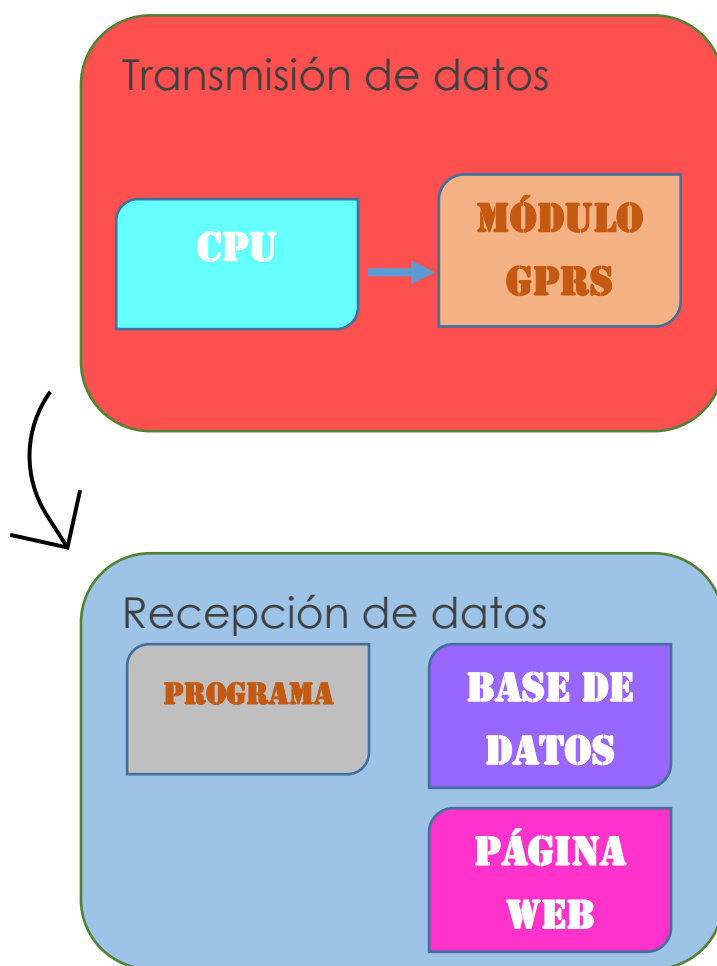


Ilustración 12. Estructura de la propuesta de solución

La comunicación entre subsistema de transmisión y recepción de datos será por medio de SMS gracias al módulo GPRS conectado a cualquier operadora de telefonía móvil.

2.3.1 Extracción de datos

El dispositivo de diagnóstico es el encargado de establecer la comunicación principal entre la ECU y nuestro Arduino mediante el DLC para la lectura de medidas arrojadas por cada sensor a través de PID del modo 1.

La comunicación será enlazada de forma inalámbrica con la tecnología Bluetooth®, las ventajas son:

- Conexión simple, sin cables
- Posicionar la tarjeta Arduino en cualquier parte del habitáculo.
- Presentación.

Una vez configurado en modo 1 para la extracción de datos en tiempo real, se solicitarán los valores mediante PID, así la ECU enviará una respuesta con los datos requeridos. La tarjeta Arduino se encargará de leer estos datos y los guardará temporalmente en un arreglo para enviarlos posteriormente a la base de datos mediante la SIM900 a través de la Red GPRS.

Los datos enviados cuentan con los siguientes parámetros:

- Un ID por vehículo registrado en la plataforma Online
- Los valores obtenidos de cada sensor.

Los parámetros anteriores, aunque son básicos, brindan lo necesario para el monitoreo del vehículo y además para evitar la saturación de la base de datos.



Ilustración 13. Formato de Trama de mensaje

A continuación, se presenta en detalle el significado de cada parámetro.

- ID: Número de identificación de cada vehículo que el propietario haya registrado en la base de datos. Permite llevar un orden y evitar errores al analizar datos, que sean correspondiente solo a la base de datos correcta.
- Valores obtenidos: Son los valores externos de la ECU correspondiente a los 4 sensores elementales para el monitoreo del rendimiento de combustible.

Con los datos obtenidos en tiempo real de los sensores CKP para las RPM, VSS para la velocidad, de acuerdo con lo descrito en la sección 3.2. se almacenarán en un arreglo bidimensional de 15x4. Se pedirá cada segundo el valor de un sensor distinto; esta operación se realizará 15 veces para que al final el arreglo sea enviado minuto a minuto. A este arreglo se le conoce como empaquetamiento.

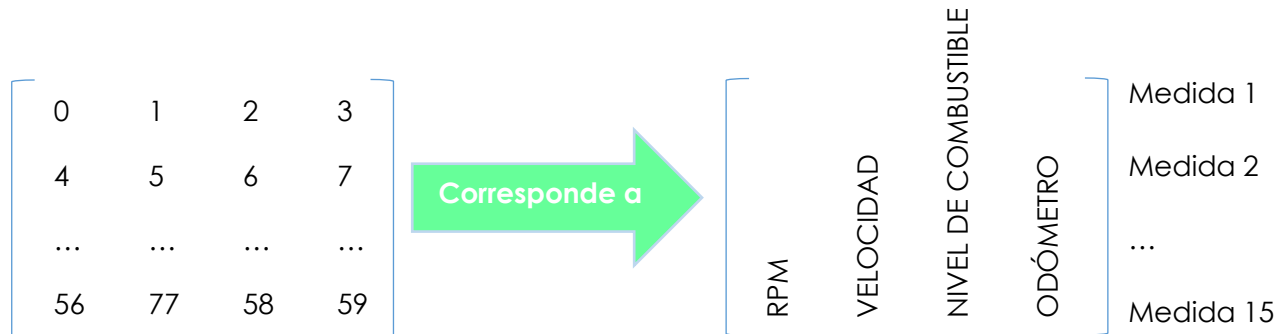


Ilustración 14. Formato de Trama de mensaje

La eficiencia en el sistema depende de la velocidad con la que almacena los datos y el tiempo de espera entre el envío de un paquete y otro, evitando saturar la base de datos y el programa con el algoritmo encargado del análisis para las recomendaciones.

2.3.2 Envío de paquetes de datos

La transmisión de cada empaquetamiento se realiza a través de la red GPRS con la implementación de un módulo, conectado con la misma tarjeta de desarrollo Arduino.

Las funciones que desempeñará son:

- Establecer la comunicación con el módulo de Red GPRS para el correcto intercambio de datos de forma inalámbrica.
- Armar la estructura del mensaje que contiene el empaquetamiento.
- Enviar los paquetes de datos minuto a minuto.

2.3.3 Análisis de los datos

Para poder analizar esta información que se obtiene mediante la ECU del automóvil por medio del dispositivo ya implementado, es necesario el descifrar que empaquetamiento pertenece a cada sensor del automóvil, ya que debido al modo 1 de extracción de información, solo se solicita los datos de velocidad y RMP del vehículo.

Por lo que en este proceso el lector ELM327 se configura en modo1, para que así podamos recibir desde la Ecu una respuesta sobre los datos solicitados. Después, el dispositivo Arduino se encarga de leer, a través del envío PIDs y posterior recepción de una respuesta, y procesar adecuadamente los datos recibidos desde la ECU. Este proceso se repite de acuerdo con el número de medidas solicitadas por el dispositivo Arduino.

El dispositivo Arduino Mega crea y envía el mensaje hacia el servidor remoto. Este mensaje está conformado por los siguientes campos: un identificador del modo de medición utilizado (modo 1) y los valores medidos por los sensores del motor.

Aunque el diseño del mensaje tiene un formato básico, entrega información suficiente para el monitoreo del automóvil.

2.4 Herramientas de trabajo

La viabilidad del proyecto se basa en las siguientes características.

- Es compatible con la mayoría de los vehículos a partir del año 2000.
- Soportará la lectura de PID en el modo 1 que arroje cada sensor; para el presente proyecto son las RPM, la velocidad, el nivel de combustible y el odómetro.
- Para la transmisión de los datos se utilizará la tecnología GPRS con ayuda de una operadora móvil de México.
- Almacenará las calificaciones finales obtenidas por el conductor para su posterior consulta y mejora.
- Se obtendrán los resultados mostrados en una página Web desde cualquier dispositivo conectado a Internet.

En otras palabras, los dispositivos electrónicos que se pretenden implementar son de bajo costo, de tal modo cualquier persona las puede adquirir.

2.4.1 Interfaces de comunicación

Una interfaz de comunicación permite la transmisión y recepción de datos entre dispositivos, siempre que cuenten con los mismos parámetros configurados como el tipo de conexión y la velocidad.

Dentro de esta subsección, se definirán los dispositivos electrónicos de acuerdo con sus prestaciones y el motivo de la implementación. Para la correcta elección de dispositivos electrónicos, se necesita conocer la conexión de cada uno de ellos. Por lo anterior, se analizarán las interfaces de comunicación.

a) ECU – ELM327

Esta interfaz de comunicación se realiza mediante el protocolo CAN; ya se ha mencionado que existen dos tipos: CAN estándar o CAN extendido, la diferencia entre uno y otro es la cantidad de bits transmitidos: 11 o 29 respectivamente.

El intercambio de datos es mediante 2 cables y otro conectado a tierra, este último sirve para comparar los valores en Volts que existe en los cables de transmisión-recepción, ya que incluso, si uno de los dos llega a fallar, el otro sigue con la operación y el sistema funciona sin mayores complicaciones. Estos cables son llamados CAN HIGH y CAN LOW, sus terminales están unidas a un conector hembra DLC.

Con lo ya visto en sección 2.2.1.4 el modo 1 permite extraer datos en tiempo real; para la aplicación que se pretende desarrollar es suficiente con entrar en este modo.

Existe un dispositivo de diagnóstico automotriz que utiliza un chip integrado llamado ELM327, de allí proviene el nombre del mismo dispositivo, este prescinde de pantalla y botones, convirtiéndolo en un dispositivo de diagnóstico automotriz de muy bajo costo. La visualización y manipulación se realiza mediante un teléfono o tableta electrónica con conexión vía Bluetooth® y una App descargable desde cualquier tienda virtual.



Ilustración 15. Aspecto físico ELM327

El bajo costo, la sencillez de uso y el código abierto del dispositivo ELM327 lo convierte en el dispositivo ideal para la aplicación que se le dará en el presente proyecto. Encontrará las características técnicas en la hoja de datos incluida en el [Apéndice B](#).

El dispositivo ELM327 cuenta con un conector macho, permitiendo la conexión con el DLC.

b) ELM327 – ARDUINO

Esta interfaz es la encargada de la solicitud de los PID correspondiente a cada sensor que requiere la aplicación presente.

La comunicación se realizará de forma inalámbrica, aprovechando el módulo Bluetooth ® incluido en el ELM327. Esto facilitará el emparejamiento entre ambos dispositivos ahorrando tiempo cada vez que se inicie la conducción del vehículo.

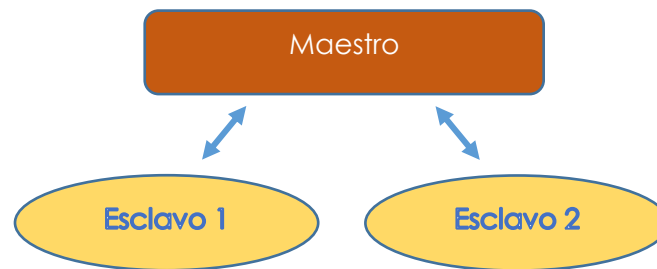


Ilustración 16. Control de Maestro-Escavo

El dispositivo que se configura como maestro dicta las "órdenes" en el tiempo que lo desee y el esclavo responde; puede haber varios esclavos conectados a un maestro.

En el mercado existen dos tipos de módulos con la tecnología Bluetooth: HC-05 y HC-06; la diferencia entre estos modelos es que el primero se puede configurar como maestro o como esclavo, para el presente proyecto, se requiere de un módulo maestro, como ya se ha mencionado, dictará las órdenes en el tiempo exacto

El módulo HC-05 (Bluetooth ® maestro) se conectará a la tarjeta Arduino por medio del puerto serial Tx-Rx.

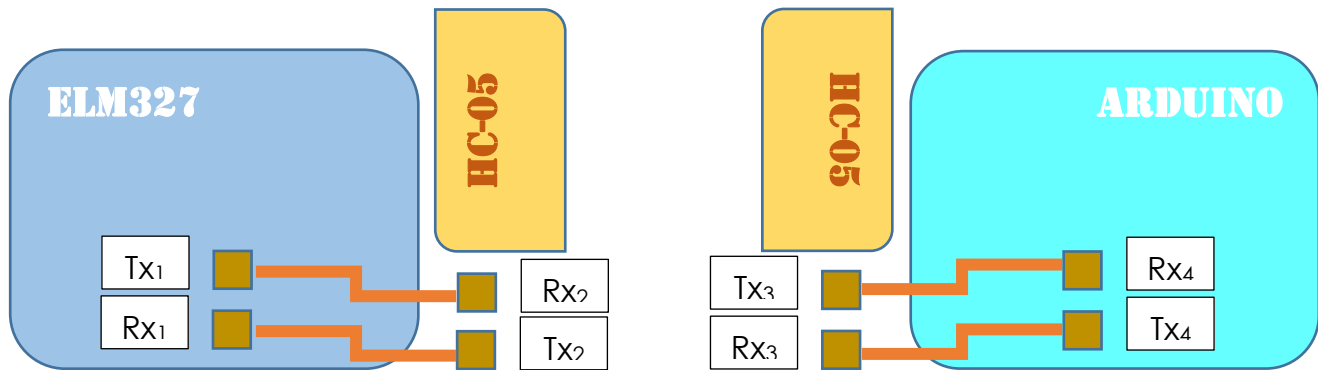


Ilustración 17. Conexión general Tx-Rx

La conexión se hace de forma cruzada, donde Tx₁ y Rx₁ pertenecen al dispositivo 1; se refiere a la transmisión y recepción respectivamente. Los pines Rx₄ y Tx₄ son un puerto UART (*Universal Asynchronous Receiver-Transmitter*) del Arduino destinado a la conexión con el módulo HC-05.

c) ARDUINO – SIM900

Al igual que con el Bluetooth® visto en el inciso anterior, la conexión se efectuará por medio del puerto serial UART de la tarjeta Arduino. Este es otro par de pines que se tiene que tomar en cuenta para la elección del modelo de placa correcto. Esta interfaz de comunicación permitirá el envío de los paquetes de datos previamente almacenados en memoria de la tarjeta Arduino.

La plataforma con interfaz gráfica para el usuario estará disponible por medio de una página web; para que los datos sean mostrados, es preciso utilizar la red GPRS.

El vehículo de cualquier usuario recorrerá diversos lugares con distintos tipos de tecnologías de red móvil, por lo tanto, es importante contar con un módulo con la red 4G, que hasta el momento es la más reciente en México [19]. Así como las tarjetas Arduino cuentan con varias versiones, existen al menos 3 diferentes módulos con el integrado capaz de conectarse a la red GSM: SIM800, SIM 900 y A6 GPRS GSM. El único módulo capaz de soportar la red 4G es el SIM900, por lo tanto, es el módulo que se implementará.



