



FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Trabajo de fin de Carrera titulado:

**“DISEÑO DE UN DATALOGGER PARA PRUEBA DE RUTA EN
AUTOMÓVILES”**

Realizado por:

BRYAN TEMÍSTOCLES MACÍAS MUÑOZ

Director del proyecto:

GUSTAVO ADOLFO MORENO

Como requisito para la obtención del título de:

INGENIERO EN MECÁNICA AUTOMOTRIZ

QUITO, 04 de agosto de 2021

DECLARACIÓN JURAMENTADA

Yo, BRYAN TEMÍSTOCLES MACÍAS MUÑOZ, ecuatoriano, con Cédula de ciudadanía N° 2200218671, declaro bajo juramento que el trabajo aquí desarrollado es de mi autoría, que no ha sido presentado anteriormente para ningún grado o calificación profesional, y se basa en las referencias bibliográficas descritas en este documento.

A través de esta declaración, cedo los derechos de propiedad intelectual a la UNIVERSIDAD INTERNACIONAL SEK, según lo establecido en la Ley de Propiedad Intelectual, reglamento y normativa institucional vigente.



BRYAN TEMÍSTOCLES MACÍAS MUÑOZ

C.I.: 2200218671

DECLARACIÓN DEL DIRECTOR DE TESIS

Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.



GUSTAVO ADOLFO MORENO

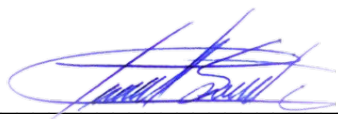
Master in Science at Technology Management

LOS PROFESORES INFORMANTES:

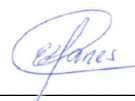
PAOLO SALAZAR

EDILBERTO LLANES

Después de revisar el trabajo presentado lo han calificado como apto para su defensa oral ante el tribunal examinador.



Ing. Paolo Salazar



Ing. Edilberto Llanes

Quito, 04 de agosto de 2021

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.



Firmado electrónicamente por:

**BRYAN
TEMISTOCLES
MACIAS MUÑOZ**

BRYAN TEMISTOCLES MACÍAS MUÑOZ

C.I.: 2200218671

DEDICATORIA

A Dios y a todos aquellos que me ayudaron en este camino.

AGRADECIMIENTOS

Temístocles Macías.

Juan Pérez, Beatríz Macías, Nicolás Peréz y Alexander Pérez. Quienes me acogieron en su hogar para cumplir este sueño.

A mis Padres, Familia e Itaty Sánchez, por todo el apoyo brindado.

Gustavo Moreno mi amigo y profesor a quien admiro.

Tabla de contenido

Resumen.....	3
Abstract.....	4
Introducción.....	5
Técnica On-Board.....	7
Técnica de la persecución del vehículo.....	8
Objetivo General.....	12
Objetivo Específico.....	12
Método.....	13
Diseño electrónico.....	13
Diseño Software.....	16
Configuración Inicial.....	16
Configuración HC-05.....	16
Software de adquisición de datos.....	20
Comprobación de conexión y uso de librerías.....	20
Software análisis de datos.....	25
Instalación Jupyter Notebook.....	25
Instalación de Pandas.....	26
Instalación de Pandastable.....	26
Diseño mecánico.....	26
Costos de producción.....	29
Experimentación.....	29
Experimento 1.....	30
Experimento 2.....	30

Experimento 3.....	31
Resultados.....	33
Experimento 1.....	34
Experimento 2.....	36
Experimento 3.....	37
Discusión.....	40
Resultados.....	41
Conclusiones.....	41
Bibliografía.....	42
Anexo 1.....	46
Anexo 2.....	47
Anexo 3.....	50
Anexo 4.....	51
Anexo 5.....	52

Resumen.

En la actualidad los científicos en busca de mejorar la eficiencia de los motores de combustión interna buscan caracterizar las rutas a través de un concepto denominado ciclos de conducción con propósito de medir los niveles de emisiones y cantidad de combustible utilizado por un vehículo en una ruta específica. Para esto se puede encontrar aplicaciones para Android que permiten ser enlazadas al OBD-II del vehículo midiendo en tiempo real variables como velocidad, consumo de combustible etc. Sin embargo, no es posible almacenar o graficar los datos que son arrojados por los sensores del vehículo siendo un limitante para la investigación de los ciclos de conducción. El presente proyecto tiene como objetivo desarrollar un prototipo de adquisición, almacenamiento y análisis de datos “datalogger” mediante el uso de microcontroladores, sistema OBD-II, y sensores para el análisis del comportamiento de vehículos en pruebas de ruta. Para poder interpretar estos datos de manera más amigable para el usuario se realizará una interfaz que permita visualizar, escoger y graficar los valores que sean de interés mediante el uso de Arduino y Python a través de la librería Pandastable que permita analizar estos valores dejando como principales resultados la caracterización de una prueba de ruta con 3000 muestras donde se evidencia los valores de RPM, temperatura de motor y cámara de admisión. Concluyendo que esta herramienta puede ser utilizada por los científicos ya que permite interactuar entre distintos modelos de gráficas y variables

Palabras clave: Eficiencia, Rutas, Ciclos de conducción, Android, OBD-II, Interfaz

Abstract.

Currently, scientists seeking to improve the efficiency of internal combustion engines search to characterize the routes through a concept called driving cycles in order to measure the levels of emissions and amount of fuel used by a vehicle on a specific route. . For this, you can find applications for Android that allow them to be linked to the OBD-II of the vehicle, measuring variables such as speed, fuel consumption, etc. in real time. However, it is not possible to store or graph the data that are thrown by the vehicle's sensors, being a limitation for the investigation of driving cycles. The objective of this project is to develop a prototype for the acquisition, storage and analysis of data "datalogger" through the use of microcontrollers, the OBD-II system, and sensors for the analysis of the behavior of vehicles in road tests. In order to interpret these data in a more user-friendly way, an interface will be executed that will allow to visualize, choose and graph the values that are of interest through the use of Arduino and Python through the Pandastable library that allows to analyze these values leaving as main results the characterization of a road test with 3000 samples where the values of RPM, engine temperature and intake chamber are evidenced. Concluding that this tool can be used by scientists and that it allows interaction between different models of graphs and variables.

Keywords: Efficiency, Routes, Drive Cycles, Android, OBD-II, Interface

Introducción

A medida que el mundo evoluciona tecnológicamente, las plantas automatizadas se incrementan y gracias a ellas las nuevas tecnologías de producción y mano de obra barata, creando productos “buenos y económicos” para los consumidores (Paz, 2005). Sin embargo, debido a estos factores tecnológicos y al constante crecimiento a escala mundial ha provocado que países como Ecuador, dentro de la industria automotriz en su última década, sufra un crecimiento de 1.4 millones de automóviles siendo la ciudad de Quito la más afectada con un total de 540.827 vehículos matriculados para el 2018 (El comercio, 2019).

Debido a este acelerado crecimiento del parque automotor surge una problemática de contaminación ambiental dado que motores de combustión interna emiten ciertos contaminantes gaseosos primarios, como el dióxido de azufre (SO₂), los óxidos de nitrógeno (NO_x) y el monóxido de carbono (CO). Además, todos los procesos de combustión producen partículas, tan pequeñas que pueden ser inhaladas como emisiones primarias, como es el caso de hollín producido por los motores diésel, o bien como partículas secundarias dadas por la transformación atmosférica como las partículas de sulfato formadas por la quema de combustible que contenga azufre como lo muestran Pineda, Muñoz, y Gil (2018)

Para identificar, estos gases de escape de un motor a gasolina el CO₂ representa al menos un 14 %, mientras que en un motor diésel ese porcentaje desciende al 12 %. Esto se debe a la variación en la composición química del propio combustible, ya que el diésel (C₁₂H₂₆) tiene una presencia mayor de átomos de carbono que la gasolina (C₇H₁₆). Por ende, si a la mayor emisión de CO₂ de gasolina se le añade que su consumo por kilómetro recorrido también es mayor, tenemos que el motor gasolina es más contaminante refiriéndose a emisiones de CO₂ (Gómez, 2020)

Como una de las consecuencias de dichas emisiones se origina el efecto invernadero, el cual menciona a breves rasgos que los medios de transporte emiten gases que guardan calor en la atmósfera y por tanto favorecen al calentamiento global (predominantemente dióxido de carbono). Este cambio climático se menciona que impactará de manera característica a las diferentes regiones del mundo, grupos etarios, socioeconómicos y hasta los géneros, acentuándose las desigualdades en salud, estado social y acceso a alimentos adecuados, servicios básicos y otros recursos. Otra consecuencia, demostrada por varios estudios epidemiológicos, como lo menciona Romero, et al. (2006) muestran que la exposición a estos contaminantes y el calentamiento global, ha dado paso a un incremento en la incidencia de asma, severidad en el deterioro de la función pulmonar, así como mayor gravedad en las enfermedades respiratorias de niños y adolescentes.

A partir de esta problemática se busca reducir las emisiones de CO₂ a la atmósfera, comenzado a utilizar combustibles que llevan el prefijo bio como: biodiesel, bioetanol, Bio-ETBE. Estos biocombustibles de origen vegetal no contribuyen a aumentar el efecto invernadero, ya que la planta ha retirado previamente de la atmósfera el CO₂ que posteriormente emitirá el vehículo por el tubo de escape (Ortiz, 2010).

Entonces, en su búsqueda por la eficiencia los científicos realizan la caracterización de rutas mediante ciclos de conducción que se definen como una gráfica estadística de la velocidad en ese instante con respecto al tiempo, obtenido dentro de un área determinada. Se han desarrollado diversos ciclos de conducción para vehículos livianos, camiones, autobuses y motocicletas. Actualmente existen dos tipos de ciclos de conducción, de acuerdo a su creación: Los ciclos legislativos, son de carácter estatal, por lo cual controlan las emisiones contaminantes que proceden de los motores a combustión. Los ciclos no legislativos, sirven para el análisis de consumo de combustible y emisiones contaminantes del motor, dentro de

laboratorios (Hurtado Gómez, 2014). Existen ciclos de conducción que incorporan series de tráfico fluido y congestionado o combinadas (Borja Pintos, 2011).