Ilustración 18. Aspecto físico SIM900

ESPECIFICACIONES TECNICAS	
Voltaje de operación	5V - 20V
Comunicación	UART
Bandas de frecuencia	850/900/1800/1900MHz
Multi-GPRS	Slot clase 10/8
Estación móvil GPRS	Clase B
Clase 1	1W a 1800/1900MHz
Clase 4	2W a 850/1900MHz
Pila embebida TCP/UDP	Carga de datos a un servidor web
Bajo consumo de energía	1 5mA (modo sleep)
Temperatura de operación	-40°C a +85°C

Tabla 7. Especificaciones técnicas del SIM900 GSM/GPRS

El módulo SIM900 tiene múltiples puertos destinados a PMW y diversos UART para conectar con otros dispositivos más, ampliando aún más las posibilidades en el desarrollo y mejora del presente proyecto.

2.4.2 Arduino

Es importante elegir correctamente el modelo de tarjeta a utilizar, debido a que conectará con la mayor cantidad de dispositivos. De la sección anterior conocemos que es necesario contar con una tarjeta Arduino con 2 puertos UART: un puerto para la conexión con el módulo Bluetooth ® HC-05 y el segundo puerto para la conexión con la SIM900.

PARAMETRO	ARDUINO UNO	ARDUINO LEONARDO	ARDUINO MEGA 2560
Microcontrolador	ATmega328	ATmega32u4	ATmega2560
Voltaje de operación	5V	5V	5V
Pines digitales I/O	14 (6 con PWM)	20 (7 con PWM)	54 (15 con PWM)
Pines analógicos de entrada	6	12	16
Memoria Flash	32KB. (0.5KB es utilizado por el gestor de arranque)	32KB, (4KB son utilizados por el gestor de arranque)	256KB (8KB son utilizados por el gestor de arranque)
SRAM	2KB	2.5KB	8KB
EEPROM	1KB	1KB	4KB
UART	1	1	4
Oscilador cristal	16MHz	16MHz	16MHz

Tabla 8. Características extendidas de los modelos de Arduino

La tabla muestra las características más sobresalientes de los modelos más comunes en el mercado. La única placa que cuenta con los requisitos es el modelo MEGA 2560.

3 Planteamiento del problema

La llegada de vehículos eléctricos de bajo costo adquisitivo para el consumidor aún se vislumbra a futuro. [20] Los vehículos de combustión actuales no serán reemplazados en su totalidad en un par de meses o años, pueden pasar incluso décadas, principalmente por la falta de infraestructura como centros de recarga [21] Mientras tanto, la contaminación emitida como resultado de la combustión de los vehículos seguirá su curso.



Ilustración 19. Infografía Contaminación del aire, ONU [22]

Algunos fabricantes automotrices han introducido tecnologías que tratan de reducir las emisiones de gases contaminantes, por ejemplo, la tecnología del *Auto Start-Stop*: cada vez que se presiona el pedal del freno hasta que el vehículo llega detenerse totalmente, el motor se apaga, y hasta que se suelta el pedal del freno, el motor vuelve a encender [23]. Los vehículos que cuentan con esta tecnología son Chevrolet Cavalier® 2020 [24], Audi® todos los modelos [25].

Otra tecnología, es el sistema de gestión de consumo de combustible y centro de mensajes brindando una calificación al conductor después de cada viaje recorrido incorporado en el modelo Prius de TOYOTA®. Estas dos tecnologías son propias de vehículos de gama media-alta por el costo que representa la inversión en alguno de



estos. A continuación, se muestra una tabla con los precios disponibles al mes de noviembre de 2020.

Modelo de Vehículo	Precio
Chevrolet Cavalier 2020	DESDE \$261,700 [24]
Audi A1 2021	A partir de 429,900.00 MXN [26]
Toyota Prius 2021	Desde \$427,300 M.N. [27]

Tabla 9. Precio de Vehículos con tecnologías para el ahorro de combustible

El costo de los vehículos que cuentan con tecnologías para ayudar a la disminución de gases contaminantes y rendimiento de combustible no está dentro de las posibilidades de todos para adquirir una unidad.

Por otro lado, la demanda de componentes y dispositivos electrónicos han aumentado drásticamente desde los últimos años, además, en Internet se encuentra una extensa cantidad de código libre para ser implementado sobre cualquier tarjeta de desarrollo, tal es el caso de Arduino ®.

3.1 Requerimientos

Para que el proyecto sea viable, la mayoría de los vehículos comercializados en México deben poseer características similares, principalmente en el tamaño o capacidad volumétrica del motor.

Venta al público de automóviles						Cuadro VMRC-03
Serie mensual de 2018 a 2020 (Unidades)						
Periodo	Total	Nacionales				Importados
		Subcompactos	Compactos	De lujo	Deportivos	
2018	855,900	125,369	203,852	102	0	526,577
2019	764,175	114,576	168,781	283	0	480,535
2020	474,855	73,020	134,136	705	0	266,994

Tabla 10. Unidades de vehículos vendidos, INEGI [28]

En México la mayoría de los vehículos vendidos son compactos, con un motor menor a 2.0L [29].

3.1.1 Vehículos en México

Según datos del INEGI, el ranking de los modelos más vendidos en México es [30]:

1. Nissan Versa.
2. Chevrolet Aveo.
3. Nissan NP300.
4. Nissan March.
5. Volkswagen Vento.
6. Chevrolet Beat Notchback.
7. KIA Rio Sedan.

8. Chevrolet Beat.
9. Nissan Sentra.
10. Volkswagen Jetta MK7.

El tamaño de los motores o capacidad volumétrica de cada coche se muestran en la siguiente tabla.

Vehículo	Capacidad volumétrica del motor [L]
Nissan Versa	1.6 [31]
Chevrolet Aveo	1.5 [32]
Nissan NP300	2.5 [33]
Nissan March	1.6 [34]
Volkswagen Vento	1.6 [35]
Chevrolet Beat Notchback	1.2 [36]
KIA Rio Sedan	1.6 [37]
Chevrolet Beat	1.4 [38]
Nissan Sentra	2.0 [39]
Volkswagen Jetta MK7	2.0 [40]

Tabla 11. Capacidad volumétrica del motor en los 10 vehículos más vendidos

Tal como lo indican las estadísticas de INEGI, los vehículos mayormente vendidos son compactos, con motores menores a 2.0 L. El promedio se muestra en la siguiente gráfica.

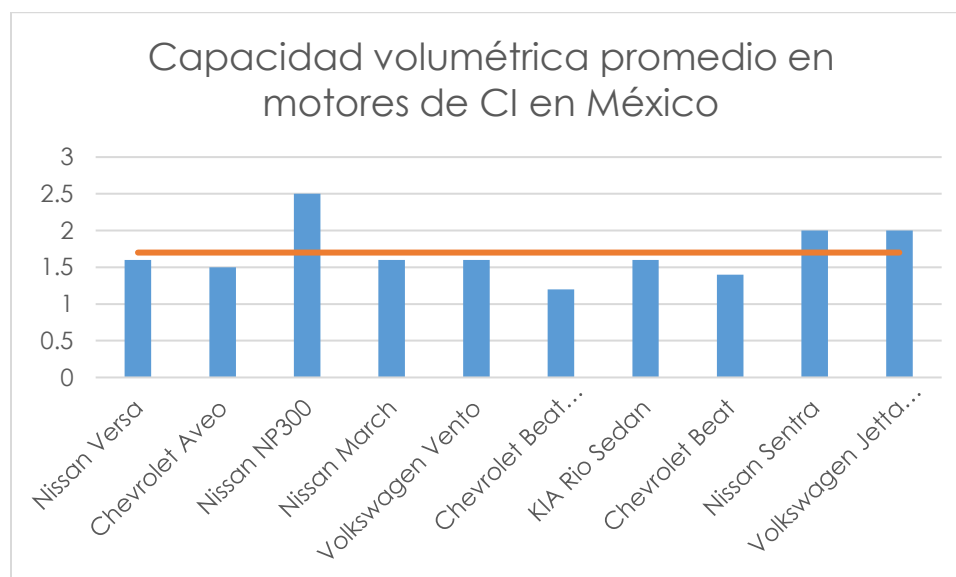


Ilustración 20. Gráfica de vehículos de mayor venta en México

El promedio del tamaño del motor ha resultado ser de 1.7 L, no obstante, no existen motores con esa capacidad volumétrica, por lo tanto, se tomará de referencia los motores de 1.6 L que es el número más cercano al valor comercializado.

Utilizando la misma información sobre los coches más vendidos, y según los fabricantes, el rendimiento en ciudad de cada vehículo se muestra en la siguiente tabla.

Vehículo	Rendimiento en [Km/L]	Referencia
Nissan Versa	15	[31]
Chevrolet Aveo	19.7	[32]
Nissan NP300	9.89	[33]
Nissan March	20.4	[34]
Volkswagen Vento	22.2	[35]
Chevrolet Beat Notchback	18.1	[38]
KIA Rio Sedan	17.81	[37]
Chevrolet Beat	18.1	[36]
Nissan Sentra	14.1	[39]
Volkswagen Jetta MK7	17.2	[40]

Tabla 12. Rendimiento de combustible en los 10 vehículos más vendidos

La tabla anterior será utilizada como referencia para la implementación en el algoritmo que ayudará para brindar recomendaciones en la conducción.

3.2 Propuesta de solución

De acuerdo con Forbes y con el fabricante de autos Ford, existen una serie de reglas para optimizar el consumo de combustible. La lista completa se encuentra en el Apéndice C, los puntos más relevantes se muestran en seguida.

- “Pisar a fondo el acelerador aumenta hasta cuatro veces el consumo de gasolina, así como frenar rápido”.
- “Circular el mayor tiempo posible en las relaciones más largas y a bajas revoluciones”.
- “Arrancar sin pisar el acelerador y usa la primera velocidad (la más potente de todas) sólo para el inicio de la marcha”.



- “Entre más baja sea la velocidad, más alta será la potencia necesaria y, por lo tanto, los requerimientos de gasolina. Manejar a partir de la segunda velocidad, evitando las aceleraciones bruscas, te puede ahorrar hasta 11% del consumo de combustible”.

De acuerdo con listado anterior, se observa que están presentes dos variables: Las RPM y la velocidad del automóvil, por lo tanto, se puede reordenar y clasificar en cada uno, además de tener presente la cantidad de combustible y el odómetro para ir comparando con el parámetro establecido por el fabricante.

RPM

- Arrancar sin pisar el acelerador.
- Cambiar marcha 2 segundos o 6 m después del arranque.
- Evitar las aceleraciones bruscas.
- Evitar sobre revolucionar el motor.
- Hacer el cambio de marcha entre 2000 y 2600 RPM.
- Circula el mayor tiempo posible en las relaciones más largas y a bajas revoluciones.

Velocidad

- Cambiar marcha cada 20 km/h, mantener 2500 rpm.
- Mantener una velocidad constante y máxima de 100 km/h.

Nivel de Combustible y Odometro

Comparar con rendimiento del fabricante.

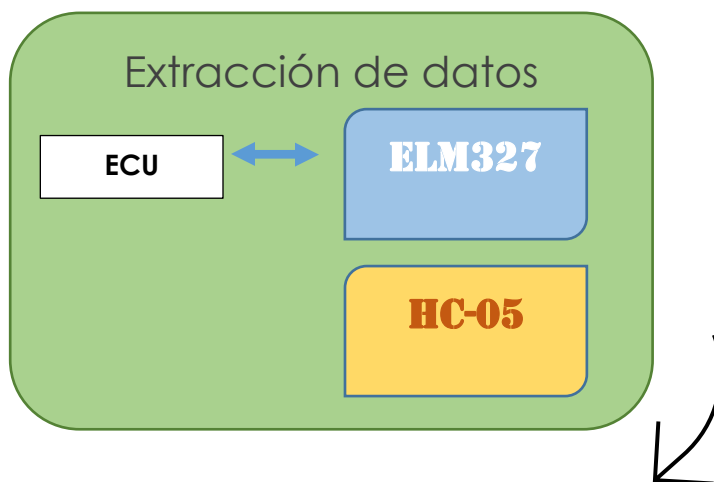
3.2.1 Sistema de monitoreo para recomendaciones finales en la conducción

Se ha mencionado que el sistema de gestión de datos para recomendaciones en la conducción se dividirá en 3 subsistemas. En esta sección se explicará la propuesta de solución a implementar.

Las funciones que cumple la tarjeta Arduino son la inicialización del ELM327, con su respectivo modo de extracción, almacenar esos datos y crear un mensaje para ser transmitido.

El módulo SIM900 envía el paquete de datos hacia un servidor.

En seguida se muestra un diagrama de bloques con los dispositivos y la interacción entre ellos.



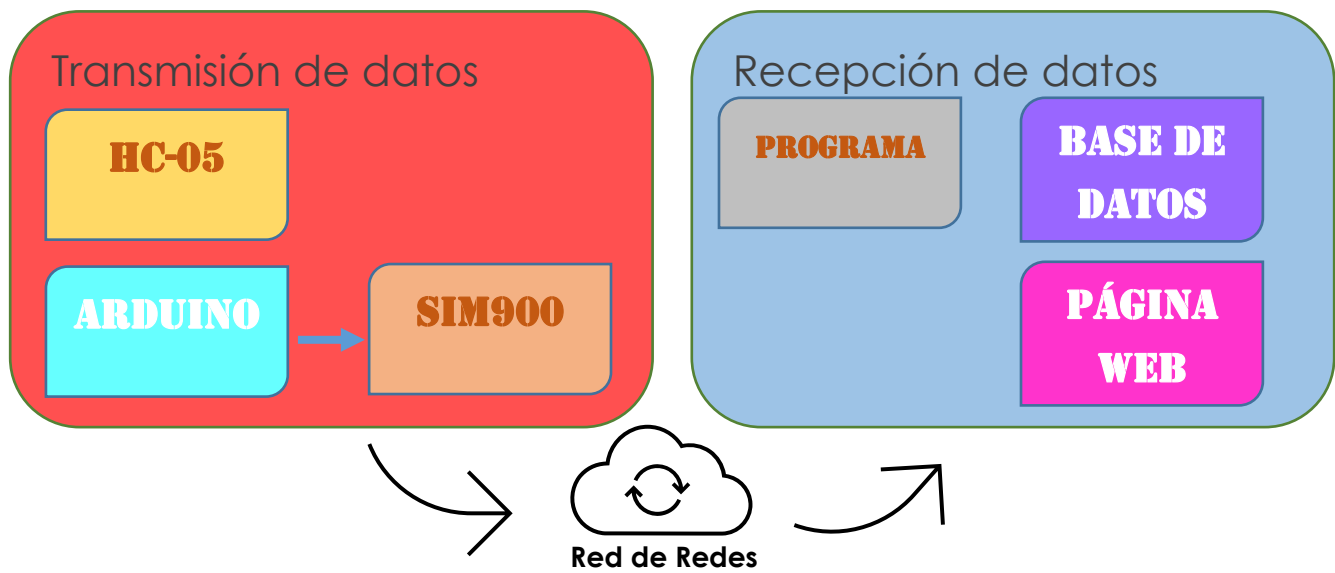


Ilustración 21. Estructura de la propuesta de solución

3.2.1.1 Subsistema de extracción de datos

Este subsistema se basa principalmente en el uso del dispositivo de diagnóstico ELM327 que fue definido por sus características en la sección 2.4.1. Adicionalmente, se usará una tarjeta de desarrollo Arduino MEGA 2560 y un módulo SIM900 GPRS que permite una gran cobertura para poder ser utilizado en cualquier parte de la República Mexicana.