En Estados Unidos, uno de los ciclos de conducción más importantes es el FTP (Federal Test Procedure), de la cual se deriva el FTP 72 y el FTP 75, siendo estos de carácter gubernamental. El motivo de su creación fue el control de las emisiones de gases contaminantes provenientes de los automotores (Hurtado Gómez, 2014). La denominación del ciclo FTP-72, también como Programa Urbano de Dinamómetro (UDDS por sus siglas en inglés, Urban Dynamometer Driving Schedule) o ciclo LA-4. Este ciclo simula una ruta urbana de 12,07 km con paradas frecuentes. La velocidad máxima es de 91.26 km/h y la velocidad promedio es de 31,6 km/h. El ciclo consta de dos fases: la primera de 505 s (5.78 km a 41,2 km/h de velocidad media) de la cual comienza con un arranque en frío y la segunda de 867 s. El mismo ciclo de conducción del motor se conoce en Australia como el ciclo ADR 27 (Australian Design Rules) y en Suecia como ciclo A10 o CVS (Constant Volume Sampler) (DieselNet, Diesel Net, 2011).

En Ecuador, para la determinación de los ciclos de conducción se obtuvieron tres ciclos aplicados a Quito: ciudad (Sentido Sur-Norte), carretera (Sentido Norte-Sur) y una combinación (Sentido Este-Oeste). Todos los ciclos fueron realizados en condiciones reales de manejo, en rutas de mayor de tráfico, realizando un recorrido total de 1325.84 km en 59 horas de conducción (Quinchimbla Pisuña y Solís Santamaría, 2017).

Para la representación de la dinámica del manejo, las variables que se consideran para la obtención de un ciclo de conducción, como: Velocidad media (km/h), aceleración media (m/s^2), entre otros. (Quinchimbla Pisuña y Solís Santamaría, 2017). Para la recolección de datos experimentales de estos parámetros representativos se emplean técnicas como:

Técnica On-Board

A través del uso de la instrumentación adecuada, condiciones reales de trabajo se pueden obtener datos reales de conducción con respecto al tiempo, velocidad, aceleración, entre otros. Para la aplicación de esta técnica se necesita más de un conductor para poder visualizar de mejor manera el comportamiento de conducción dentro de una determinada ruta (Quinchimbla Pisuña y Solís Santamaría, 2017). Esta técnica fue aplicada en Los Ángeles en los Estados Unidos para la obtención del FTP 75, la prueba se realizó en la zona centro de la Ciudad, consistió en recorrer el vehículo por diferentes rutas con mayor flujo vehicular (Bosch, 2005).

Técnica de la persecución del vehículo

Esta técnica consiste en adecuar un vehículo "caza" y un vehículo "objetivo" en la cual el "caza" sigue al "objetivo" dentro de una ruta determinada, en donde se trata de copiar la forma de conducción del vehículo "objetivo" constituyendo así una mejor representación de manejo. Si el "objetivo" se sale de la ruta trazada, el "caza" selecciona otro "objetivo" al azar e inicia otro seguimiento. En este procedimiento no se recomienda perseguir a vehículos públicos. Esta técnica fue empleada por primera vez en el 2001 en la ciudad de Edimburgo, Escocia (Jhonson, Formenti, Gray, y Paterson, 1975).

Pero ¿cómo acceden a los datos requeridos para determinar si un vehículo es eficiente cuando se encuentra en una ruta controlada? Para dar respuesta a esta pregunta es necesario situarnos en los años 70's y 80's donde los vehículos tuvieron una evolución significativa, dejaron de ser completamente mecánicos para ser electromecánicos cuyo sistema era controlado rudimentariamente por una computadora (Calderón, 2016).

Aquí, es donde aparece por primera vez la unidad de control del motor (ECU), la cual tuvo impulso por la necesidad de obtener mayor potencia en los motores reduciendo el consumo de combustible. Con esto evolucionó de lo mecánico a lo electrónico en cuanto a la

ejecución y regulación de los diversos parámetros como la inyección de combustible, para así poder controlar de manera más eficaz la combustión del motor (Panadero, 2012).

Hoy en día todos los sistemas del vehículo son controlados por la ECU, siendo la encargada de enviar señales a los actuadores como respuesta a los datos obtenidos por sus sensores. Para acceder a dichos valores se necesitó establecer un protocolo de comunicación que permita al usuario obtener la información que estos envían. A este nuevo sistema integrado se lo denominó como diagnóstico a bordo ó OBD-II (On Board Diagnostics) por sus siglas en inglés, cuya función principal es establecer la comunicación entre la computadora y los sensores del vehículo. Este dispositivo aparece en el año 1988 en la ciudad de California, Estados Unidos como requisito para todos los vehículos con el fin de buscar reducir los gases de efecto invernadero que son provocados por los motores de combustión (Alfaro, 2008).

Como se mencionó en los párrafos anteriores, los ciclos de conducción permiten interpretar el comportamiento del manejo dentro de una ruta determinada. Permitiendo así obtener datos relevantes de los combustibles del vehículo como su eficiencia, rendimiento y consumo a partir de los datos obtenidos durante un trayecto. Y es aquí en donde se encuentra una oportunidad para satisfacer las necesidades de caracterización de rutas con el uso de la ECU y el protocolo de comunicación OBD II como medio de adquisición de datos de sensores y actuadores directamente del vehículo para lograr establecer dicha caracterización (Pérez Llanos Pablo Santiago, 2016).

Para satisfacer esta necesidad Fabián Arévalo (2016) desarrolló una comunicación más amigable para el usuario, que permita entrelazar una interfaz gráfica mediante un programa de computadora que se encuentra conectado al OBD-II con cables empleando comunicación serial. Sin embargo, se encontró como desventaja que la instalación y

preparación de los equipos terminó requiriendo demasiado cableado para la conexión entre el OBD-II y el computador haciéndolo inestable y costoso Figura 1.

Figura 1

Conexión del dispositivo de comunicación serial ELM327



Nota: La figura representa el montaje que utilizó el autor en su conexión serial del EML327 directamente con la PC. Tomado de Fabián Eduardo Arévalo Calderón, A. G. (2016).

Desarrollo de una interfaz para la visualización y adquisición de datos provenientes de la ECU a través de OBD-II mediante un dispositivo de comunicación serial y del analizador de gases QROTECH 6000. *UNIVERSIDAD POLITÉCNICA SALESIANA SEDE MATRIZ CUENCA*, <https://dspace.ups.edu.ec/bitstream/123456789/12029/1/UPS-CT005836.pdf>.

Pero, esta metodología puede realizarse también mediante un protocolo de comunicación que no requiere cableado, es decir inalámbrico. Uno de los protocolos más utilizados para resolverlo es el denominado estándar Bluetooth IEEE 802.15.1. Que permite una conexión más estable ya que el movimiento del vehículo no afecta su lectura y transmisión de datos, al contrario, permite al conductor tener el dispositivo receptor en un lugar seguro.

Hoy en día este dispositivo ELM327 se conecta directamente al OBD-II permitiendo la transmisión de datos a una aplicación móvil. Sin embargo, estas aplicaciones son limitadas por los fabricantes, y no permiten establecer parámetros completos como es el caso de la aplicación Torque, MotorData OBD, ya que su interfaz no permite visualizar más de dos variables en la misma gráfica Figura 2.

Figura 2

Aplicación Android Torque Funcionamiento



Nota: Como se puede observar en la aplicación si bien ofrece visualizar los datos en tiempo real no es el objetivo de este proyecto. La idea es realizar un datalogger que permite graficar estos valores y obtener un manejo más completo en la realización de gráficas de los mismos.

Con la aparición de estos dispositivos bluetooth para el OBD-II también aparecen los primeros proyectos cuya ejecución consiste en emplear el ELM327 y arduino que permite visualizar y acceder a borrar códigos de fallas de manera inalámbrica. Sin embargo, no ofrecen un registro histórico completo, manipulación de gráficas y datos estadísticos

avanzados necesarios para una buena caracterización de rutas, Sin embargo, se presenta como una solución ante el manejo de datos de la ECU (Sánchez, 2015).

Bajo este contexto, el presente estudio busca solventar las falencias encontradas en trabajos previos utilizando el OBD-II como puerta de enlace entre los sensores del vehículo y un prototipo para el almacenamiento de datos “datalogger” a través del módulo ELM37 cuya función principal será monitorear en tiempo real los diferentes sensores del vehículo.

Adicionalmente, como parte del sistema se desarrollará el software de visualización y análisis de los datos adquiridos en una interfaz de computador para facilitar los análisis técnicos y/o científicos en vehículos.

Objetivo General

Desarrollar un prototipo de adquisición, almacenamiento y análisis de datos “datalogger” mediante el uso de microcontroladores, sistema OBD-II, y sensores para el análisis del comportamiento de vehículos en pruebas de ruta.

Objetivos específicos

- Diseñar los diagramas eléctricos de conexión entre los distintos módulos del prototipo mediante la plataforma libre Fritzing.
- Diseñar la interfaz de software a través de un módulo bluetooth HC-05 para la comunicación entre el puerto OBD-II y el microcontrolador.
- Diseñar una interfaz gráfica en la plataforma Python mediante el uso de la librería pandas, que permita el análisis y representación gráfica de los valores de los sensores del vehículo.
- Diseñar la estructura para albergar los componentes del prototipo mediante el software inventor e impresión 3D para que el movimiento del vehículo no provoque desconexiones y ruido en la recepción de datos.

Método

Diseño Electrónico

El dispositivo para adquisición de datos consta de tres elementos principales: Arduino Uno, Data Logger Shield V1 y ELM327 (OBD-II bluetooth). Adicionalmente se utilizará un módulo bluetooth (HC-05) para establecer la comunicación entre el Arduino Uno y el ELM327.

El Data Logger Shield V1 y el Arduino Uno se conectan a través de sus pines de conexión como se muestra en la Figura 3. Es decir, van conectados uno encima del otro permitiendo reducir el cableado y obtener la misma cantidad de puertos disponibles en el Arduino.