Para programar esta parte la aplicación, se utiliza el lenguaje de programación basado en C++ que brinda el IDE Arduino. Este entorno permite incluir librerías que otros usuarios han creado con la libertad de modificarlo. La librería por utilizar es "ELMduno.h". Esta librería creada por Stanley Huang, acelera el tiempo de programación, solo basta con declarar el nombre del sensor a medir utilizando métodos.

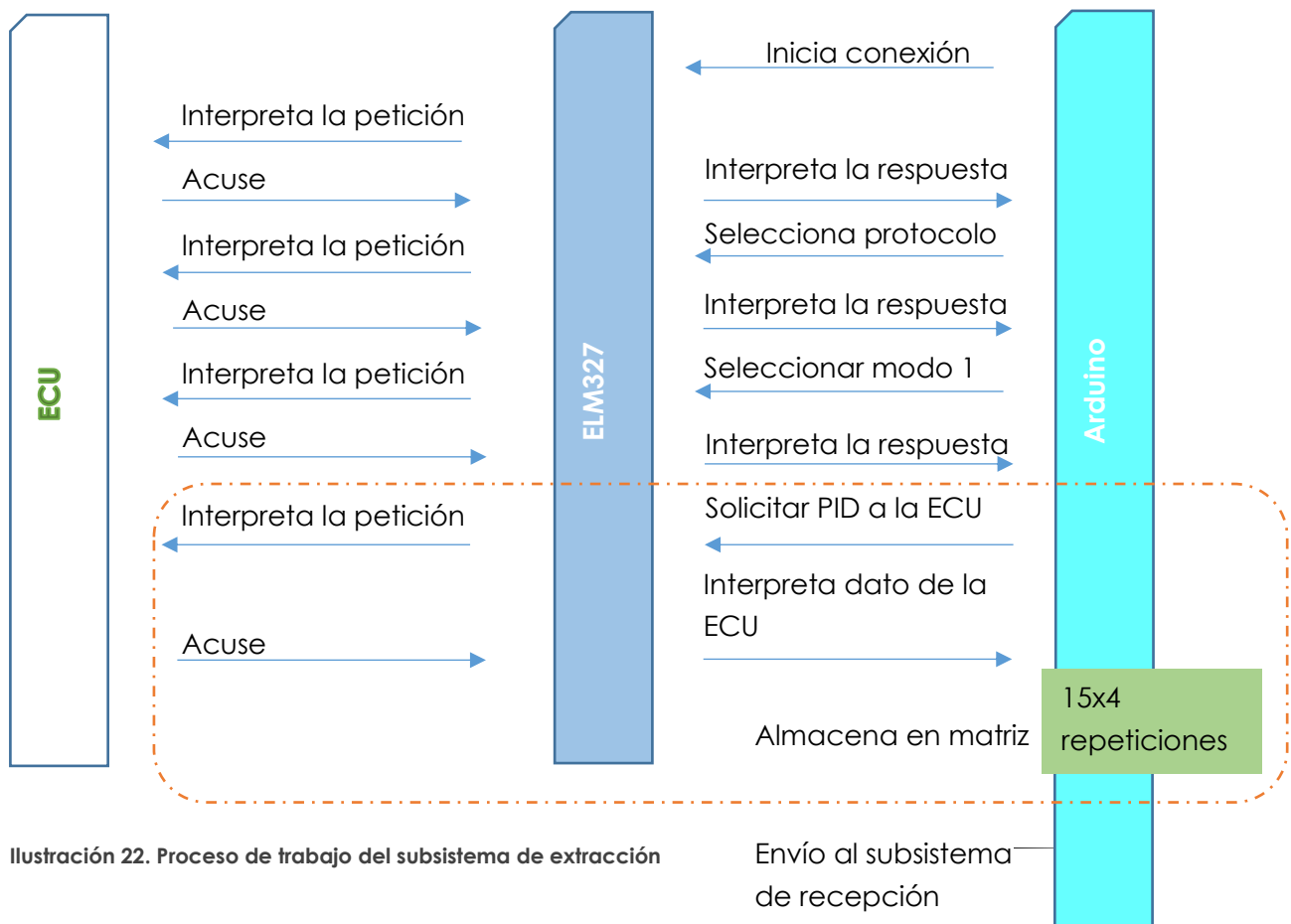


Ilustración 22. Proceso de trabajo del subsistema de extracción

Envío al subsistema de recepción



El proceso inicia con el Arduino, manda un mensaje al ELM372 con la instrucción de entrar al modo 1. La ECU lo recibe y manda una respuesta, de regreso el ELM327 lo interpreta y le confirma al Arduino que se ha conectado correctamente. Siguiendo el código, el Arduino solicita el PID correspondiente al sensor de las RPM, la ECU lo recibe gracias al ELM327 como intérprete y de vuelta obtiene el valor en tiempo real de las RPM, lo mismo sucede con la VSS, el nivel de combustible y el odómetro. El proceso anterior, se repite 15 veces. Lo que procede a continuación es encargado el sistema de transmisión.

3.2.2 Subsistema de transmisión de datos

La misma tarjeta Arduino se encargará de crear el mensaje que contiene el empaquetamiento de datos, lo envía al módulo SIM900 y este se encarga de mandarlo al servidor. Cabe resaltar que este módulo ha sido iniciado desde el encendido del vehículo, no obstante, es hasta este momento que entra en funcionamiento, anteriormente estando en un modo *sleep* o de reposo.

3.2.3 Fuente de alimentación

Es importante la implementación de una fuente de alimentación correcta, ya que el fabricante establece los parámetros mínimos necesarios para el correcto funcionamiento de los módulos. El amperaje de la fuente de alimentación debe ser como mínimo de 2A, las características se muestran a detalle en las hojas de dato ubicadas en el Apéndice C.

La fuente de alimentación se dividirá en 2 partes: La primera de 12V alimentará la SIM900 y la segunda parte alimentará el Arduino y el HC-05 con 5V. pg71

4 Desarrollo de la solución

La importancia del ahorro siempre está presente en todos los ámbitos, y también lo está cuando se trata del uso eficiente del combustible que se utiliza para el transporte de pasajeros y mercancías. El usuario de un vehículo debe saber cómo mejorar en sus hábitos de conducción para lograrlo.

Al final del viaje, el usuario registrado en la plataforma Web, observará en cualquier dispositivo conectado a Internet, los resultados de su condición en forma de una calificación. El proceso de esta aplicación es mostrado en la ilustración.

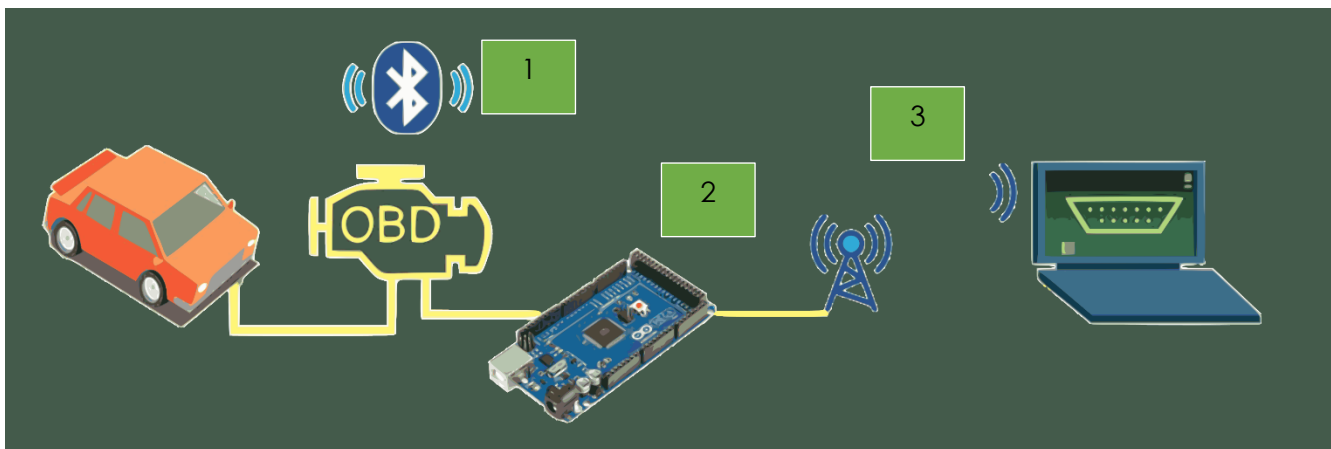


Ilustración 23. Diagrama Aplicación del proyecto

En el número 1 se encuentra la etapa de extracción de datos por medio de un dispositivo de diagnóstico automotriz.

Una vez extraído los datos, se procede con el almacenamiento temporal con la ayuda de una tarjeta de desarrollo, además este subsistema envía los paquetes de datos por medio de la Red GPRS, se muestra con el número 2.

Posteriormente el subsistema de recepción y análisis mostrado con el número 3, es la que dará la calificación al usuario utilizando el algoritmo programado.

Las etapas o subsistemas serán presentadas a detalle en los siguientes subcapítulos.

4.1 Subsistema de extracción de datos

Debido a la comunicación inalámbrica proporcionada por el ELM327 mediante el módulo Bluetooth incorporado, es indispensable utilizar un dispositivo Bluetooth maestro que indique los parámetros o información necesaria que la tarjeta Arduino necesita. La conexión entre el módulo Bluetooth HC-05 y el ELM327 se muestra a continuación.

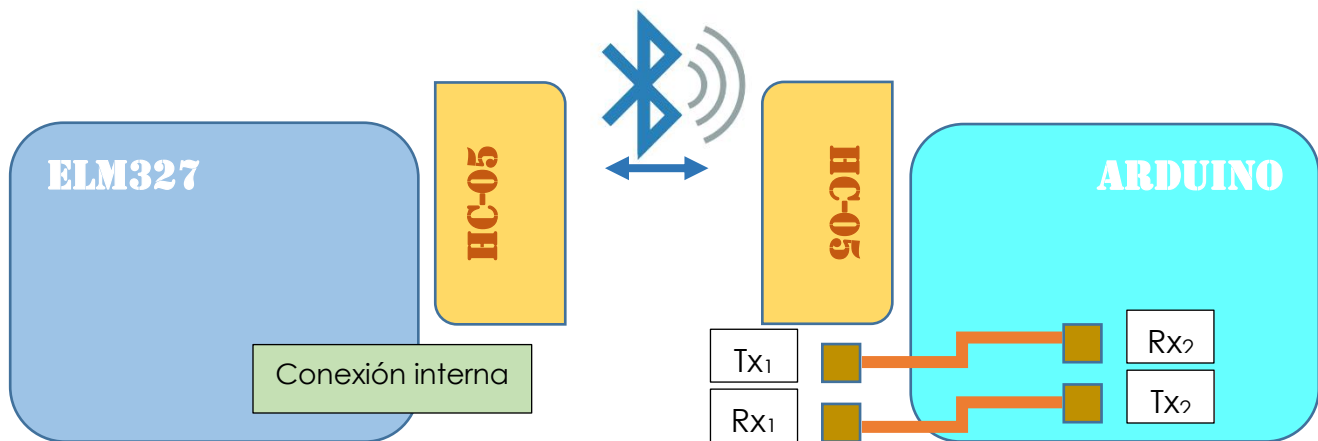


Ilustración 24. Conexión HC-05 y Arduino

La única conexión alámbrica será entre el módulo HC-05 y el Arduino y se hace de forma cruzada, anteriormente explicada.

4.1.1.1 Comandos AT

Los comandos AT son un lenguaje desarrollado para la configuración de módems y otros dispositivos [41]. Para poder modificar los parámetros que vienen por defecto, es indispensable conectar la tarjeta Arduino con la PC por medio de USB.

En el IDE (Entorno de Desarrollo Integrado) de Arduino, seleccionar el modelo de la placa y compilar el código completo que se encuentra en el Anexo A.


```
void setup(){  
  Serial.begin(9600);    // comunicación de monitor serial a 9600 bps  
  Serial.println("ok");  // escribe "ok" en el monitor  
  miBT.begin(38400);     // comunicación serie entre Arduino y el módulo a 38400 bps  
}
```

Ilustración 25. Fragmento del código Comandos AT

Es importante mencionar que la velocidad del puerto Serial para visualizar las respuestas a los comandos AT es de 9600 baudios y la velocidad de trabajo del módulo es a 38400 baudios.

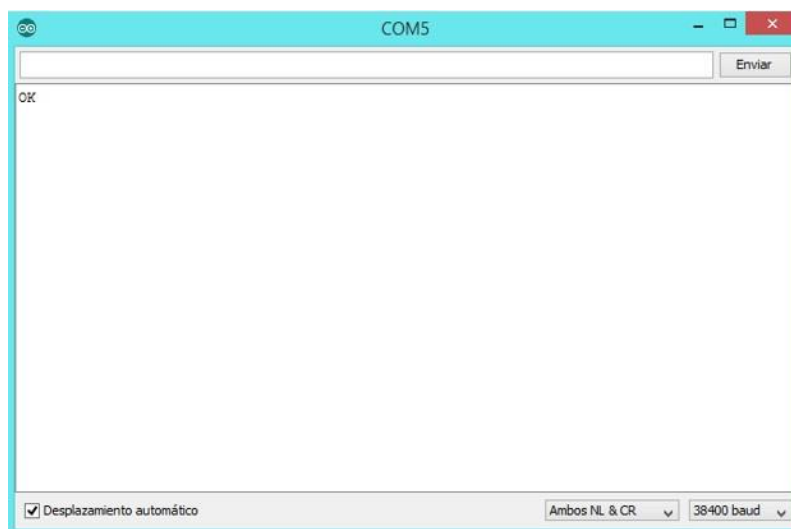


Ilustración 26. Monitor Serie Arduino

Por defecto el módulo Bluetooth está configurado en el modo esclavo=0, para cambiar de rol se utiliza la siguiente instrucción:

AT+ROLE=1

Cada vez que el vehículo sea puesto en marcha, se realizará el censado de la ECU, esto quiere decir, que cada vez que el vehículo se encienda, debe conectarse de forma automática la tarjeta Arduino mediante el HC-05 hacia el ELM327; así, el módulo

Bluetooth debe conectarse a un único dispositivo siempre que esté disponible, es decir, dentro del rango de los 10 m; con la siguiente sentencia se logra lo anterior:

`AT+CMODE=0`

Posteriormente, indicamos la dirección o el ID del dispositivo al cual debe conectarse:

`AT+BIND=000D,18,3A6789`

La dirección Bluetooth la obtuvimos con ayuda de un smartphone Android en la ventana de configuración al buscar un dispositivo nuevo.

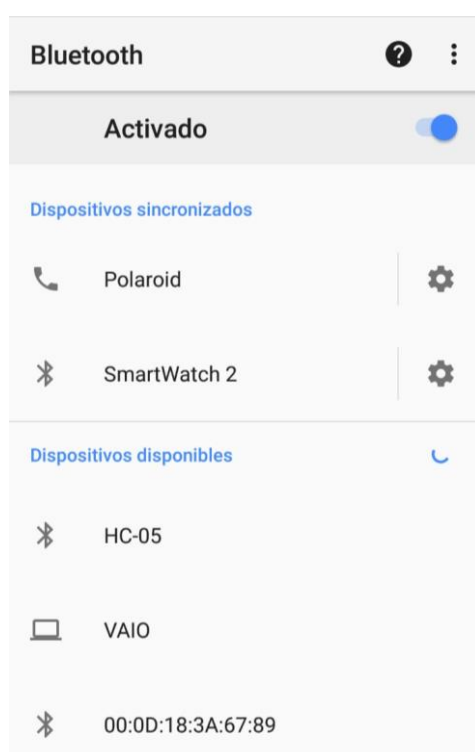


Ilustración 27. Captura de pantalla Dispositivos Bluetooth

Para que la comunicación sea eficaz, así como una persona debe hablar el mismo idioma, el módulo HC-05 debe ser configurado con la misma velocidad a la que trabaja el ELM327: 38 400 baudios

`AT+UART=38400,0,0`

4.1.2 Módulo de censado.

Utilizando el código ejemplo de la librería llamada "ELMduino", se programó la tarjeta Arduino para la obtención de los primeros datos provenientes de la ECU. En el siguiente fragmento de código se muestra de qué manera es posible.

```
if (myELM327.status == ELM_SUCCESS)
{
    rpm = (uint32_t)tempRPM;
    Serial.print("RPM: "); Serial.println(rpm);
}
```

Ilustración 28. Fragmento del código Obtención de datos

Mientras el programa es ejecutado, se muestra en pantalla lo siguiente

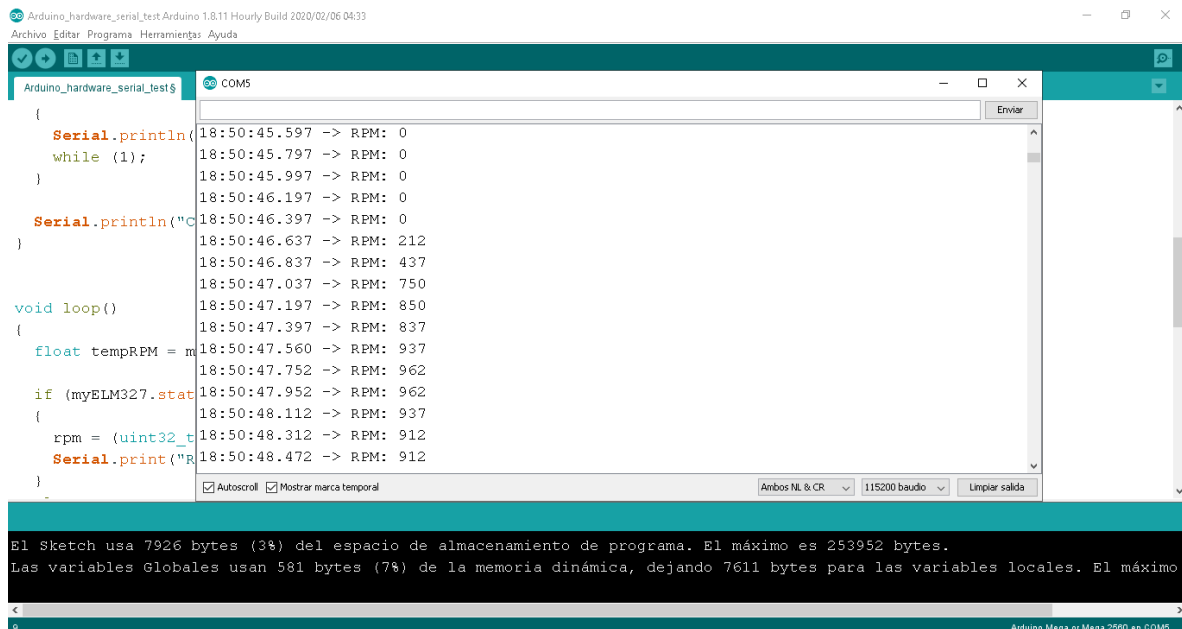


Ilustración 29. Ventana del Puerto Serial

Una vez realizado el correcto censado de las RPM, se procede para obtener las variables restantes: Velocidad, Nivel de Combustible y Odómetro.

4.1.3 Módulo de almacenamiento

Para almacenar la información que se extrae del vehículo es necesario implementar una base de datos que va a ser la encargada de guardar la información recibida.

Se escogió trabajar con una base de datos llamada SQLite3, la cual almacenará toda la información de nuestros datos de los sensores, en este caso velocidad, RPM y odómetro.

SQLite3 viene implementada con nuestro framework para desarrollo web, llamado django [50], este mismo es el que se utilizará para crear nuestro prototipo de plataforma web, en la cual se mostrará nuestros resultados de manejo y a la vez una calificación de manejo.

```
>>> from django.db import connection
>>> connection.queries
[{'sql': 'SELECT polls_polls.id, polls_polls.question, polls_polls.pub_date FROM polls_polls',
'time': '0.002'}]
```

Ilustración 30. Comandos para solicitar ver base de datos

4.2 Bloque de servicio

La principal función que cumple el bloque de servicio consiste en la recepción del mensaje UDP, con los valores medidos por los sensores del motor de un automóvil.

Para programar el servicio, se utilizó el editor de texto Sublime Text 2[51], el cual es un entorno de desarrollo integrado con código abierto basado en el lenguaje de programación Python[52]. La Utilización de Sublime Text se debe a los siguientes factores: fácil instalación y uso, se le puede utilizar para el desarrollo de aplicaciones web, permite la conexión con la base de datos ya que dentro de la misma viene incluida.

El programa de del bloque de servicio se compone de tres subprocesos que son: apertura del puerto y lectura del mensaje, decodificación del mensaje UDP y conexión a la base de datos.

La decodificación realiza la separación de cada uno de los valores medidos por los sensores del motor del vehículo que forman parte del mensaje UPD, mediante el uso de una función Split para posteriormente guardarlos uno a uno en una matriz. Adicionalmente, en el caso que se produzcan espacios en blanco en los valores, se procede a eliminarlos a través de la utilización de la función "trim".

4.3 Subsistema de recepción

Para poder desarrollar el prototipo de plataforma es necesario tener instalado el lenguaje de programación Python, en este caso utilizamos la versión más reciente de está que es la 3.9, posteriormente al ya tener instalado Python se procede a crear una carpeta en la cual, se guardara toda nuestra información del proyecto.

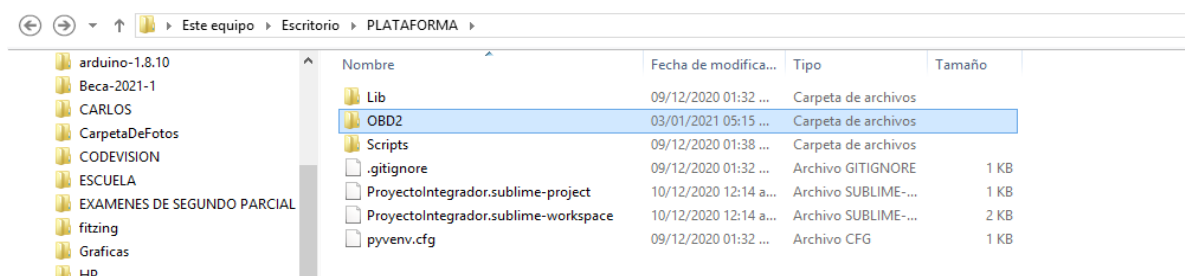
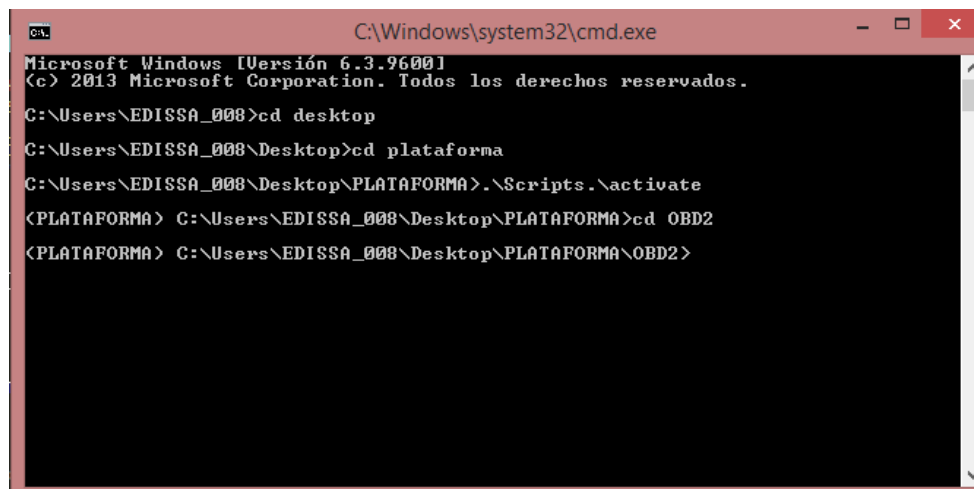


Ilustración 31. Rutas y Carpetas donde se encuentra guardado el proyecto

En el presente caso se guardó en el escritorio de nuestra PC local. Lo siguiente que se realizó fue habilitar un entorno virtual el cual es donde desarrollamos la plataforma en la cual se visualiza la información de nuestros vehículos y datos de manejo.

Para habilitar el entorno virtual se hace desde el CMD (Símbolo del sistema) de la máquina y se procede a escribir comandos para activar el entorno y al mismo tiempo "django" este disponible; la creación del entorno virtual se muestra en el Anexo C.



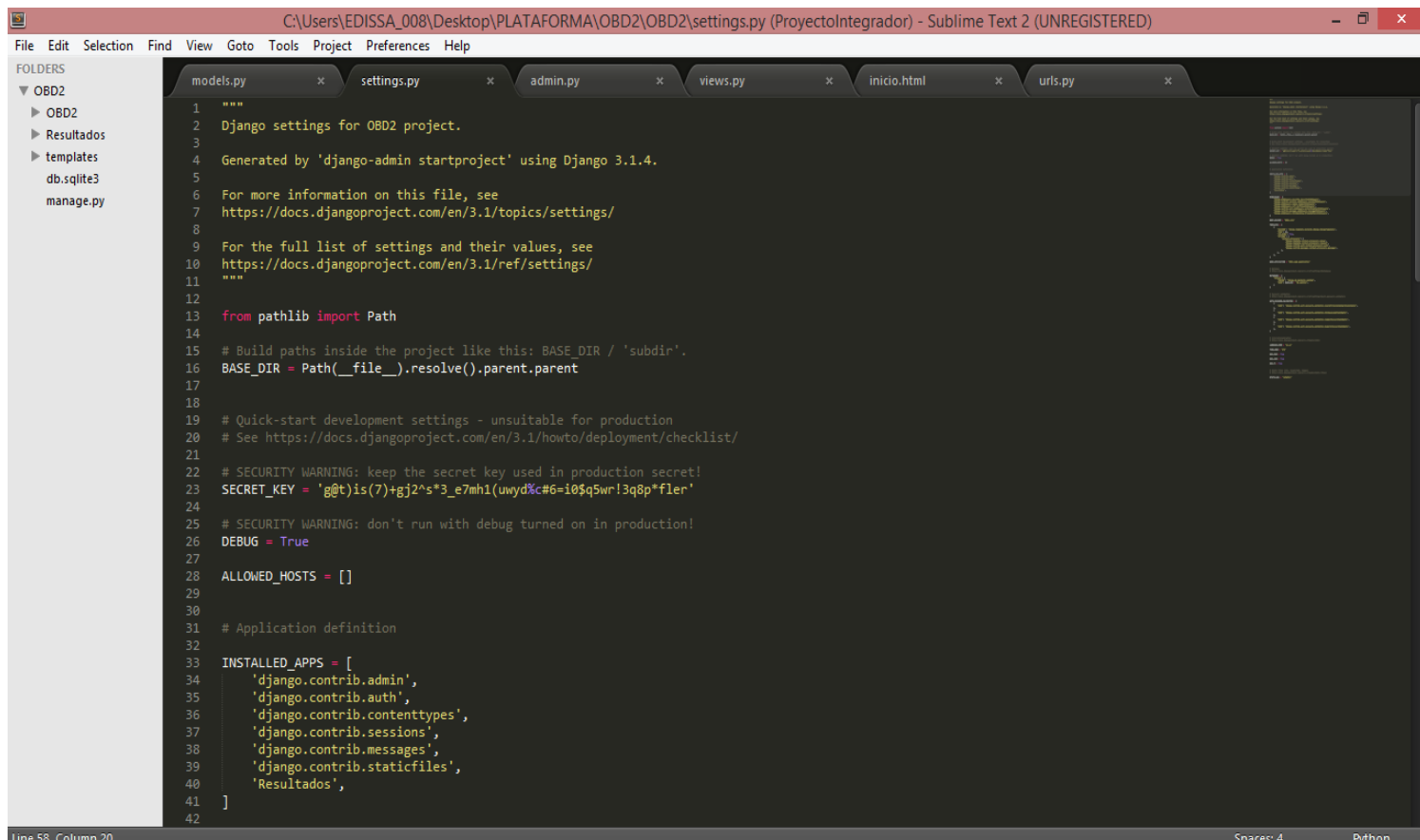
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\EDISSA_008>cd desktop
C:\Users\EDISSA_008\Desktop>cd plataforma
C:\Users\EDISSA_008\Desktop\PLATAFORMA>.\Scripts.\activate
(PLATAFORMA) C:\Users\EDISSA_008\Desktop\PLATAFORMA>cd OBD2
(PLATAFORMA) C:\Users\EDISSA_008\Desktop\PLATAFORMA\OBD2>
```

Ilustración 32. CMD y comandos para activar el entorno virtual

Una vez activado el entorno virtual, se procedió a desarrollar la plataforma web, la plataforma se desarrolló en el editor de texto ya antes mencionado “Sublime Text 2”, para esto el editor se tiene que emparejar con nuestro entorno virtual, se hace de la siguiente forma, se abre el editor de texto y se procede a abrir un proyecto nuevo, y nos dirigimos hacia la ruta en donde tenemos guardada nuestra carpeta la cual contiene nuestro proyecto.

“Sublime Text 2” es un editor muy amable con el usuario, fácil de ocupar y ahí mismo se muestra todo lo que tenemos integrado de nuestro proyecto raíz. De forma más clara se desarrolló el prototipo de plataforma.



```
1  """
2  Django settings for OBD2 project.
3
4  Generated by 'django-admin startproject' using Django 3.1.4.
5
6  For more information on this file, see
7  https://docs.djangoproject.com/en/3.1/topics/settings/
8
9  For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/3.1/ref/settings/
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18 # Quick-start development settings - unsuitable for production
19 # See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/
20
21 # SECURITY WARNING: keep the secret key used in production secret!
22 SECRET_KEY = 'g@t)is(7)+gj2^s*3_e7mh1(uwyd#c#6=10$q5wr!3q8p*fler'
23
24 # SECURITY WARNING: don't run with debug turned on in production!
25 DEBUG = True
26
27 ALLOWED_HOSTS = []
28
29 # Application definition
30
31 INSTALLED_APPS = [
32     'django.contrib.admin',
33     'django.contrib.auth',
34     'django.contrib.contenttypes',
35     'django.contrib.sessions',
36     'django.contrib.messages',
37     'django.contrib.staticfiles',
38     'Resultados',
39 ]
```

Ilustración 33. Editor de texto Sublime Text 2

4.3.1 Módulo de Interfaz hombre-máquina

Una vez desarrollada la aplicación se procede a ponerla a funcionar, está se activa por medio del CMD y se escribe el siguiente comando:

```
python manage.py runserver
```

Este comando le ordena al servidor activarse y comenzar a funcionar: por lo siguiente nos arroja una dirección IP (<http://127.0.0.1:8000/>), copiamos esa dirección en nuestro navegador web y nos arroja a nuestra plataforma.