Figura 3

Data Logger Shield V1 y Arduino UNO



Nota: El gráfico representa el montaje del Arduino uno, y el Data logger shield V1. Tomado de *Adafruit Assembled Data Logging shield for Arduino*. Obtenido de <https://www.adafruit.com/product/1141>

Por otro lado, para conectar el HC-05 al Arduino, se utilizarán los puertos 5 y 6. Esto se debe a que estas entradas son PWM, es decir permite receptor una señal eléctrica analógica cuyo valor puede variar en el tiempo (Wordpress, 2017). Mientras tanto en el módulo estarán conectadas a las entradas Rx y Tx.

Por último, solo queda conectar el ELM327 directamente en el puerto de entrada del OBD-II del vehículo Figura 4. En este punto se puede realizar una prueba utilizando la aplicación MotorData OBD que permite monitorear ciertos sensores del vehículo a través del dispositivo, esto se realiza como método de comprobación de que el módulo se encuentra en buen estado.

Figura 4

Dispositivo ELM327



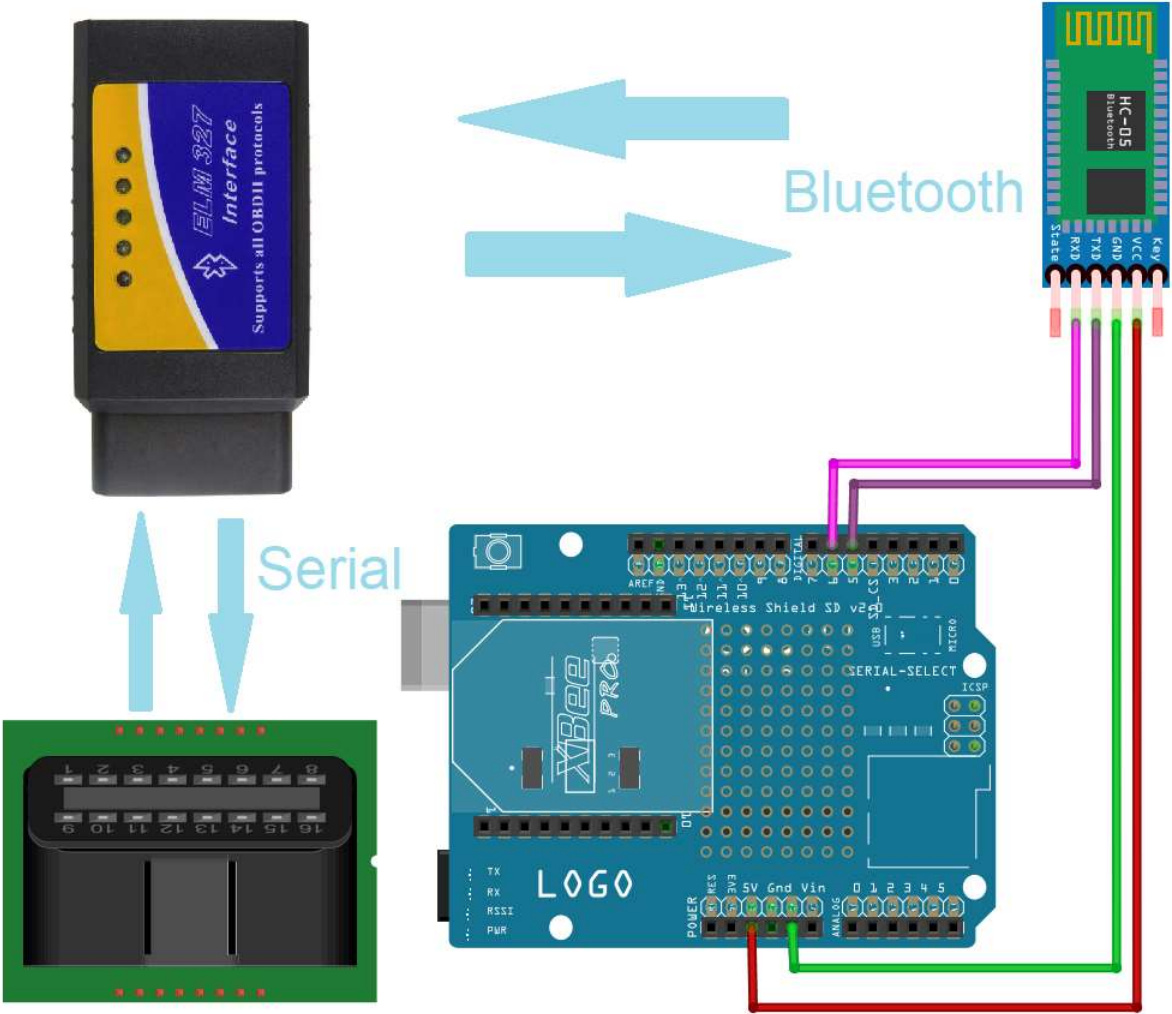
Nota: El gráfico representa el dispositivo que estará conectado al OBDII del vehículo.

Tomado de Aliexpress. (2018). *ELM327-herramienta de diagnóstico de coche, lector de código para Android/IOS, OBD2, WIFI, escáner ELM327, elm 327 V 1,5 OBD 2*. Obtenido de <https://es.aliexpress.com/item/4000387155062.html>

A continuación, en la Figura 5 se puede observar el diagrama de las conexiones en conjunto con todos los sistemas, integrando el Arduino Uno, Data Logger Shield V1, módulo bluetooth y ELM327.

Figura 5

Diagrama de conexión y comunicación



Nota: El gráfico representa cómo se comunican los dispositivos y sus conexiones. Es necesario mencionar que todo quedará almacenado dentro de la tarjeta SD del data logger shield V1.

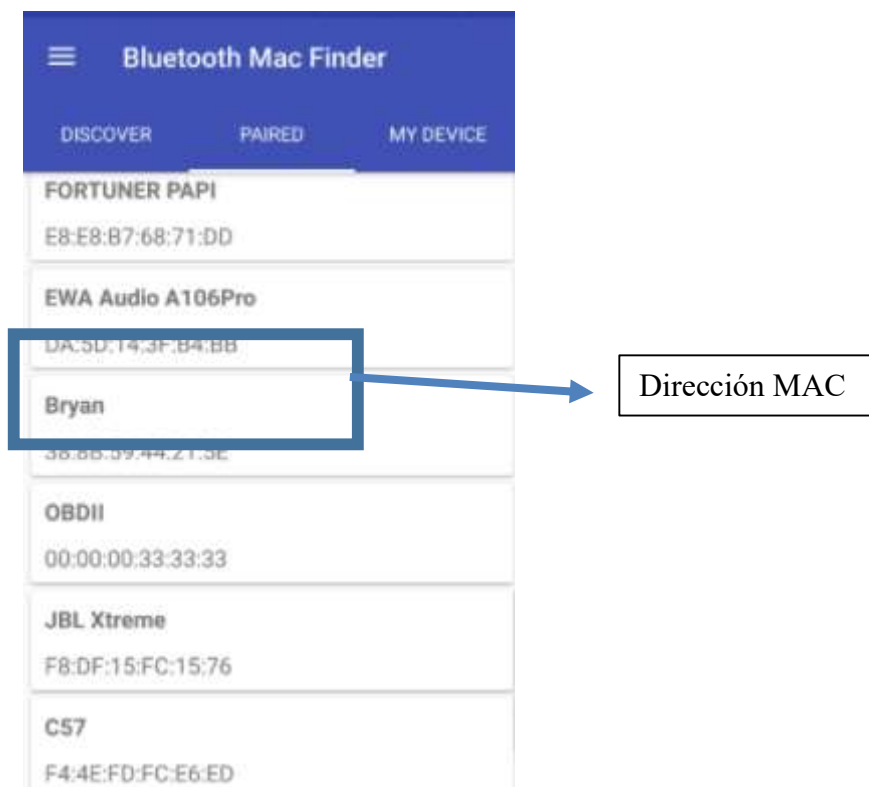
Diseño Software

Configuración inicial

Configuración HC-05. Se deberá realizar ajustes al módulo HC-05 para que se ejecute en modo maestro. Es decir, al encenderse automáticamente deberá conectarse a una MAC específica, para este caso será la que pertenece al ELM327. Para obtener la MAC del módulo bluetooth se recurrió a la aplicación “*Bluetooth Mac Finder*”, tal como se muestra en la Figura 6.

Figura 6

Aplicación Bluetooth Mac Finder



Nota: En la imagen se puede apreciar las direcciones MAC que se encuentran en cada dispositivo Bluetooth almacenado en el teléfono.

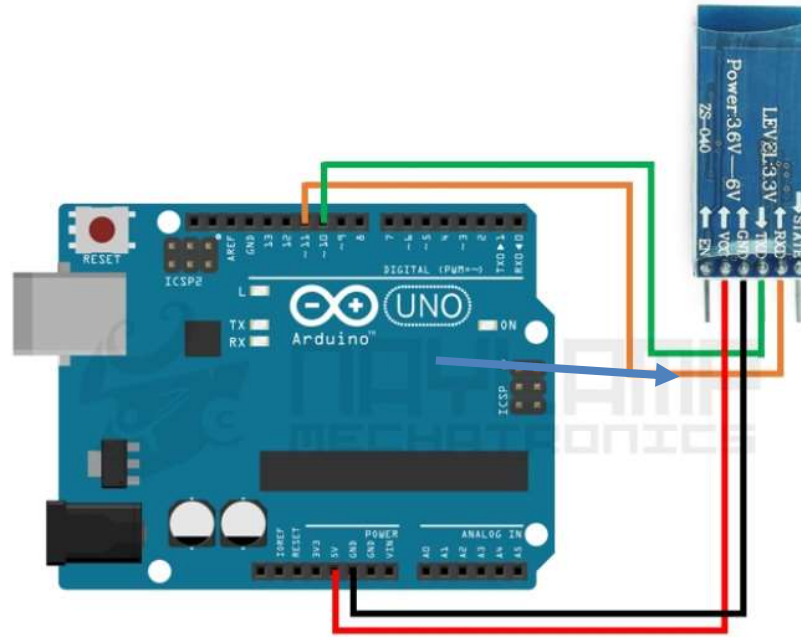
Una vez que se ha identificado la dirección MAC del dispositivo, se procede a configurar el HC-05 mediante los siguientes pasos que están especificados en el manual del usuario (Naylampmechatronics, 2016)

1. Conectar el módulo Bluetooth como se especifica en el manual del usuario

Figura 7.