```
C:\Windows\system32\cmd.exe - python manage.py runserver
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.
C:\Users\EDISSA_008>cd desktop
C:\Users\EDISSA_008\Desktop>cd plataforma
C:\Users\EDISSA_008\Desktop\PLATAFORMA>.\Scripts.\activate
(PLATAFORMA) C:\Users\EDISSA_008\Desktop\PLATAFORMA>cd OBD2
(PLATAFORMA) C:\Users\EDISSA_008\Desktop\PLATAFORMA\OBD2>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 06, 2021 - 14:29:30
Django version 3.1.4, using settings 'OBD2.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[06/Jan/2021 14:30:57] "GET / HTTP/1.1" 200 16351
[06/Jan/2021 14:30:58] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[06/Jan/2021 14:30:58] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
```

Ilustración 34. CMD y comandos para correr el servidor

Al abrir la dirección en nuestro navegador, nos cargara la página principal de nuestra plataforma, si por alguna razón se tiene algún error el momento de programar la página en “Sublime Text 2”, el mismo CMD no nos permitirá abrir la página y nos mostrará un mensaje de error.

A continuación, se muestra la vista principal de la interfaz de nuestra plataforma.

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add Change
Users	+ Add Change

RESULTADOS

Registrados	+ Add Change
-------------	--

Recent actions

My actions

None available

Ilustración 35. Vista principal de la plataforma en la parte administrativa

5 Validación del sistema

5.1 Subsistema de extracción de datos

La solución propuesta implementada puede ser utilizada en automóviles que cuentan con el conector DLC para poder conectarse directamente a la Ecu. Las pruebas se planeaban realizar en dos automóviles, pero debido a la pandemia que azotó a nuestra sociedad, solamente se tuvo acceso a una camioneta marca Chevrolet Tracker® año 2007.

En esta se logró verificar que no siempre se pueden obtener todas las medidas que producen los sensores del motor. Esto se deba a que antes de que el sistema OBD-II se estandarice, el fabricante utiliza su propio criterio para la transmisión de datos, y por esta razón en alguno de ellos se producen incompatibilidades para la obtención de medidas.

5.1.1 Pruebas de Funcionamiento

Anterior a la realización de las pruebas de funcionamiento se instaló dentro del automóvil, el sistema de adquisición de datos, el bloque de procesamiento y transmisión de información, el Primero, es el compuesto por el lector ELM327 que esta emparejado con nuestro dispositivo Bluetooth, el bloque de procesamiento está compuesto por el Arduino mega, y el SIM900 GSM/GPRS.

Se procede a conectar el lector ELM327 al conector del automóvil DLC. Por lo general el conector DLC siempre viene posicionado del lado del conductor debajo del volante. Si la ubicación del conector no está ahí se tendrá que encontrar para poder realizar la conexión.



Ilustración 36. Conexión de ELM327 a vehículo



Ilustración 37. Conexión de ELM327 a vehículo

Las pruebas fueron realizadas y enfocadas en verificar el funcionamiento del dispositivo Arduino Mega, para recibir y procesar las respuestas provenientes de la ECU del automóvil.

En la siguiente fotografía se muestra un ejemplo del resultado de la prueba realizada para la obtención las RPM. Se observa la medida de forma analógica en el tacómetro de la camioneta. Por lo que se llega la conclusión que las lecturas son correctas, así como el proceso que realiza el dispositivo Arduino Mega mediante el programa implementado.



Ilustración 38. conexión de sistema de transmisión a vehículo

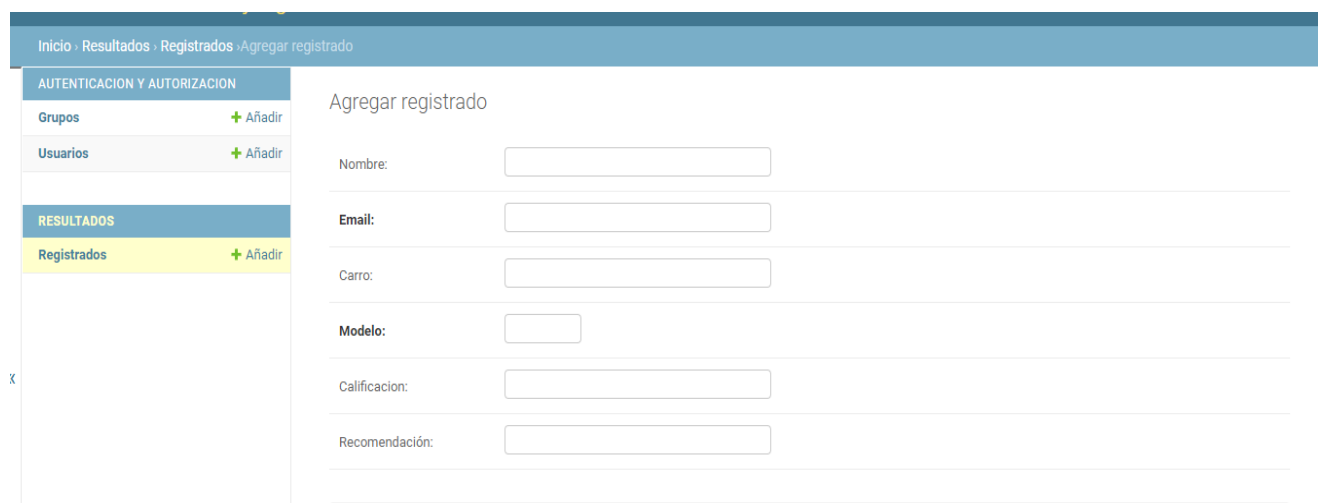
5.2 Validación de la plataforma.

Para validar el correcto funcionamiento de la plataforma se tuvieron que hacer varias pruebas para comprobar que efectivamente se estaba conectando con la base de datos, al hacer las pruebas correspondientes se llegó a la conclusión que si se tenía un enlace correcto.

Para lograr dar una calificación se tuvo que desarrollar un algoritmo que pudiera lograr captar los datos de la base de datos de los sensores, el censado de los valores tiene que ser cada tres segundos, ya que si se deja de mandar valores durante un determinado tiempo el conductor puede acelerar de más en un tiempo corto y al mandar la información no se podría captar está misma.

Así mismo como solo se pudo tener acceso a un solo vehículo, se tomaron las especificaciones del vehículo en el cual se hicieron las pruebas, en este caso fue la camioneta Chevrolet Tracker 2007, entra dentro de los motores que queremos abarcar ya que es de cilindrada baja, pero de un poco más de capacidad volumétrica.

En la imagen que se muestra a continuación se expresa los datos que se obtienen y se guardan del vehículo, dando una calificación y una recomendación.



Inicio · Resultados · Registrados · Agregar registrado	
AUTENTICACION Y AUTORIZACION	
Grupos	+ Añadir
Usuarios	+ Añadir
RESULTADOS	
Registrados	+ Añadir

Agregar registrado

Nombre:

Email:

Carro:

Modelo:

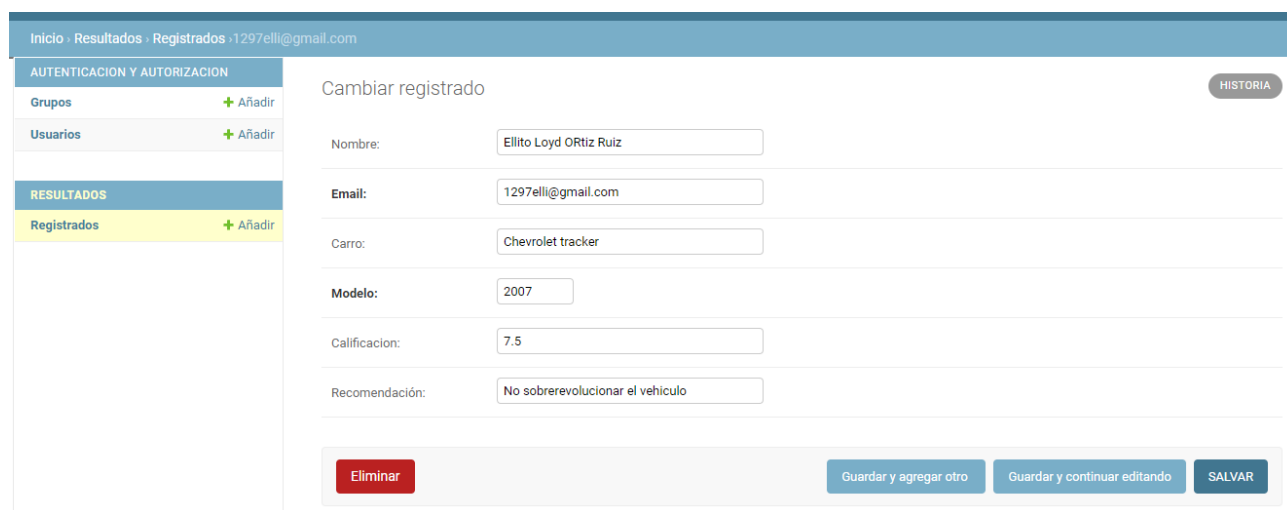
Calificacion:

Recomendación:

Ilustración 39 validación de la plataforma

Al momento de terminar el viaje, se los datos se ven reflejados, y estos mismos se muestran en la plataforma.

Como el nombre del proyecto lo dice, no enfocamos hacer un prototipo para que de esa forma en trabajos futuros se pueda ir mejorando eh implementando más variables, lo que se tiene que recordar es que la plataforma únicamente funciona en una PC local donde se tenga guardados los datos del proyecto, ya que si uno intenta ingresar a la dirección IP no cagara la página y no se podrá acceder a esta misma.



Inicio · Resultados · Registrados · 1297elli@gmail.com

AUTENTICACION Y AUTORIZACION

- Grupos [+ Añadir](#)
- Usuarios [+ Añadir](#)

RESULTADOS

- Registrados [+ Añadir](#)

Cambiar registrado [HISTORIA](#)

Nombre:

Email:

Carro:

Modelo:

Calificacion:

Recomendación:

[Eliminar](#) [Guardar y agregar otro](#) [Guardar y continuar editando](#) [SALVAR](#)

Ilustración 40. mensaje final en la plataforma al momento de terminar la conducción



6 Costos

En este apartado se incluyen los costos referenciales en la implementación de la solución propuesta.

En la tabla que se muestra a continuación se detallan los costos directos relacionados con los materiales adquiridos, estos sirvieron para la implementación de la solución propuesta.

La presente cotización se realizó en pesos mexicanos.

Objeto	Cantidad	Costo Unitario	Costo Total
Arduino Mega 2560	1	\$250.00	\$250.00
Bluetooth HC-05	1	\$110.00	\$110.00
SIM900	1	\$450.00	\$450.00
ELM327	1	\$180.00	\$180.00
Jumpers	20	\$2.00	\$40.00
Cargador para auto 5V	1	\$130.00	\$130.00
Total			\$1,160.00

Tabla 13. Costos de componentes para la elaboración del proyecto



7 Conclusiones

7.1 CONCLUSIONES

Se comprobó la importancia de los sensores en el vehículo, ya que permiten la gestión electrónica del automóvil, los valores obtenidos son utilizados por la ECU para registrar de manera exacta el estado real del motor de un vehículo.

De acuerdo con los protocolos en torno a comunicación en OBD-II: SAE J1850 PWM, SAE J1850 VPM, ISO 9141-2, ISO 14230 KWP e ISO 15765 CAN; conocerlo fue de gran ayuda para entender la forma en cómo se enlaza el ELM327, reconociendo automáticamente el protocolo del vehículo, y luego envía la información de manera inalámbrica a través del módulo Bluetooth que tenía incorporado.

En base al estudio de trabajos previos se determinó que la mayoría de ellos se orientan a la creación de aplicaciones de software que simulan el funcionamiento de un escáner automotriz, para monitorear los sensores del motor de un automóvil. Dichas aplicaciones funcionan en la proximidad de un automóvil.

Dentro del bloque de transmisión, el dispositivo Arduino Mega 2560 cumple un papel fundamental ya que se encarga de inicializar el lector ELM327 y seleccionar el protocolo de comunicación a través de comandos AT.

Se comprobó la versatilidad de la librería "ELMDuino.h" en cuanto a la obtención de datos necesarios para este proyecto, debido a que soporta el modo 1 del sistema OBD-II (adquisición de datos en tiempo real), y debido a que está programada para que la transferencia de datos se realice a través de los puertos UART de los dispositivos Arduino.

El uso de un servicio telemático hizo posible que se puedan recibir todos los datos que provengan de uno o más automóviles, y que los mismos sean gestionados para su almacenamiento y procesamiento. El uso del *framework* Django ayudaron a implementar el servicio, a procesar y almacenar la información.



El uso de la base de datos como medio de almacenamiento de información, ayudó a tener un esquema más sistemático, tanto para la inserción de datos como para la consulta, permitiendo realizar reportes fácilmente.

La captura de datos y su visualización llevada a cabo por medio de la interfaz web básica, y a través de notificaciones, podría ayudar a realizar un control preventivo del automóvil.

7.2 RECOMENDACIONES

Teniendo en cuenta que el lector ELM327 utilizado en la solución propuesta soporta todos los modos de medición del sistema OBD-II, sería interesante desarrollar nuevos programas donde la solución propuesta también trabaje en el modo 3 para poder determinar directamente las fallas que se producen en los sensores del motor de un automóvil y corregirlas de forma remota.

Se podría implementar todo el sistema gestionándolo desde un teléfono Android, con la posibilidad de utilizar la geolocalización y ubicar el vehículo en caso de robo.

Considerando que en el programa desarrollado para el dispositivo Arduino Mega 2560 se utiliza la librería "ELMDuino.h" creada por Stanley Huang, se recomienda hacer un seguimiento a todas las actualizaciones que se realicen en la librería.

Teniendo en cuenta que la solución propuesta es un prototipo inicial, se recomienda desarrollar un sistema más robusto, el cual pueda trabajar sin ningún inconveniente con mayor número de medidas y automóviles.

Se recomienda desarrollar una aplicación web con una mejor presentación hacia el usuario, donde se pueda realizar el registro tanto de clientes como de automóviles que utilicen la solución propuesta. Adicionalmente se debería mejorar el diseño de la base de datos creada.

8 Referencias

- [1] geriatricarea, «Los altos niveles de contaminación agravan y favorecen la expansión del COVID-19,» 1 junio 2020. [En línea]. Available: <https://www.geriatricarea.com/2020/06/14/investigaciones-relacionan-altos-niveles-de-contaminacion-con-la-mayor-expansion-del-covid-19/>. [Último acceso: 13 agosto 2020].
- [2] CARB, «The California Air Resources Board,» 8 abril 2019. [En línea]. Available: <https://ww2.arb.ca.gov/es/about>. [Último acceso: 13 agosto 2020].
- [3] FORD, «Consejos Para Ahorrar Gasolina en tu Auto,» 19 mayo 2017. [En línea]. Available: <https://www.ford.mx/blog/experto/ahorra-gasolina-7-consejos-201810/>. [Último acceso: 13 agosto 2020].
- [4] OBD-II ELM327, «Sistema OBD-II,» 2017. [En línea]. Available: <https://obd2-elm327.es/sistema-obd2-historia-descripcion-futuro>. [Último acceso: 20 agosto 2020].
- [5] ibtaller, «Historia del OBD,» 2011. [En línea]. Available: <http://ibtaller.com/historia-del-obd/>. [Último acceso: 20 agosto 2020].
- [6] COMISIÓN NACIONAL PARA EL USO EFICIENTE DE LA ENERGÍA, «CONUEE,» [En línea]. Available: https://www.gob.mx/cms/uploads/attachment/file/187221/diagnosticoabordo_1_260117.pdf.
- [7] S. Y. Kim, «Prototype remove vehicle diagnostics for police cruisers,» B. S. University of New Hampshire, 2006.
- [8] J. E. M. Anastasio, «Caracterización de los estilos de conducción mediante smartphones, dispositivos OBD-II y redes neuronales,» Universidad Politecnica de Valencia, 2012.
- [9] Car-Tec, «¿Cómo funciona la ECU?,» [En línea]. Available: <https://www.car-tec.es/blog/como-funciona-una-ecu/>. [Último acceso: 15 agosto 2020].
- [10] M. A.-O. y S. R. Peñacoba, «Efecto Hall,» [En línea]. Available: <http://rsefalicante.umh.es/TemasElectromagnetismo/Electromagnetismo07.htm>. [Último acceso: 25 agosto 2020].
- [11] CódigosDTC, «Sensor de posición del cigüeñal - Sensor CKP,» octubre 2018. [En línea]. Available: <https://codigosdtc.com/sensor-ckp/>. [Último acceso: 25 agosto 2020].



- [12] S. O. E. I. Cervantes, «Escaner automotiz de pantalla táctil,» Instituto Politécnico Nacional, 2010.
- [13] J. A. R. Marín, «Sistema eléctrico y de seguridad y de confortabilidad,» 2011.
- [14] elmelectronics, «ELM327-OBD TO RS232,» [En línea]. Available: <https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf>. [Último acceso: 30 agosto 2020].
- [15] Automotive Diagnostic, «Manual OBD-II,» [En línea]. Available: http://alvarestech.com/temp/murilo/Manual%20OBD_II.pdf. [Último acceso: 15 octubre 2020].
- [16] ARDUINO, «¿Que es Arduino?,» 2010. [En línea]. Available: <https://arduino.cl/que-es-arduino/>. [Último acceso: 27 agosto 2020].
- [17] nper, [En línea]. Available: <https://www.nperf.com/es/map/MX/-/363.Telcel/signal/?ll=21.497680827871587&lg=-96.59179553389552&zoom=5>. [Último acceso: 17 septiembre 2020].
- [18] BBC NEWS, «Cuáles son las diferencias entre E, GPRS, 3G, 4G, 5G y esas otras redes a las que se conecta tu celular,» 16 mayo 2016. [En línea]. Available: <https://www.bbc.com/mundo/noticias-37247130>. [Último acceso: 3 septiembre 2020].
- [19] INSTITUTO FEDERAL DE TELECOMUNICACIONES, «Sabías qué la Telefonía Móvil...,» [En línea]. Available: <http://www.ift.org.mx/usuarios-telefonía-movil/sabias-que-la-telefonía-movil>. [Último acceso: 30 agosto 2020].
- [20] El Universal, «7 autos eléctricos "accesibles",» 25 junio 2020. [En línea]. Available: <https://www.eluniversal.com.mx/autopistas/7-autos-electricos-accesibles-en-mexico>.
- [21] El Universal, «Esto cuesta recargar la batería de un auto eléctrico,» 27 junio 2020. [En línea]. Available: <https://www.eluniversal.com.mx/autopistas/cuanto-cuesta-recargar-la-bateria-de-un-auto-electrico-en-mexico>.
- [22] ONU Environment, [En línea]. Available: <http://www.web.unep.org/geo/assessments/regional-assessments/regional-assessment-latin-america-and.caribbean>.
- [23] Ford Mx, «¿Qué es el Sistema Auto Start-Stop y para qué Sirve?,» [En línea]. Available: <https://www.ford.mx/blog/experto/sistema-auto-start-stop-funcionamiento-201909/>. [Último acceso: 6 septiembre 2020].



- [24] Chevrolet Cavalier, [En línea]. Available: <https://www.chevrolet.com.mx/autos/familiares-cavalier/especificaciones-tecnicas>.
- [25] Audi, [En línea]. Available: <https://www.audi.es/es/web/es/innovacion-audi/tecnologia/sistema-start-stop.html>.
- [26] Audi, «Audi A1,» [En línea]. Available: <https://www.audi.com.mx/mx/web/es/models/a1/a1sb20.html>.
- [27] TOYOTA, «Prius,» [En línea]. Available: <https://www.toyota.mx/modelo/prius>.
- [28] INEGI, «Vehículos de motor registrados en circulación,» [En línea]. Available: <https://www.inegi.org.mx/programas/vehiculosmotor/#Tabulados>. [Último acceso: 23 septiembre 2020].
- [29] INEGI, «Glosario,» [En línea]. Available: <https://www.inegi.org.mx/app/glosario/default.html?p=vmrcm>. [Último acceso: 23 septiembre 2020].
- [30] MotorPasión, «Autos Vendidos en México,» [En línea]. Available: <https://www.motorpasion.com.mx/industria/autos-vendidos-mexico-2019>. [Último acceso: 15 septiembre 2020].
- [31] Nissan, [En línea]. Available: https://www.nissan.com.mx/content/dam/Nissan/mexico/brochures/v-drive/MY20/031019_VDrive2020_Manteleta%20BAJA.pdf.
- [32] Chevrolet, [En línea]. Available: <https://www.chevrolet.com.mx/content/dam/chevrolet/na/mx/es/index/cars/2020-aveo/mov/02-pdf/2020-aveo-hoja-especificaciones.pdf>.
- [33] Nissan NP, [En línea]. Available: https://www.nissan.com.mx/content/dam/Nissan/mexico/brochures/np300-frontier/MY20/np300Frontier_2020_catalogo.pdf.
- [34] Nissan M, [En línea]. Available: https://www.nissan.com.mx/content/dam/Nissan/mexico/brochures/march/MY20/020819%20Manteleta%20MARCH%20MY20_BAJA.pdf.

- [35] VW, [En línea]. Available:
https://www.vw.com.mx/idhub/content/dam/onehub_pkw/importers/mx/fichas-tecnicas/2020/vento/vw-nuevo-vento-2020-ft.pdf.
- [36] Chevrolet, [En línea]. Available:
<https://www.chevrolet.com.mx/content/dam/chevrolet/na/mx/es/index/cars/2020-beat-hb/mov/02-pdf/2020-beat-hb-ficha-tecnica-v2.pdf>.
- [37] KIA, [En línea]. Available: https://www.kia.com.mx/showroom/rio-sedan.html?gclid=EAlaQobChMlyg_gxbOh7QIVTtbACh2dcgU4EAAYASABEgJVNPB_BwE.
- [38] Chevrolet Bt, [En línea]. Available:
<https://www.chevrolet.com.mx/content/dam/chevrolet/na/mx/es/index/cars/2020-beat-nb/mov/02-pdf/2020-beat-nb-ficha-tecnica-v2.pdf>.
- [39] Nissan, «Sentra,» [En línea]. Available:
https://www.nissan.com.mx/content/dam/Nissan/mexico/brochures/sentra/MY20/CAT_SENTRA_2020.pdf.
- [40] VW, «Jetta,» [En línea]. Available:
https://www.vw.com.mx/idhub/content/dam/onehub_pkw/importers/mx/fichas-tecnicas/producto/2021/jetta/vw-jetta-2021-ft.pdf.
- [41] naylamp Mechatronics, «Configuración del módulo bluetooth HC-05 usando comandos AT,» 28 diciembre 2016. [En línea]. Available:
https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html. [Último acceso: 24 octubre 2020].
- [42] PowerBroker2, «ELMDUINO,» [En línea]. Available: <https://github.com/PowerBroker2/ELMduino>. [Último acceso: 9 septiembre 2020].
- [43] Copro, «PIDs OBD-II,» [En línea]. Available: https://copro.com.ar/PIDs_OBD-II.html. [Último acceso: 20 octubre 2020].
- [44] Forbes, «5 consejos para ahorrar hasta el 30% de combustible,» 11 septiembre 2018. [En línea]. Available: <https://www.forbes.com.mx/5-consejos-para-ahorrar-hasta-el-30-de-combustible-al-conducir/>. [Último acceso: 14 octubre 2020].
- [45] N. Molina, «¿Qué es el estado del arte?,» *Ciencia & Tecnología para la Salud Visual y Ocular*, vol. 12, p. 73, 2005.



- [46] N. Cross, Métodos de diseño, estrategias para el diseño de productos, Ciudad de México: LIMUSA, 2013.
- [47] Ford Mx, «7 Consejos Para Ahorrar Gasolina en tu Auto,» [En línea]. Available: <https://www.ford.mx/blog/experto/ahorra-gasolina-7-consejos-201810/>. [Último acceso: 14 octubre 2020].
- [48] Nissan, «Versa 2020,» [En línea]. Available: https://www.nissan.com.mx/content/dam/Nissan/mexico/brochures/v-drive/MY20/031019_VDrive2020_Manteleta%20BAJA.pdf.
- [49] Chevrolet, «Aveo,» [En línea]. Available: <https://www.chevrolet.com.mx/content/dam/chevrolet/na/mx/es/index/cars/2020-aveo/mov/02-pdf/2020-aveo-hoja-especificaciones.pdf>.
- [50] «Base de datosDjango,» [En línea]. Available: <https://docs.djangoproject.com/es/3.0/faq/models/>.
- [51] «Sublime Text 2,» [En línea]. Available: <https://www.sublimetext.com/2>.
- [52] «Python 3.9,» [En línea]. Available: <https://www.python.org/downloads/>.





Anexo A

Código fuente para visualizar comandos AT

```
#include <SoftwareSerial.h> // libreria que permite establecer pines digitales
    // para comunicacion serie

SoftwareSerial miBT(10, 11); // pin 10 como RX, pin 11 como TX

void setup(){
    Serial.begin(9600); // comunicacion de monitor serial a 9600 bps
    Serial.println("Listo"); // escribe Listo en el monitor
    miBT.begin(38400); // comunicacion serie entre Arduino y el modulo a 38400 bps
}

void loop(){
    if (miBT.available())    // si hay informacion disponible desde modulo
        Serial.write(miBT.read()); // lee Bluetooth y envia a monitor serial de Arduino

    if (Serial.available()) // si hay informacion disponible desde el monitor serial
        miBT.write(Serial.read()); // lee monitor serial y envia a Bluetooth
}
```

Anexo B

Métodos o Funciones principales de la librería “elmduino.h” [42]

Función	Descripción
sendQuery(byte pid)	Pregunta sobre el tipo de PID que se desea solicitar.
read()	Lee los PID de la ECU que han sido solicitados.
available()	Permite la comunicación por los puertos UART.
normalizeData(byte pid, char* data)	Realiza la conversión para obtener el valor real medido por el sensor del motor de un automóvil.
getResponse(byte& pid, char* buffer)	Obtiene las solicitudes que se le envía a la ECU del automóvil y lo almacena en un buffer.
getResult(byte& pid, int& result)	Obtiene el resultado de la conversión realizada para obtener el valor medido real.
setProtocol(byte h)	Envía el comando AT para configurar el protocolo de comunicación del sistema OBD-II.
sleep()	Estado de reposo en caso de que no exista intercambio de datos.
wakeup°	Volver a estado normal, luego de estar en estado de reposo.
begin()	Inicia la comunicación OBD-II a una velocidad de serial de 38400.
init(byte protocol)	Selecciona el protocolo de comunicación del sistema OBD-II de forma automática.

Tabla 14. Métodos o Funciones principales de la librería “elmduino.h” [42]

Esta librería fue desarrollada por Stanley Huang.

Anexo C

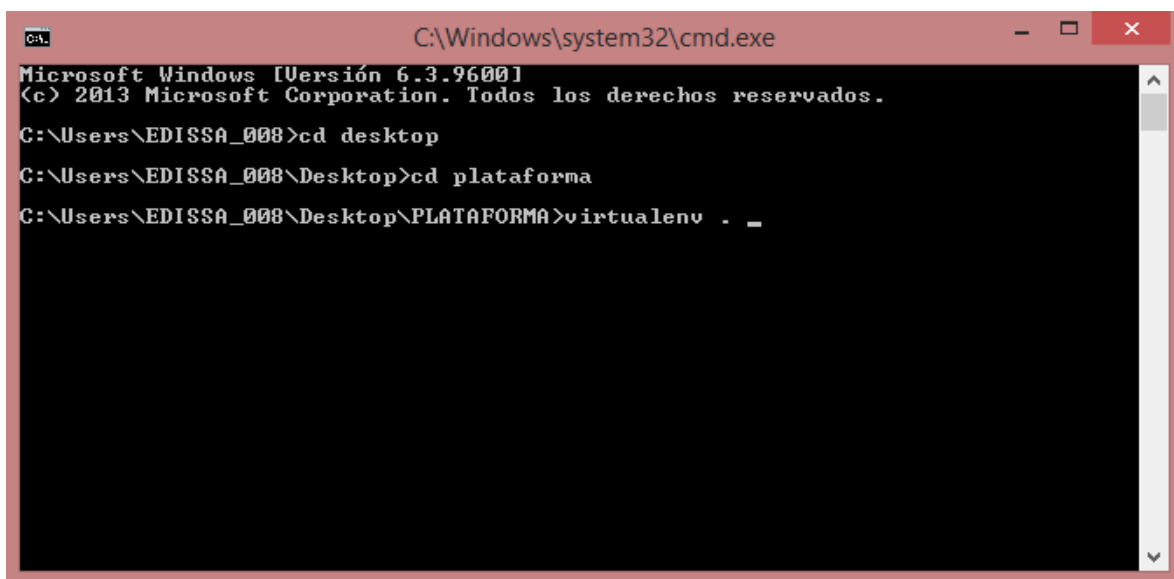
Instalación y creación del entorno virtual para desarrollar la plataforma

Se debe tener instalado previamente el lenguaje de programación Python en su versión más reciente (en nuestro caso es la versión 3.9), si se tiene versiones anteriores puede ocasionar un problema de incompatibilidad ya que django se descarga en su versión más reciente.

Teniendo instalado Python se procede a abrir la carpeta mediante un CMD y escribir el siguiente comando:

`virtualenv .`

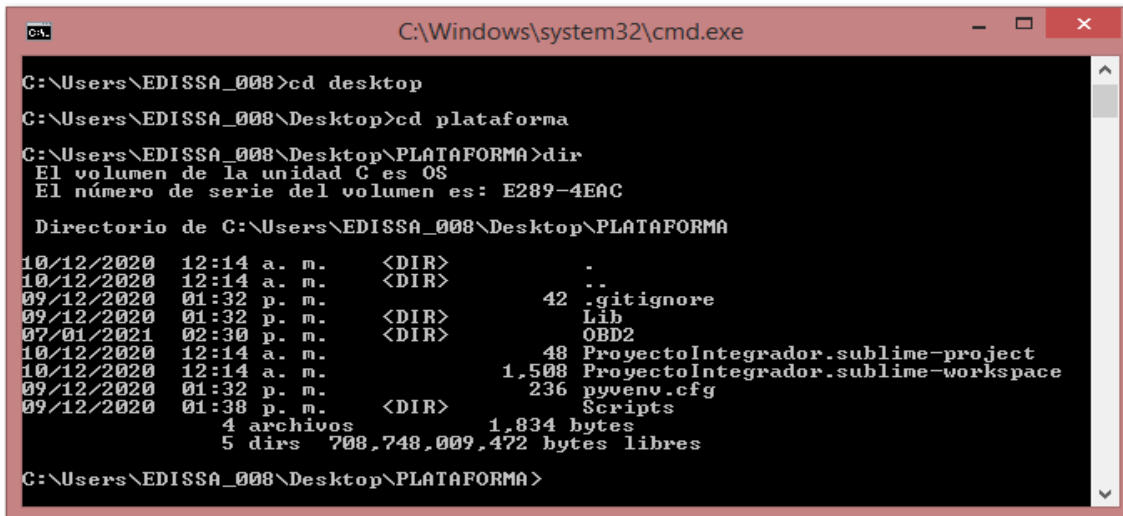
Esta Instrucción es para que en nuestra carpeta en donde planeamos guardar nuestro proyecto se cree nuestro entorno virtual.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.
C:\Users\EDISSA_008>cd desktop
C:\Users\EDISSA_008\Desktop>cd plataforma
C:\Users\EDISSA_008\Desktop\PLATAFORMA>virtualenv . _
```