Figura 7

Conexión Bluetooth y Arduino para configuración del HC-05



Nota: Se deberá tomar en cuenta que los puertos del arduino para configuración son el 11 y 10 al Rx y Tx respectivamente del HC-05. Tomado de Naylampmechatronics. (28 de Diciembre de 2016). *CONFIGURACIÓN DEL MÓDULO BLUETOOTH HC-05 USANDO COMANDOS*

A. Obtenido de Comandos AT HC-05:

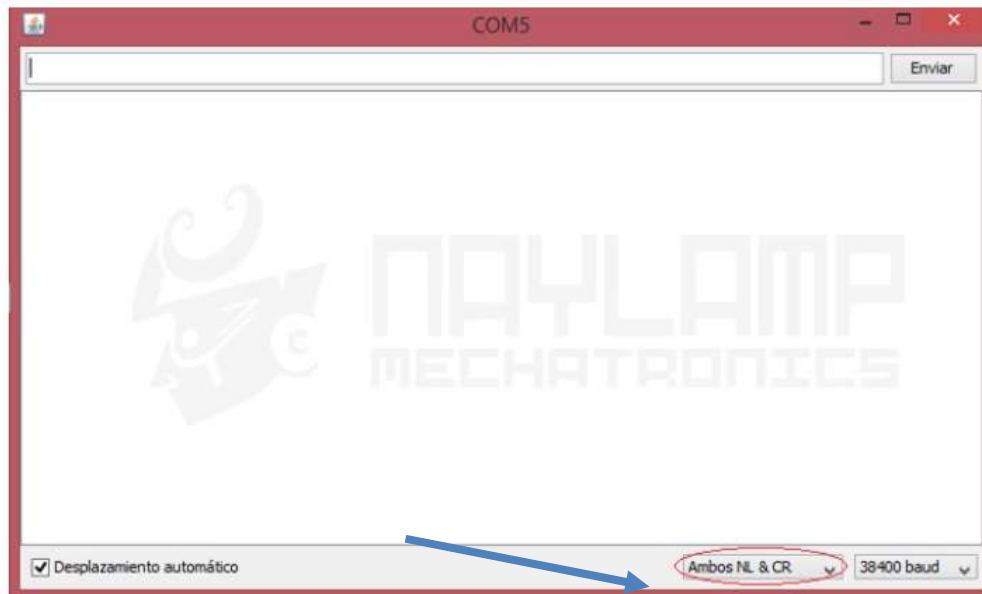
https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usando-comandos-at.html

2. Cargar el código que se encuentra en el manual de usuario Anexo 1
3. Desconectar el módulo bluetooth, y volver a conectar manteniendo presionado el botón del HC-05 hasta que comience a parpadear. Se activa el modo de configuración manual con comando AT.

4. Abrir el monitor serial y establecer la velocidad de comunicación tal como se muestra en la Figura 8

Figura 8

Monitor serial, configuración inicial



Nota: Se deberá colocar la configuración tal como se muestra en la parte inferior izquierda con el objetivo de establecer una correcta comunicación. Naylampmechatronics. (28 de Diciembre de 2016). *CONFIGURACIÓN DEL MÓDULO BLUETOOTH HC-05 USANDO COMANDOS A*. Obtenido de Comandos AT HC-05:
https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usando-comandos-at.html

5. Escribir el comando AT, deberá responder “OK” para indicar que la conexión se ha realizado con éxito, en caso de marcar “ERROR” revisar desde el paso 1.
6. Para cambiar el nombre del dispositivo se debe ingresar el comando
AT+NAME = <Nombre>. Utilizado: AT+NAME = OBDII. Respuesta “OK”
en caso de marcar error revisar desde el paso 1.

7. Para establecer la velocidad de comunicación se debe acceder mediante el siguiente comando: AT+UART = <Baud>, <StopBit>, <Parity>. Utilizado: AT+UART = 38400, 1,1. *Nota: Este paso es uno de los más importantes, ya que de no establecer este parámetro no podrá recibir ningún valor del ELM327.*
8. Ahora se configurará nuestro HC-05 como maestro, para que se conecte siempre a una MAC específica para esto se utilizará el siguiente comando. AT+ROLE = <Role> Utilizado: AT+ROLE = 1. *Nota: 0 se configura como esclavo, 1 se configura como maestro.*
9. Para que el HC-05 sepa con quien debe iniciar el emparejamiento se deberá configurar la MAC específica, para esto se debe escribir el siguiente comando: AT+CMODE = <Mode> Utilizado = AT+CMODE = 0 *Nota: 0 permite establecer una MAC específica, 1 conecta el modulo a cualquier MAC disponible.*
10. Por último, se tendrá que especificar la MAC del ELM327 que, como se explicó en párrafos anteriores fue obtenida mediante la aplicación “Bluetooth Mac Finder”. Se escribió el siguiente comando AT+BIND = <Address> Utilizado: AT+BIND=0000,00,33333. Reiniciar el HC-05 y comprobar que el emparejamiento es correcto.

Software de adquisición de datos

Comprobación de conexión y uso de librerías. El HC-05 se conectará automáticamente al EM327 gracias a la dirección MAC que se especificó anteriormente. Se puede comprobar que se encuentra emparejado ya que ambos dispositivos comenzarán a parpadear simultáneamente. Esto indicará que se encuentran transmitiendo y recibiendo datos.

Para obtener los valores de los sensores a través del OBD-II se utilizará una librería denominada “Elmduino” que logra convertir el paquete de datos de todos los sensores y muestra únicamente el valor requerido a través de una lista de comandos (Anexo 2). Para el presente proyecto se obtendrán los valores de RPM, Temperatura del radiador, Presión en la cámara de combustión y temperatura en la cámara de admisión. Aplicando el ejemplo que su librería trae por defecto y adicionando los sensores que se requieran (Anexo 3) (PowerBroker2, 2019)

Una vez cargado el código, se comprueba que se encuentre emparejado ambos dispositivos y se procederá abrir el Monitor Serial del Arduino para visualizar que se impriman los datos de las RPM del vehículo. Una vez que la prueba se ejecutó con éxito se puede agregar más comandos. Tener en cuenta que se debe establecer la misma velocidad que fue configurada en el módulo bluetooth para que no exista problemas con la lectura.

Es necesario apoyarse de librerías que se encuentran pre-instaladas en el arduino como SD.lib y a través de los siguientes pasos se podrá crear un archivo dentro de la memoria SD que contenga la información de registro de los sensores:

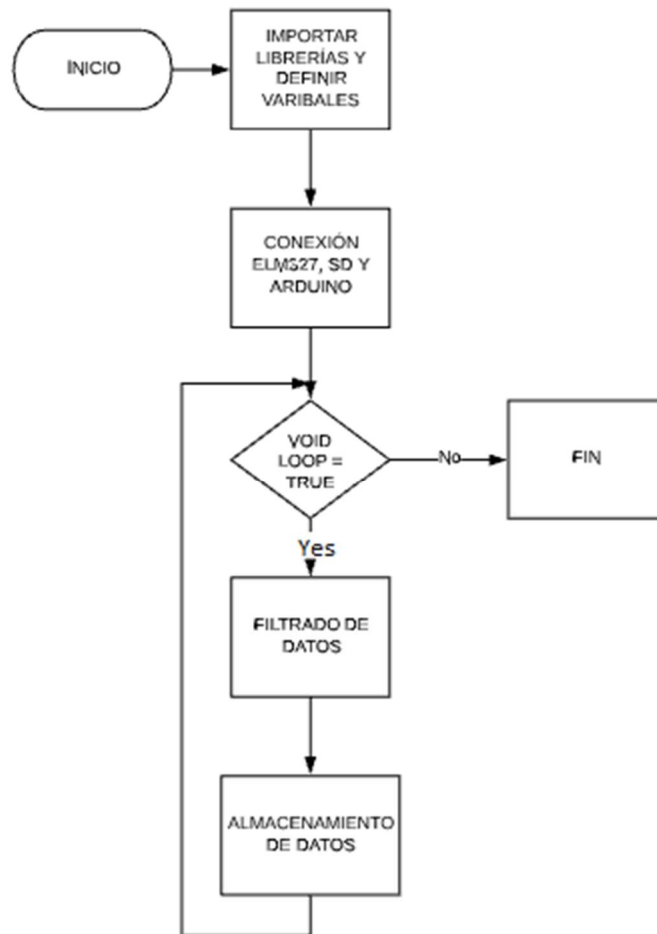
1. Verificar que el Data Logger Shield V1 se encuentre correctamente conectado como se puede evidenciar en la Figura 1.
2. Abrir el ejemplo DS1307RTC y seleccionar Set Time. Al cargar el código imprimirá la fecha y hora actual del equipo, es decir se configurará en el RTC del Data logger Shield V1. En el caso de no encontrar la librería instalada por defecto en el arduino, se deberá instalar manualmente de la siguiente manera:
 - a. Ir a la barra de menú y seleccionar “Programa”
 - b. Dar clic en “Incluir librería”
 - c. Ir a “Gestionar librería”

- d. Buscar y descargar librería DS1307RTC
3. Una vez que la hora y la fecha se encuentren correctamente configuradas en el Data logger Shield V1 se procederá a cargar el código de ejemplo que viene instalado en el arduino por defecto. Y en combinación con la librería de Elmduino se imprimirán los valores obtenidos del sensor a OBD-II través del ELM327. Con esto cada valor quedará almacenado directamente en la SD para su posterior representación gráfica en el software de Jupyter.