Ilustración 41. Comando en CMD para crear el virtualización

Al escribir la instrucción y dar Enter se tiene que crear el entorno virtual, y sabemos que el entorno se creó al escribir el comando *dir* en nuestra carpeta llamada PLATAFORMA, si al escribir el comando vemos que dentro de la carpeta se crearon más archivos quiere decir que nuestro entorno virtual se ha creado exitosamente.



```
C:\Windows\system32\cmd.exe

C:\Users\EDISSA_008>cd desktop
C:\Users\EDISSA_008\Desktop>cd plataforma
C:\Users\EDISSA_008\Desktop\PLATAFORMA>dir
El volumen de la unidad C es OS
El número de serie del volumen es: E289-4EAC

Directorio de C:\Users\EDISSA_008\Desktop\PLATAFORMA

10/12/2020  12:14 a. m.      <DIR>          .
10/12/2020  12:14 a. m.      <DIR>          ..
09/12/2020  01:32 p. m.             42  .gitignore
09/12/2020  01:32 p. m.             Lib
07/01/2021  02:30 p. m.             <DIR>          OBD2
10/12/2020  12:14 a. m.             48  ProyectoIntegrador.sublime-project
10/12/2020  12:14 a. m.          1,508  ProyectoIntegrador.sublime-workspace
09/12/2020  01:32 p. m.             236  pyvenv.cfg
09/12/2020  01:38 p. m.             <DIR>          Scripts
                                4 archivos          1,834 bytes
                                5 dirs  708,748,009,472 bytes libres

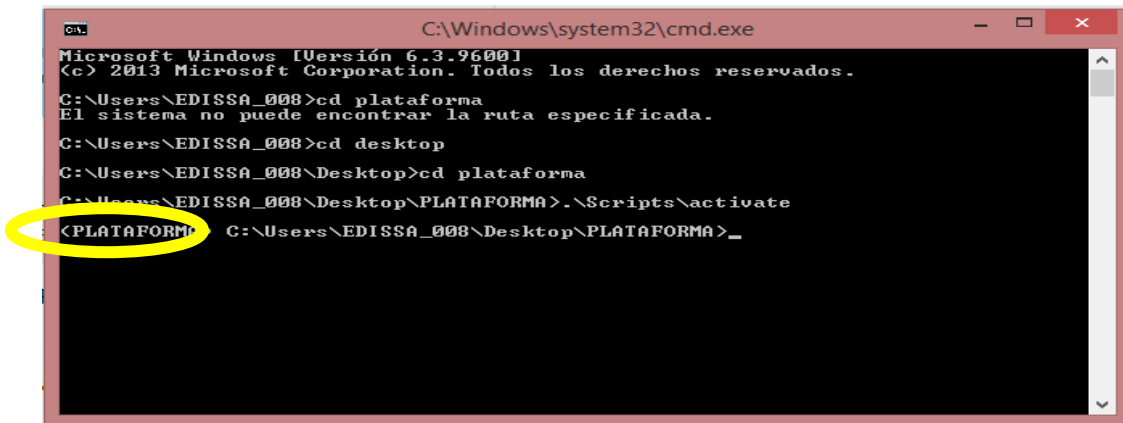
C:\Users\EDISSA_008\Desktop\PLATAFORMA>
```

Ilustración 42. Comprobación de creación exitosa del virtualenv

Después de tener instalada nuestro entorno virtual tenemos que activarlo para así poder comenzar a trabajar con django y nuestro editor de texto Sublime Text 2. Ponemos el siguiente comando para activar el entorno virtual:

```
.\Scripts\activate
```