A continuación, se explicará el funcionamiento del software mediante el diagrama de flujo de la Figura 9. Su secuencia comienza con la importación de librerías y creación de variables que se puede evidenciar en el Anexo 5 en las líneas de código 1 - 19. Se inicializa el puerto serial y puerto ELM327 a la velocidad de 5700 baudios. Seguido de esto, el programa ingresa en un bucle infinito en donde se ejecuta la adquisición de datos, filtrado y almacenamiento. El funcionamiento interno de los bloques de conexión, adquisición, filtrado y almacenamiento serán explicados en diagramas de flujo, figuras 10 a la 12, para facilitar su comprensión.

Figura 9

Diagrama de flujo de software de adquisición de datos



Nota: El presente diagrama flujo muestra el comportamiento del código en sus distintas etapas.

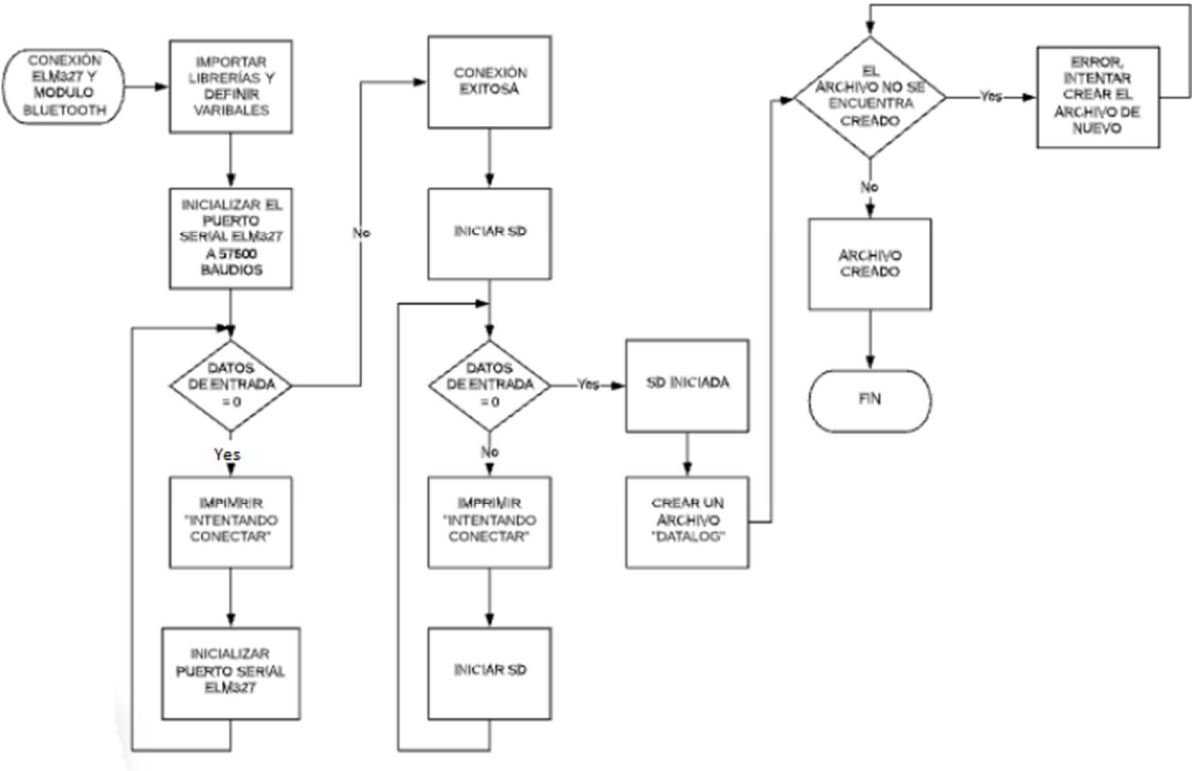
Es necesario mencionar que al apagar el sistema se finalizará el bucle de almacenamiento de datos, tras lo cual se podrá retirar manualmente la memoria SD y llevarla al computador para el análisis de los mismos mediante Pandastable como se puede observar en el anexo 4.

El primer bloque denominado “CONEXIÓN ELM237 Y ARDUINO” establece la comunicación mediante el estándar Bluetooth IEEE 802.15.1 entre el módulo ELM237 conectado a la ECU del vehículo y el prototipo desarrollado en la plataforma arduino. Se

configura un bucle hasta que se logren conseguir datos en la comunicación entre los dos dispositivos, finalizando dicho bucle cuando se establece una comunicación exitosa. El código de conexión puede observarse en el anexo 5 líneas 21 a la 33, de igual manera puede apreciar el diagrama de flujo en la Figura 10. Seguidamente el próximo bloque iniciará la SD. Se configura un bucle hasta que se logre conseguir la comunicación entre el arduino y el datalogger. Cuando el bucle finaliza se logra una comunicación exitosa. El código de conexión puede observar en el anexo 5 líneas 40 a la 58, el diagrama de flujo puede evidenciarse en la Figura 10.

Figura 10

Diagrama de flujo conexión ELM327 y Arduino

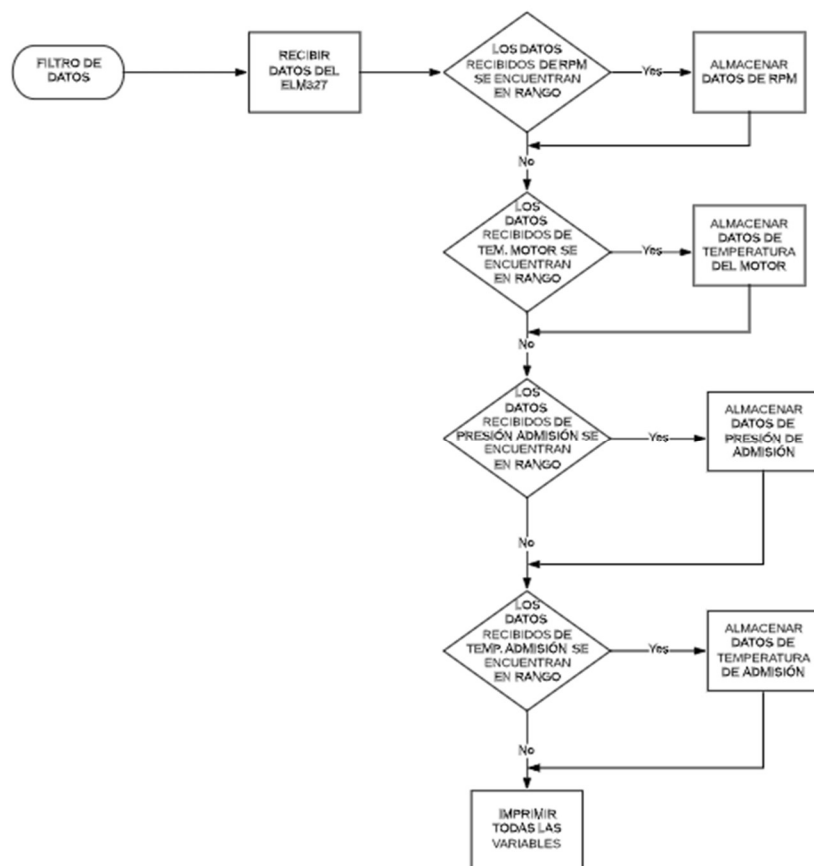


Nota: El presente diagrama flujo muestra la comunicación entre el módulo ELM327 y el módulo HC-05. Este bloque de código permite recibir los datos que son enviados desde la ECU del vehículo.

Una vez se realiza la conexión se ingresa en un bucle infinito “void loop” donde la primera acción es adquirir los datos del módulo ELM327. A continuación, se ejecuta el segundo bloque llamado “FILTRADO DE DATOS” para cada uno de los valores de RPM, Temperatura del motor, Presión de admisión y Temperatura de admisión. Para evitar valores erróneos se establecieron rangos donde: para las RPM el valor debe estar entre 0 a 60.000 y para el resto de valores debe oscilar entre 0 a 150. Si los datos recibidos se encuentran dentro de los intervalos seleccionados se almacenarán en una variable que será escrita en la SD. Si el valor se encuentra fuera de rango no se almacenará el dato erróneo, conservando el último valor válido. El código de filtrado puede observarse en el anexo 5 líneas 77-96, de igual manera puede apreciarse en el diagrama de flujo de la Figura 11.

Figura 11

Diagrama de flujo bloque de filtrado



Nota: El presente diagrama flujo muestra el bloque de filtrado. Este código permite purgar los valores que son considerados ruidos o datos erróneos recibidos del ELM327 para ser escritos en la SD.

Software de análisis de datos

Para el análisis de datos se usará un entorno de desarrollo integrado (IDE) basado en JupyterLab que permite programar a través de la Web y utiliza el lenguaje Python, este entorno presenta una bitácora de eventos donde se permite ejecutar el código bloque por bloque. Es capaz de admitir una amplia gama de datos, informática científica entre otros. Se basa principalmente en el desarrollo de Machine Learning. (Jupyter ORG, 2021)

Se escogió Jupyter notebook ya que permite ser ejecutado desde una aplicación web en cualquier navegador estándar facilitando el uso por cualquier usuario y presenta una instalación simplificada. Además, permite visualizar el código mediante bloques o líneas de texto lo que permite una mejor comprensión del funcionamiento.

Por otra parte, la librería Pandas permite un manejo especializado de los datos estadísticos en forma de hoja de cálculo. Es decir, permite trabajar los datos como DataFrame al igual que Microsoft Excel. Adicionalmente esta herramienta se combina fácilmente con Jupyter y Matplotlib.

Instalación Jupyter Notebook. A continuación, se detalla los pasos para su instalación:

1. Comprobar que Python se encuentra instalado en el sistema.
2. Abrir el CMD del sistema y escribir el siguiente comando **pip install jupyter**.
3. Esperar que se encuentre instalado y cerrar el CMD.

4. Para abrir Jupyter debemos escribir CMD en cualquier carpeta, y se escribe el comando **Jupyter notebook**. Se abrirá el explorador predeterminado con el IDE de Jupyter.

Instalación Pandas. A continuación, se detalla los pasos para su instalación:

1. Comprobar que Python se encuentra instalado en el sistema.
2. Abrir el CMD del sistema y escribir el siguiente comando **pip install pandas**.
3. Esperar que se encuentre instalado y cerrar el CMD.
4. Para comprobar que Pandas se encuentre correctamente instalado se procederá abrir Jupyter y se introducirá el siguiente comando: **import pandas as pd**. Si al ejecutar el código se produce un error se deberá volver a realizar los pasos de instalación.

Instalación Pandastable. A continuación, se detalla los pasos para su instalación:

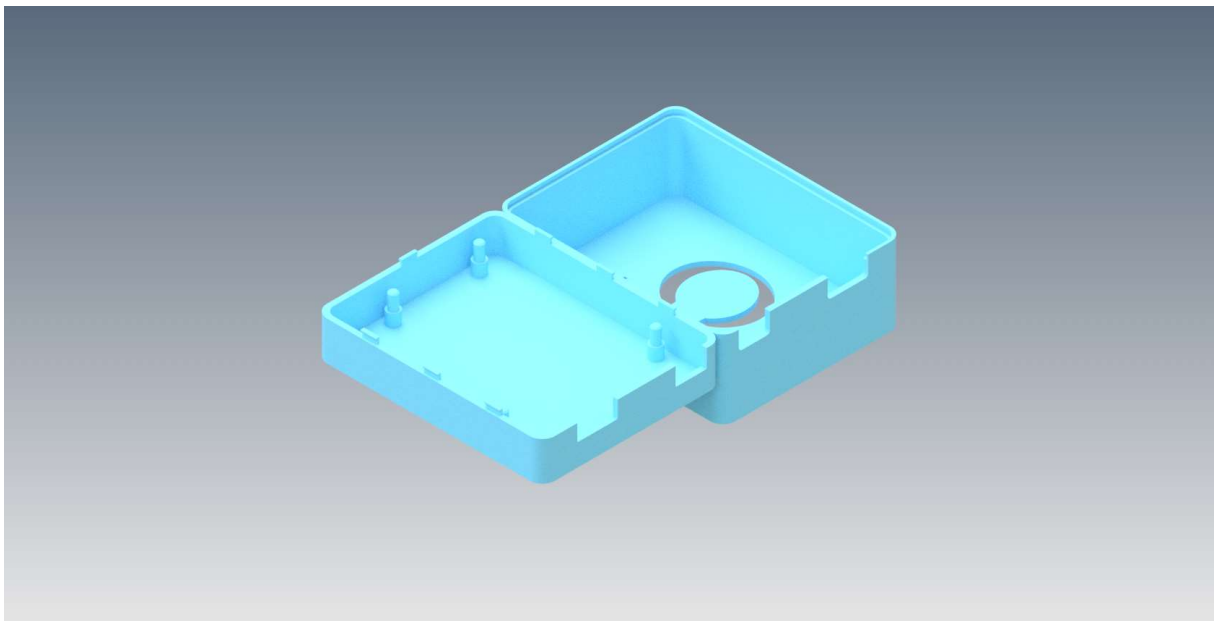
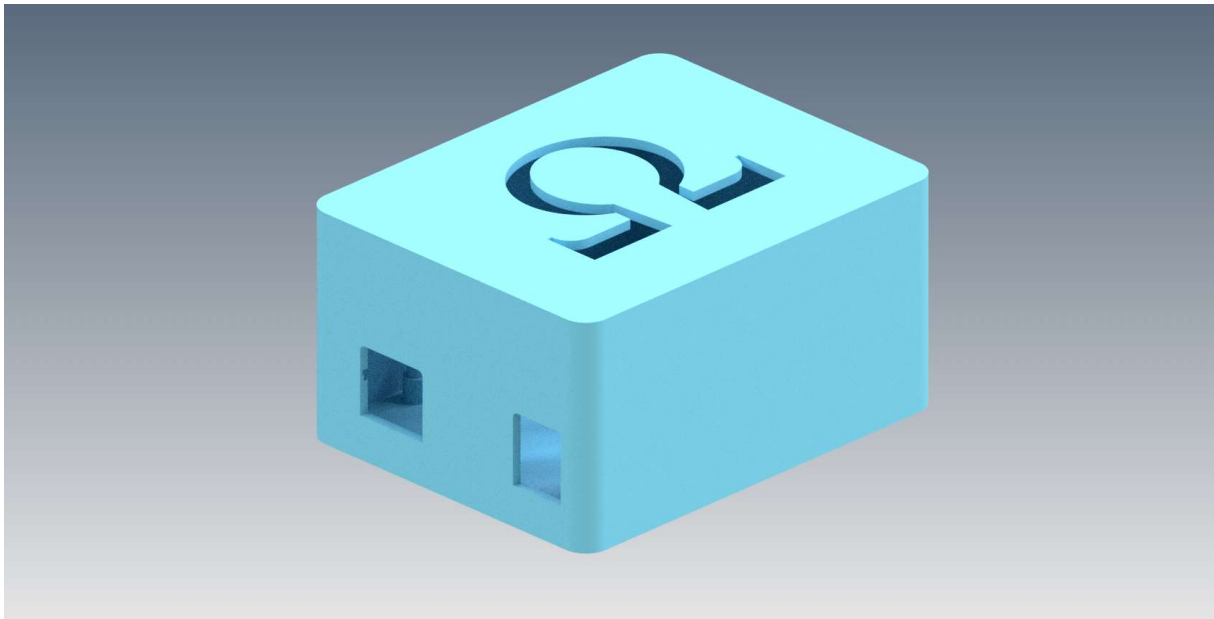
1. Comprobar que Python se encuentra instalado en el sistema.
2. Abrir el CMD del sistema y escribir el siguiente comando **pip install pandastable**.
3. Esperar que se encuentre instalado y cerrar el CMD.
4. Para comprobar que Pandas Table se encuentre correctamente instalado se procederá abrir Jupyter y se introducirá el siguiente comando: **import pandastable as pd**. Si al ejecutar el código se produce un error se deberá volver a realizar los pasos de instalación.

Diseño Mecánico

Para la protección del sistema electrónico se diseñó un soporte que permita mantener los componentes conectados de forma segura pero que pueden ser desconectados fácilmente para su manipulación dentro del vehículo. Para ello se recurrió al software de diseño mecánico Inventor como se puede observar el prototipo virtual Figura 12 y posteriormente realizar su fabricación mediante tecnología de impresión en 3D Figura 13.

Figura 12

Prototipo virtual Inventor



Nota: Las entradas USB y de poder se encuentran ubicadas en la parte frontal, internamente contiene pines en donde encaja el arduino y se asegura con unas vinchas en la parte superior.

Figura 13

Construcción en impresora 3D y ensamble.



Nota: El proceso de impresión tardó aproximadamente 4 horas en completar ambas caras del protector.

Costos de producción

En la siguiente Tabla 1 se encuentra detallado los costos de cada componente que interviene en el ensamble y fabricación del datalogger.

Tabla 1

Costos de producción para el datalogger

Cantidad	Descripción	Precio \$
1	Arduino Uno	19,00
1	Datalogger Shield V1	22,00
1	Módulo ELM327	35,00

1	Módulo Bluetooth HC-05	13,00
1	Soporte impresión 3D	3,00
1	Tarjeta SD	5,00
4	Jumpers	2,50
TOTAL		99,50

Nota: En esta tabla se puede evidenciar la cantidad de elementos necesarios, modelo y costos de cada ítem.

Experimentos

Para validar que el prototipo se encuentra funcionando correctamente, se realizarán tres experimentos bajo condiciones distintas. El número de pruebas dependerá de los recursos de metrología disponibles para este proyecto en donde se comparan los resultados obtenidos por el sistema con los resultados medidos con instrumentos patrón como se detalla a continuación:

Experimento 1:

El primer experimento se realiza en condiciones estáticas, es decir con el vehículo detenido. Esta prueba tiene como objetivo determinar que la conexión entre el módulo ELM327 y el arduino se encuentren conectados correctamente. Además, se comprobará si la SD se ha iniciado correctamente, para lo cual:

- Se conectará el prototipo a un computador
- Abrir el monitor serial y verificar los mensajes de conexión exitosa emitidos por el sistema.

Experimento 2:

El segundo experimento tiene como objetivo comparar los resultados obtenidos del OBD-II a través del ELM327 para esto se realizará una prueba física utilizando una pistola de temperatura marca PeakMeter, Figura 14, para comprobar que la temperatura del motor es igual que la temperatura marcada por el sensor. Para medir las RPM, se utilizará el tablero del vehículo. Por último, para verificar la presión y temperatura de admisión no se cuenta con la instrumentación adecuada por lo cual se comparan dichos resultados empleando la aplicación MotorDataOBD.

Figura 14

Pistola de temperatura marca PEAKMETER

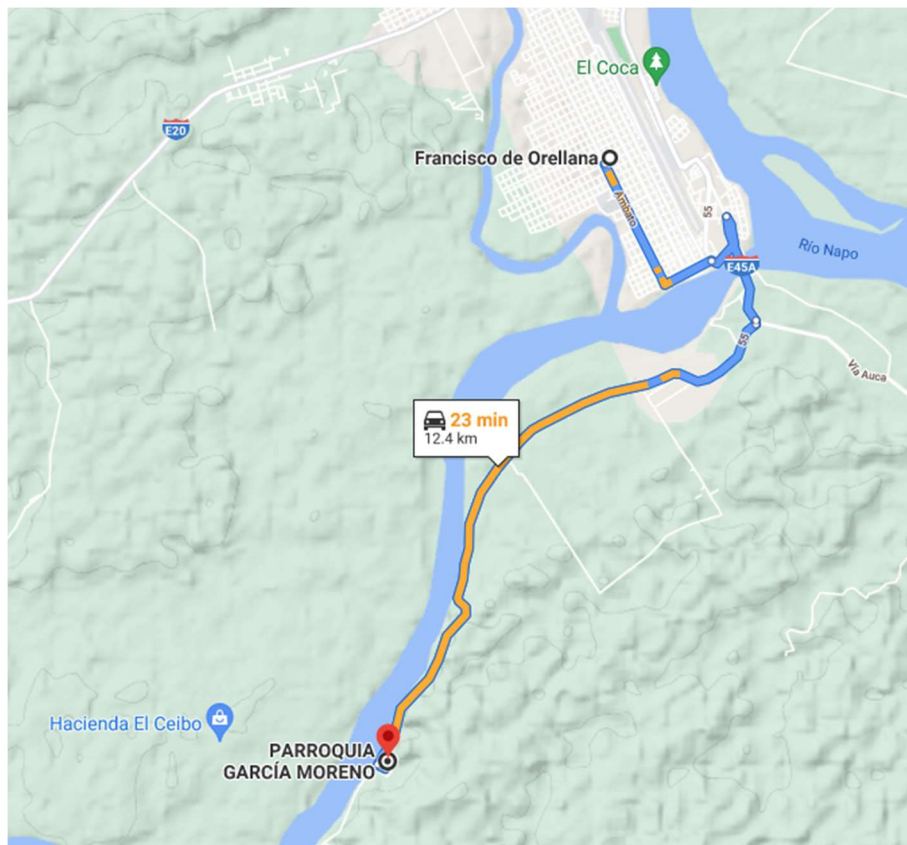


Experimento 3:

Para este experimento se realizará una prueba de ruta en condiciones reales de manejo. La ruta seleccionada es un trayecto asfaltado de una carretera con poca afluencia de tráfico. En la siguiente Figura 15 se puede observar el recorrido marcado en Google Maps. Esta prueba tendrá una duración de 35 minutos con una temperatura ambiental de 25 grados aproximadamente en la ciudad de Coca-Orellana. El vehículo seleccionado para realizar esta prueba de ruta es un Peugeot 206 1.6.