Al dar entre a la instrucción se ve reflejado un cambio en el CMD en la carpeta PLATAFORMA si sucede ese cambio se ha creado y activado con éxito nuestro entorno virtual.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.
C:\Users\EDISSA_008>cd plataforma
El sistema no puede encontrar la ruta especificada.
C:\Users\EDISSA_008>cd desktop
C:\Users\EDISSA_008\Desktop>cd plataforma
C:\Users\EDISSA_008\Desktop\PLATAFORMA>.\Scripts\activate
C:\Users\EDISSA_008\Desktop\PLATAFORMA>_
```

Ilustración 43. Activación del entorno virtual



Apéndice A

PIDS OBD-II MODO 1 [43]

PID (0x)	Bytes de datos devuelto	Descripción	Unidades	Fórmula
00	4	PID apoyado [01-20]		Bit codificado [A7...D0] == [PID \$01.. PID \$20]
01	4	Estado del monitor desde DTCs borra. (Incluye estado (MIL) la lámpara indicadora de malfuncionamiento y número de DTC).		Bit codificado.
02	2	Congelación DTC		
03	2	Estado del sistema de combustible		Bit codificado.
04	1	Calcula el valor de la carga del motor	%	$A * 100/255$

05	1	Temperatura del refrigerante del motor	° C	A-40
06	1	A corto plazo combustible % ajuste — Banco 1	%	$(A-128) * 100/128$
07	1	Largo plazo combustible % ajuste — Banco 1	%	$(A-128) * 100/128$
08	1	A corto plazo combustible % ajuste — banco 2	%	$(A-128) * 100/128$
09	1	Largo plazo combustible % ajuste — banco 2	%	$(A-128) * 100/128$
0A	1	Presión de combustible	kPa (calibre)	$A * 3$
0B	1	Presión absoluta del múltiple de admisión	kPa (absoluta)	A
0C	2	RPM del motor	rpm	$((A*256) + B) / 4$

0D	1	Velocidad del vehículo	km/h	A
0E	1	Avance de sincronización	° en relación con cilindro #1	(A-128) / 2
0F	1	Temperatura del aire de admisión	° C	A-40
10	2	Flujo de aire MAF	gramos/seg.	((A*256) + B) / 100
11	1	Posición del acelerador	%	A * 100/255
12	1	Estado de aire secundario ordenada		Bit codificado.
13	1	Sensores de oxígeno presentes		[A0...A3] == Banco 1, sensores 1-4. [A4...A7] == Banco 2...
14	2	Bancada 1, Sensor 1: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	Voltios %	A/200 (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)

15	2	Bancada 1, Sensor 2: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	Voltios %	$A/200$ $(B-128) * 100/128$ (si B == \$FF, sensor no se utiliza en calc trim)
16	2	Bancada 1, Sensor 3: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	Voltios %	$A/200$ $(B-128) * 100/128$ (si B == \$FF, sensor no se utiliza en calc trim)
17	2	Bancada 1, Sensor 4: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	Voltios %	$A/200$ $(B-128) * 100/128$ (si B == \$FF, sensor no se utiliza en calc trim)
18	2	Bancada 2, Sensor 1: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	Voltios %	$A/200$ $(B-128) * 100/128$ (si B == \$FF, sensor no se utiliza en calc trim)

19	2	Bancada 2, Sensor 2: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	Voltios %	$A/200$ (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)
1A	2	Bancada 2, Sensor 3: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	Voltios %	$A/200$ (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)
1B	2	Bancada 2, Sensor de 4: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	Voltios %	$A/200$ (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)
1C	1	Este vehículo se ajusta a las normas OBD		Bit codificado. Vea a continuación
1D	1	Sensores de oxígeno presentes		Similar a PID 13, pero [A0...A7] == [B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2]

1E	1	Estado de la entrada auxiliar		A0 == estado poder tomar Off (PTO) (1 == activo) [A1...A7] no se utiliza
1F	2	Tiempo de ejecución desde el motor	segundos	$(A * 256) + B$
20	4	PID apoyado [21-40]		Bit codificado [A7...D0] == [PID \$21.. PID \$40] Vea a continuación
21	2	Distancia recorrida con lámpara indicadora de malfuncionamiento (MIL) en	km	$(A * 256) + B$
22	2	Presión de carril de combustible (en relación al vacío del múltiple)	kPa	$((A * 256) + B) * 0.079$
23	2	Combustible presión del carril (diesel o gasolina directo inyectar)	kPa (calibre)	$((A * 256) + B) * 10$

24	4	O2S1_WR_lambda(1): Relación de equivalencia Voltaje	N / A V	$((A*256) + B) * 2/65535$ o $((A*256) + B) / 32768$ $((C*256) + D) * 8/65535$ o $((C*256) + D) / 8192$
25	4	O2S2_WR_lambda(1): Relación de equivalencia Voltaje	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
26	4	O2S3_WR_lambda(1): Relación de equivalencia Voltaje	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
27	4	O2S4_WR_lambda(1): Relación de equivalencia Voltaje	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
28	4	O2S5_WR_lambda(1): Relación de equivalencia Voltaje	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
29	4	O2S6_WR_lambda(1): Relación de equivalencia Voltaje	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$

2A	4	O2S7_WR_lambda(1): Relación de equivalencia Voltaje	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
2B	4	O2S8_WR_lambda(1): Relación de equivalencia Voltaje	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
2C	1	Ordenado E GR	%	$A * 100/255$
2D	1	Error EGR	%	$(A-128) * 100/128$
2E	1	Purga ordenada por evaporación	%	$A * 100/255$
2F	1	Entrada de nivel de combustible	%	$A * 100/255$
30	1	# de calentamiento o ya que borra códigos	N / A	A
31	2	Distancia recorrida desde que	km	$(A * 256) + B$

		borra códigos		
32	2	Presión de Vapor de gas sistema	Pa	$((A*256) + B) / 4$ (A y B son complemento a dos firmado)
33	1	Presión barométrica	kPa (absoluta)	A
34	4	O2S1_WR_la mbda(1): Relación de equivalencia Corriente	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
35	4	O2S2_WR_la mbda(1): Relación de equivalencia Corriente	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
36	4	O2S3_WR_la mbda(1): Relación de equivalencia Corriente	N / A mA	$((A*256) + B) / 32768$ $((C*256) + D) / 256-128$
37	4	O2S4_WR_la mbda(1): Relación de equivalencia Corriente	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
38	4	O2S5_WR_la mbda(1): Relación de	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$

		equivalencia Corriente		
39	4	O2S6_WR_lambda(1): Relación de equivalencia Corriente	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
3A	4	O2S7_WR_lambda(1): Relación de equivalencia Corriente	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
3B	4	O2S8_WR_lambda(1): Relación de equivalencia Corriente	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
3C	2	Temperatura del catalizador Bancada 1, Sensor 1	° C	$((A*256) + B) / 10-40$
3D	2	Temperatura del catalizador Bancada 2, Sensor 1	° C	$((A*256) + B) / 10-40$
3E	2	Temperatura del catalizador	° C	$((A*256) + B) / 10-40$

		Bancada 1, Sensor 2		
3F	2	Temperatura del catalizador Bancada 2, Sensor 2	° C	$((A*256) + B) / 10-40$
40	4	PID apoyado [41-60]		Bit codificado [A7...D0] == [PID \$41... PID \$60] Vea a continuación
41	4	Supervisar el estado de este ciclo de coche		Bit codificado. Vea a continuación
42	2	Voltaje del módulo de control	V	$((A*256) + B) / 1000$
43	2	Valor absoluto de la carga	%	$((A*256) + B) * 100/255$
44	2	Relación de equivalencia de comandos	N / A	$((A*256) + B) / 32768$
45	1	Posición relativa del acelerador	%	$A * 100/255$

46	1	Temperatura del aire ambiente	° C	A-40
47	1	Posición del acelerador absoluta B	%	$A * 100/255$
48	1	Posición absoluta del acelerador C	%	$A * 100/255$
49	1	Posición del pedal acelerador D	%	$A * 100/255$
4A	1	Posición del pedal del acelerador E	%	$A * 100/255$
4B	1	Posición del pedal acelerador F	%	$A * 100/255$
4C	1	Actuador de válvula reguladora ordenada	%	$A * 100/255$
4D	2	Ejecutan con MIL	minutos	$(A * 256) + B$
4E	2	Tiempo que borra los	minutos	$(A * 256) + B$

		códigos de problemas		
4F	4	Valor máximo para la relación de equivalencia, voltaje del sensor de oxígeno, actual del sensor del oxígeno y la presión absoluta del múltiple de admisión	V, mA, kPa	A, B, C, D * 10
50	4	Valor máximo de caudal de masa de aire sensor de flujo de aire	g/s	A * 10, B, C y D están reservados para uso futuro
51	1	Tipo de combustible		De tabla de tipo de combustible vea a continuación
52	1	% Etanol combustible	%	A * 100/255
53	2	Sistema Evap absoluto presión de Vapor	kPa	$((A*256) + B) / 200$

54	2	Presión de vapor del sistema EVAP	Pa	$((A*256) + B) - 32767$
55	2	Sensor de oxígeno secundario recortar Banco 1 y 3 a corto plazo	%	$(A-128) * 100/128$ $(B-128) * 100/128$
56	2	Largo plazo sensor de oxígeno secundario ajuste Banco 1 y 3 del Banco	%	$(A-128) * 100/128$ $(B-128) * 100/128$
57	2	Sensor de oxígeno secundario recortar banco 2 y 4 a corto plazo	%	$(A-128) * 100/128$ $(B-128) * 100/128$
58	2	Largo plazo sensor de oxígeno secundario ajuste banco 2 y 4 del Banco	%	$(A-128) * 100/128$ $(B-128) * 100/128$
59	2	Presión del carril del	kPa	$((A*256) + B) * 10$

		combustible (absoluta)		
5A	1	Posición del pedal acelerador relativa	%	$A * 100/255$
5B	1	Batería híbrida vida restante	%	$A * 100/255$
5C	1	Temperatura de aceite del motor	° C	$A - 40$
5D	2	Tiempo de inyección de combustible	°	$((A*256) + B) - 26,880 / 128$
5E	2	Tasa de combustible del motor	L. / h	$((A*256) + B) * 0.05$
5F	1	Requisitos de emisiones que el vehículo está diseñado		Bit codificado
60	4	PID apoyado [61-80]		Bit codificado [A7...D0] == [PID \$61.. PID \$80] Vea a continuación
61	1	Motor de la demanda del conductor -	%	$A-125$

		por ciento del esfuerzo de torsión		
62	1	Motor real - por ciento del esfuerzo de torsión	%	A-125
63	2	Par motor referencia	Nm	$A * 256 + B$
64	5	Datos de porcentaje de esfuerzo de torsión del motor	%	A-125 inactivo Punto B-125 motor 1 Punto C-125 motor 2 Punto D-125 motor 3 Punto E-125 motor 4
65	2	Auxiliar de entrada / salida apoyada		Bit codificado
66	5	Sensor de flujo de masa de aire		
67	3	Temperatura del refrigerante del motor		
68	7	Sensor de temperatura de aire de admisión		

69	7	EGR ordenada y Error de EGR		
6A	5	Control de flujo de aire de admisión Diesel ordenada y admisión relativa posición de flujo de aire		
6B	5	Temperatura de recirculación de gases de escape		
6C	5	Control del actuador del acelerador ordenada y la posición relativa del acelerador		
6D	6	Sistema de control de presión de combustible		
6E	5	Sistema de control de presión de inyección		

6F	3	Presión de entrada del turbocompresor		
70	9	Control de refuerzo de presión		
71	5	Control de turbo de geometría variable (VGT)		
72	5	Control de la válvula de derivación		
73	5	Presión de escape		
74	5	Turbocompresor RPM		
75	7	Temperatura del turbocompresor		
76	7	Temperatura del turbocompresor		

77	5	Carga la temperatura del aire del refrigerador (CACT)		
78	9	(EGT) temperatura del Gas de escape Banco 1		Especial PID. Vea a continuación
79	9	(EGT) temperatura del Gas de escape banco 2		Especial PID. Vea a continuación
7A	7	Filtro de partículas diesel (DPF)		
7B	7	Filtro de partículas diesel (DPF)		
7C	9	Diesel Particulate temperatura filtro (DPF)		
7D	1	Estado de zona de control de NOx NTE		

7E	1	Estado de zona de control PM NTE		
7F	13	Motor de tiempo de ejecución		
80	4	PID apoyado [81 - A0]		Bit codificado [A7...D0] == [PID \$81.. PID \$A0] Vea a continuación
81	21	Motor de tiempo de ejecución para auxiliar Device(AECD) de Control de emisiones		
82	21	Motor de tiempo de ejecución para auxiliar Device(AECD) de Control de emisiones		
83	5	Sensor de NOx		
84		Temperatura de superficie múltiple		

85		Sistema reactivo de NOx		
86		Sensor de partículas (PM)		
87		Presión absoluta del múltiple de admisión		
A0	4	PID apoyado [A1 - C0]		Bit codificado [A7...D0] == [PID \$A1..PID \$C0] Vea a continuación
C0	4	PID apoyado [C1 - E0]		Bit codificado [A7...D0] == [PID \$C1..PID \$E0] Vea a continuación
C3	?	?	?	Devuelve datos numerosos, incluyendo conducir condición ID y motor de velocidad *
C4	?	?	?	B5 es ocioso solicitud de motor B6 es petición parada motor *
PID (hexadecimal)	Bytes de datos devuelto	Descripción	Unidades	Fórmula^[a]



Apéndice B



Hoja de datos circuito integrado ELM327 [14]

ELM327

OBD to RS232 Interpreter

Description

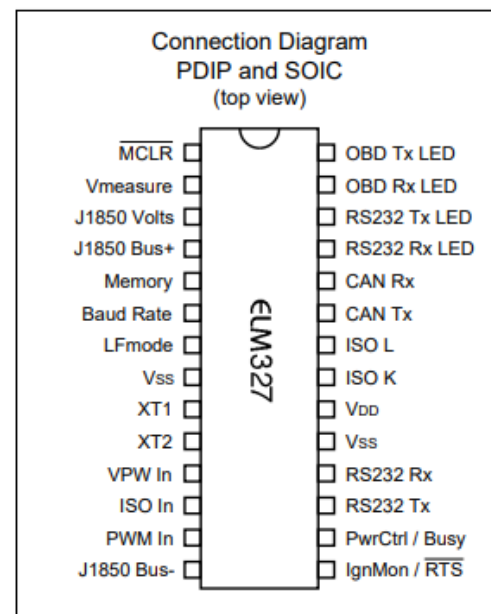
Almost all of the automobiles produced today are required, by law, to provide an interface for the connection of diagnostic test equipment. The data transfer on these interfaces follow several standards, but none of them are directly usable by PCs or smart devices. The ELM327 is designed to act as a bridge between these On-Board Diagnostics (OBD) ports and a standard RS232 serial interface.

In addition to being able to automatically detect and interpret nine OBD protocols, the ELM327 also provides support for high speed communications, a low power sleep mode, and the J1939 truck and bus standard. It is also completely customizable, should you wish to alter it to more closely suit your needs.

The following pages discuss all of the ELM327's features in detail, how to use it and configure it, as well as providing some background information on the protocols that are supported. There are also schematic diagrams and tips to help you to interface to microprocessors, construct a basic scan tool, and to use the low power mode.

Features

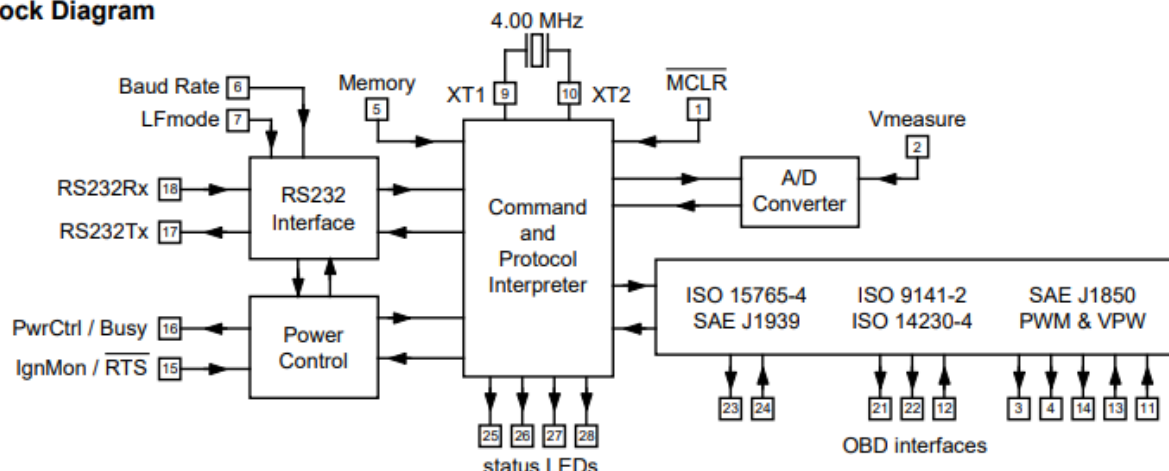
- Power Control with standby mode
- Universal serial (RS232) interface
- Automatically searches for protocols
- Fully configurable with AT commands
- Low power CMOS design



Applications

- Diagnostic trouble code readers
- Automotive scan tools
- Teaching aids

Block Diagram





ELM327

Electrical Characteristics

All values are for operation at 25°C and a 5V supply, unless otherwise noted. For further information, refer to note 1 below.

Characteristic	Minimum	Typical	Maximum	Units	Conditions
Supply voltage, V_{DD}	4.2	5.0	5.5	V	
V_{DD} rate of rise	0.05			V/ms	see note 2
Average current, I_{DD}		12		mA	ELM327 device only - does not include any load currents
normal					
low power		0.15		mA	
Input logic levels					Pins 5, 6, 7, and 24 only
low	V_{SS}		0.8	V	
high	3.0		V_{DD}	V	
Schmitt trigger input thresholds		2.9	4.0	V	Pins 1, 11, 12, 13, 15 and 18 only
rising					
falling	1.0	1.5		V	
Output low voltage		0.3		V	current (sink) = 10 mA
Output high voltage		4.4		V	current (source) = 10 mA
Brown-out reset voltage	2.65	2.79	2.93	V	
A/D conversion time		9		msec	AT RV to beginning of response
Pin 18 wake pulse duration	128			μsec	to wake from Low Power mode
IgnMon debounce time	50	65		msec	
AT LP to PwrCtrl output time		1.0		sec	
LP ALERT to PwrCtrl output time		2.0		sec	
Reset time		800		msec	Measured from the end of the command to the start of the ID message (ELM327 v2.1)
AT Z					
AT WS		2		msec	

Notes:

1. This integrated circuit is based on Microchip Technology Inc.'s PIC18F2480 device. For more detailed device specifications, and possibly clarification of those given, please refer to the Microchip documentation (available at www.microchip.com).
2. This spec must be met in order to ensure that a correct power on reset occurs. It is quite easily achieved using most common types of supplies, but may be violated if one uses a slowly varying supply voltage, as may be obtained through direct connection to solar cells or some charge pump circuits.



ELM327

AT Commands

Several parameters within the ELM327 can be adjusted in order to modify its behaviour. These do not normally have to be changed before attempting to talk to the vehicle, but occasionally the user may wish to customize these settings – for example by turning the character echo off, adjusting a timeout value, or changing the header bytes. In order to do this, internal 'AT' commands must be used.

Those familiar with PC modems will immediately recognize AT commands as a standard way in which modems are internally configured. The ELM327 uses essentially the same method, always watching the data sent by the PC, looking for messages that begin with the character 'A' followed by the character 'T'. If found, the next characters will be interpreted as an internal configuration or 'AT' command, and will be executed upon receipt of a terminating carriage return character. If the command is just a setting change, the ELM327 will reply with the characters 'OK', to say that

it was successfully completed.

Some of the following commands allow passing numbers as arguments in order to set the internal values. These will always be hexadecimal numbers which must generally be provided in pairs. The hexadecimal conversion chart in the OBD Commands section (page 30) may be helpful if you wish to interpret the values. Also, you should be aware that for the on/off types of commands, the second character is the number 1 or the number 0, the universal terms for on and off.

The remainder of this page, and the two pages following provide a summary of all of the commands that the current version of the ELM327 recognizes. A more complete description of each command begins on page 12. Note that the settings which are shown with an asterisk (*) are the default values.

AT Command Summary

General Commands

<CR>	repeat the last command
BRD hh	try Baud Rate Divisor hh
BRT hh	set Baud Rate Timeout
D	set all to Defaults
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 identifier
@3 ccccccccccc	store the @2 identifier

Programmable Parameter Commands

PP xx OFF	disable Prog Parameter xx
PP FF OFF	all Prog Parameters 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

Voltage Reading Commands

CV dddd	Calibrate the Voltage to dd.dd volts
CV 0000	restore CV value to factory setting
RV	Read the input Voltage

Other

IGN	read the IgnMon input level
------------	-----------------------------



ELM327

AT Command Summary (continued)

OBD Commands

AL	Allow Long (>7 byte) messages
AMC	display Activity Monitor Count
AMT hh	set the Activity Mon Timeout to hh
AR	Automatically Receive
AT0, 1, 2	Adaptive Timing off, auto1*, auto2
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	Responses off, or on*
RA hh	set the Receive Address to hh
S0, S1	printing of Spaces off, or on*
SH xyz	Set Header to xyz
SH xxyyzz	Set Header to xxyyzz
SH wwxyyzz	Set Header to wwxyyzz
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

J1850 Specific Commands (protocols 1 and 2)

IFR0, 1, 2	IFRs off, auto*, or on
IFR H, S	IFR value from Header* or Source

ISO Specific Commands (protocols 3 to 5)

FI	perform a Fast Initiation
IB 10	set the ISO Baud rate to 10400*
IB 48	set the ISO Baud rate to 4800
IB 96	set the ISO 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
SW 00	Stop sending Wakeup messages
WM [1 - 6 bytes]	set the Wakeup Message

CAN Specific Commands (protocols 6 to C)

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 hhh	set the ID Mask to hhh
CM hhhhhhhh	set the ID Mask to hhhhhhhh
CP hh	set CAN Priority to hh (29 bit)
CRA	reset the Receive Address filters
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*
CTM1	set Timer Multiplier to 1*
CTM5	set Timer Multiplier to 5

**ELM327****AT Command Summary (continued)****CAN Specific Commands (continued)**

D0, D1	display of the DLC off*, or on
FC SM h	Flow Control, Set the Mode to h
FC SH hhh	FC, Set the Header to hhh
FC SH hhhhhhhh	Set the Header to hhhhhhhh
FC SD [1 - 5 bytes]	FC, Set Data to [...]
PB xx yy	Protocol B options and baud rate
RTR	send an RTR message
V0, V1	use of Variable DLC off*, or on

J1939 CAN Specific Commands (protocols A to C)

DM1	monitor for DM1 messages
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 hhhh n	" " and get n messages
MP hhhhhh	Monitor for PGN hhhhhh
MP hhhhhh n	" " and get n messages



Apéndice C

Lista completa sobre ahorro de combustible [3] [44]

- “Pisar a fondo el acelerador aumenta hasta cuatro veces el consumo de gasolina, así como frenar rápido. En tanto, acelerar gradualmente sobre una pendiente ayudará a reducir el desgaste del auto”.
- “En los autos con transmisión manual, es importante colocar la velocidad correcta para sacar un mejor provecho del rendimiento del combustible y tratar de llevar lo indispensable en la cajuela, ya que, por cada 50 kilos extras, el consumo aumenta”.
- “Aunque a veces es inevitable prender el aire acondicionado, esta acción aumenta hasta 10% el consumo de combustible, así como viajar a altas velocidades”.
- “La velocidad crucero es un sistema inteligente que viene integrado en carros de gamas cada vez más bajas, esto, permitirá mantener una velocidad constante en las carreteras y, por ende, ahorra combustible.
- “Ir con la marcha más larga y no revolucionar el motor”.
- “Optimiza al máximo el cambio de marchas. Circula el mayor tiempo posible en las relaciones más largas y a bajas revoluciones (en la ciudad, siempre que sea posible, utilizar la 4ª o 5ª marcha, respetando los límites de velocidad)”.
- “Sobre el papel, el par motor de un coche de gasolina, el momento óptimo se sitúa entre 2.000 y 2.500 rpm. Si tu coche no dispone de cuentarrevoluciones, el sonido del motor te puede servir como referencia”
- “Arrancar sin pisar el acelerador y usa la primera velocidad (la más potente de todas) sólo para el inicio de la marcha”.
- “Cambia a segunda enseguida de pasar los 5 metros iniciales (algunos vehículos indican con una señal en el tablero cuando es tiempo de hacer un cambio de velocidad)”.
- “Entre más baja sea la velocidad, más alta será la potencia necesaria y, por lo tanto, los requerimientos de gasolina. Manejar a partir de la segunda velocidad, evitando las aceleraciones bruscas, te puede ahorrar hasta 11% del consumo de combustible”.