Figura 15

Ruta para experimento 3



Nota: La parroquia García Moreno se encuentra ubicada a 12.4 Km de la ciudad de Coca.

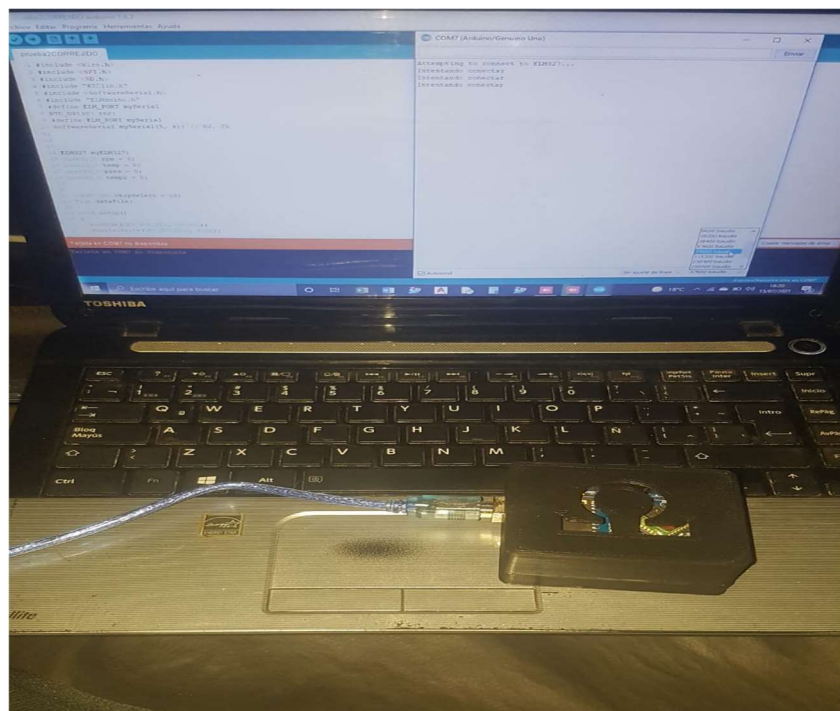
Fuente: Google Maps.

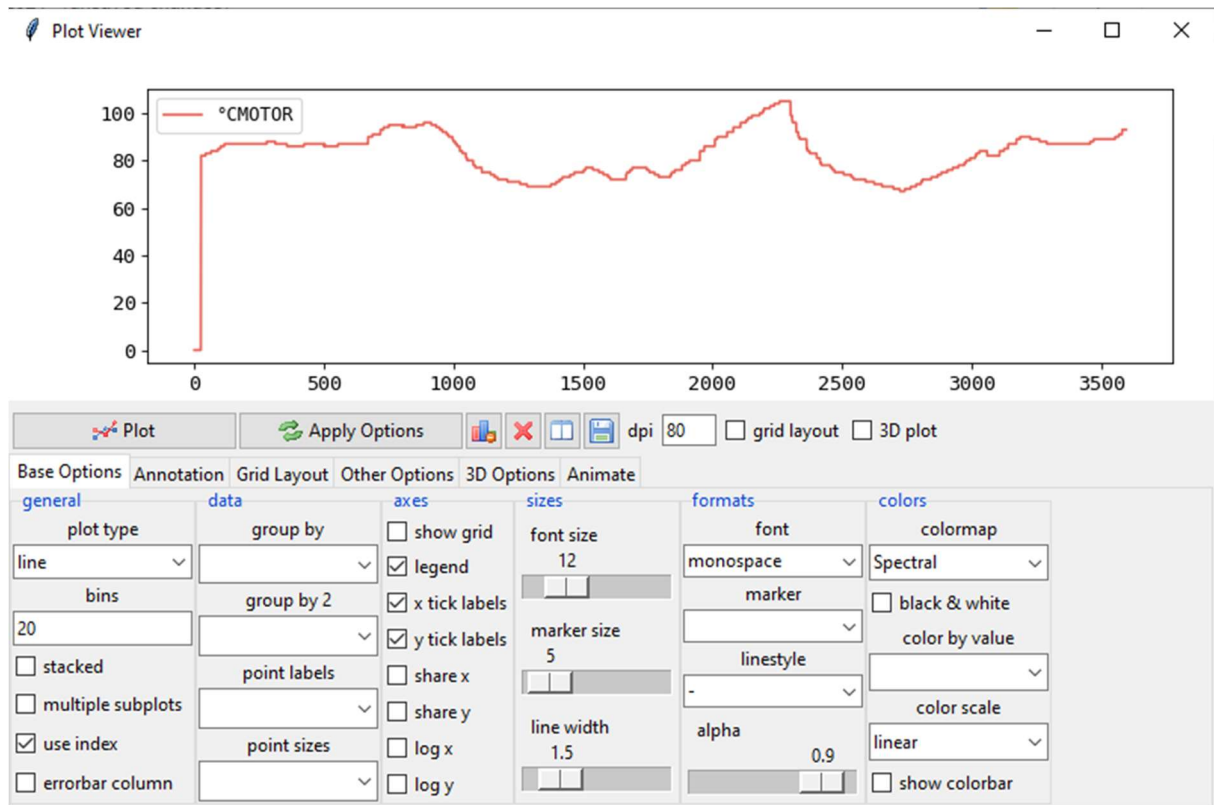
Resultados

A continuación, en la Figura 16, se muestra el resultado final del prototipo conectado al ELM327 al OBD-II del vehículo, el prototipo en su protector y un ejemplo de las gráficas obtenidas en el software de análisis.

Figura 16

Resultado de la construcción mecánica y eléctrica



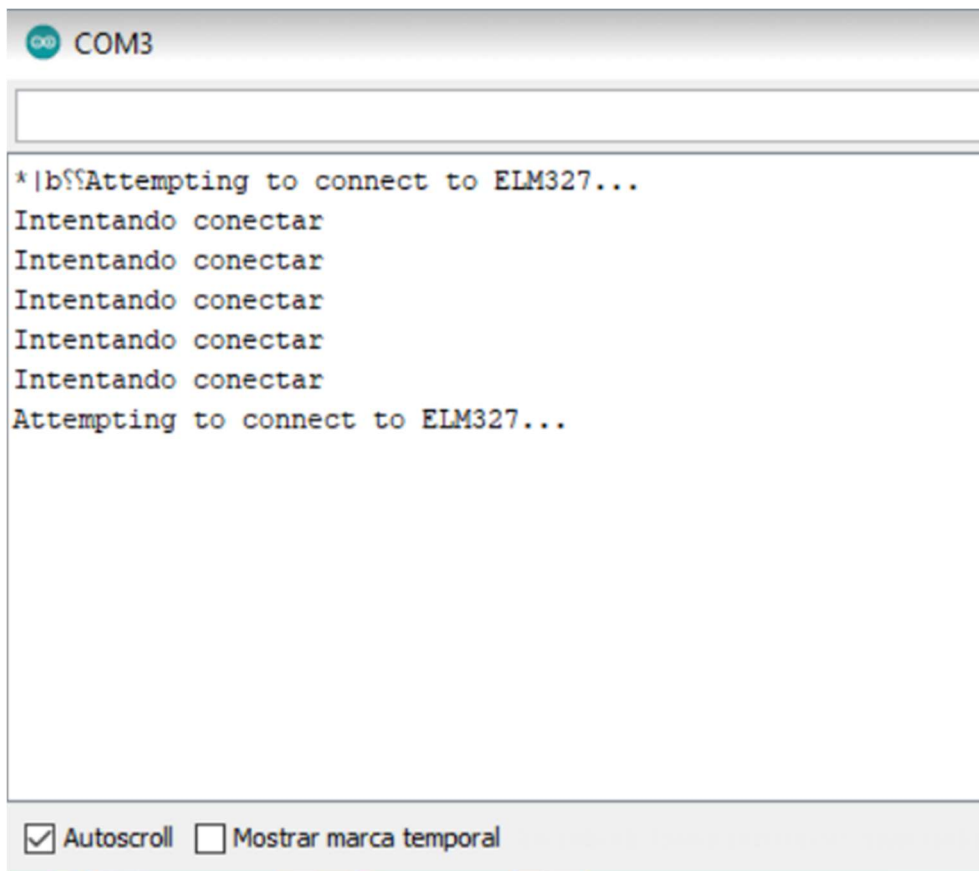


Experimento 1

El experimento fue realizado con éxito al establecer la comunicación entre el módulo ELM327 y el prototipo. Tal como se puede evidenciar en la Figura 16. Es necesario mencionar que mientras no se establezca la conexión el prototipo intentará establecer la comunicación hasta que pueda emparejarse.

Figura 16

Enlace de comunicación entre ELM327 y el prototipo



The image shows a serial terminal window titled 'COM3'. The output text is as follows:

```
*|b??Attempting to connect to ELM327...
Intentando conectar
Intentando conectar
Intentando conectar
Intentando conectar
Intentando conectar
Attempting to connect to ELM327...
```

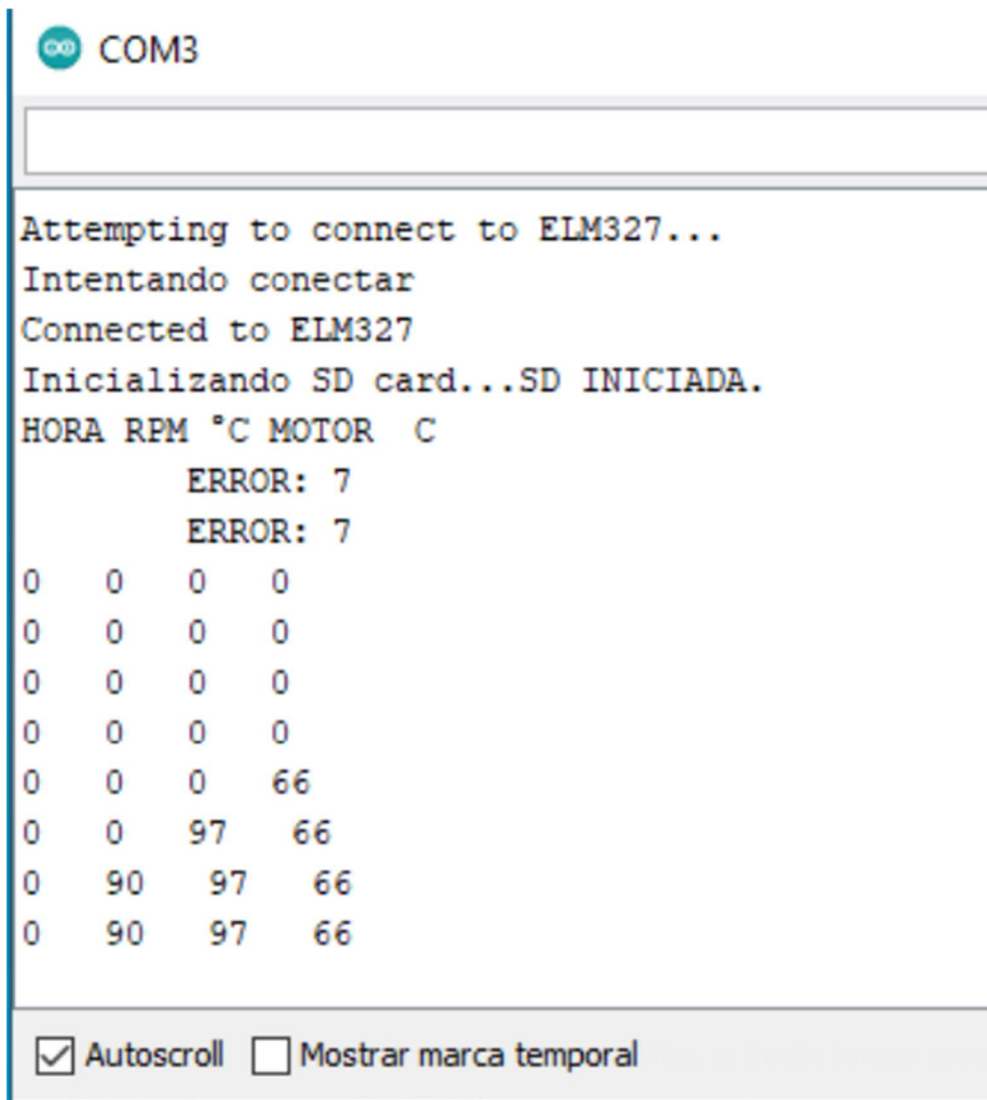
At the bottom of the window, there are two checkboxes: 'Autoscroll' (checked) and 'Mostrar marca temporal' (unchecked).

Nota: El código está diseñado para que espere a una conexión exitosa entre el módulo Bluetooth y el ELM327. Como se puede observar una vez que está conectado se inicia la comunicación.

Si la conexión fue exitosa entre el prototipo y el ELM327 seguido comprueba si puede iniciar el Módulo datalogger Shield V1. Al igual que conexión previa intentará establecer una conexión en un bucle infinito. En caso de no conectarse seguirá intentando. Si la conexión está establecida continuará con las siguientes líneas de código como se puede evidenciar en la Figura 17.

Figura 17

Inicializando la SD



```
COM3
Attempting to connect to ELM327...
Intentando conectar
Connected to ELM327
Inicializando SD card...SD INICIADA.
HORA RPM °C MOTOR C
      ERROR: 7
      ERROR: 7
0    0    0    0
0    0    0    0
0    0    0    0
0    0    0    0
0    0    0    66
0    0    97   66
0   90    97   66
0   90    97   66
 Autoscroll  Mostrar marca temporal
```

Nota: Una vez que la conexión entre el módulo Bluetooth y ELM327 se haya realizado con éxito se inicializa la SD para comenzar a almacenar los datos. En este caso los errores iniciales no serán almacenados en la SD.

Como se mencionó anteriormente esta prueba fue realizada en condiciones estáticas, es decir con el vehículo detenido y tiene como intención comprobar que el software no presente problemas en la realizar la comunicación entre el ELM327, Arduino y Datalogger SD. Tal como se puede observar en la Figura 16 y 17. Las conexiones fueron realizadas con éxito.

Experimento 2

Figura 18

Temperatura del motor

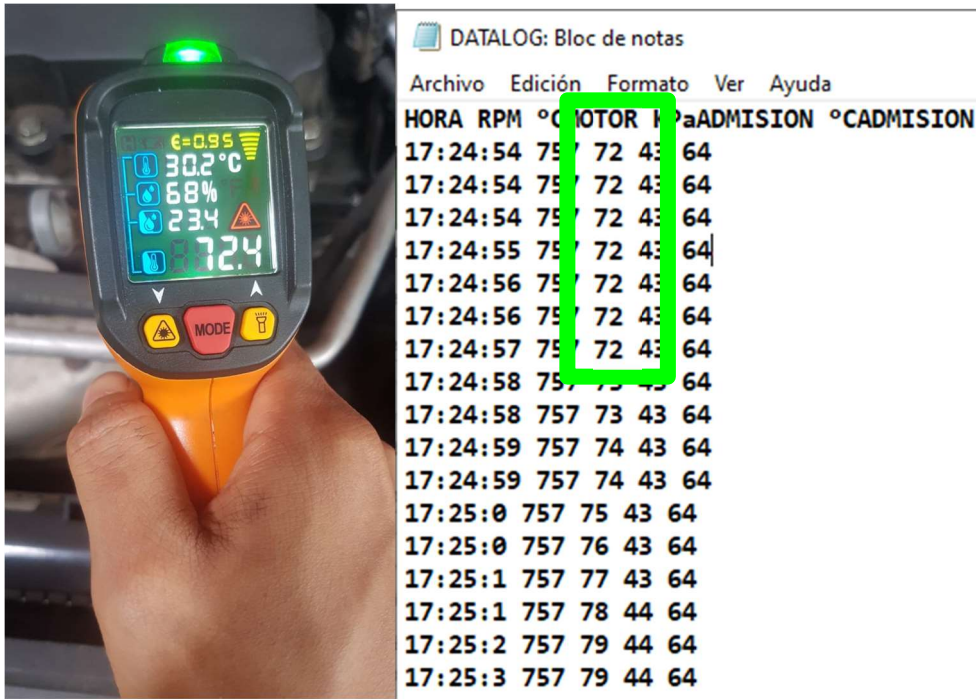


Figura 19

RPM del motor

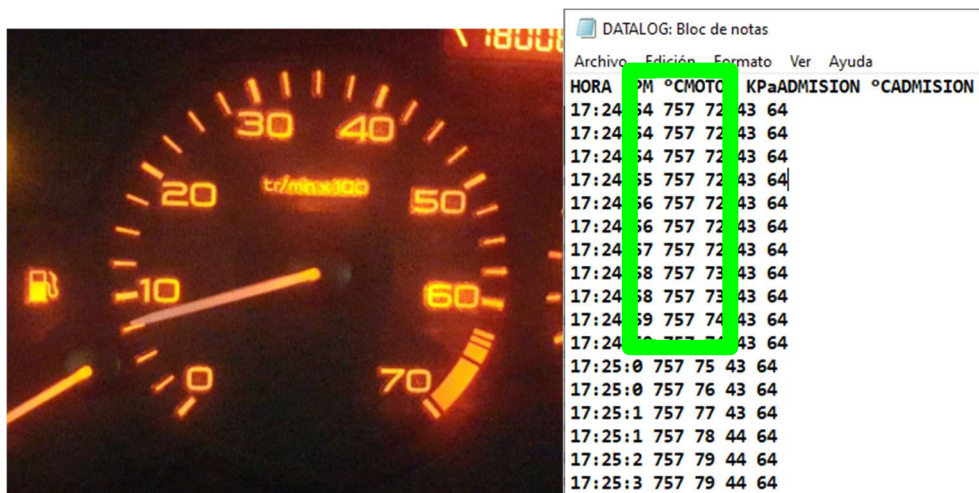


Figura 20

Presión y temperatura de admisión

PID	Parámetro	Valor
0C	Revoluciones del cigüeñal	760
0F	Temperatura del aire de admisión	64
0B	Presión absoluta en el múltiple de admisión	43
04	Carga del motor - valor calculado	4.31
0E	Angulo de avance del encendido (cilindro 1)	6.5
11	Válvula de mariposa - posición absoluta	12.55
0D	Velocidad del vehículo	0
14	Sensor de oxígeno (B1S1) - voltaje	0.52

HORA	RPM	°CMOTOR	OR	KPaADMISION	°CADMISION
17:24:54	757	72	43	64	
17:24:54	757	72	43	64	
17:24:54	757	72	43	64	
17:24:55	757	72	43	64	
17:24:56	757	72	43	64	
17:24:56	757	72	43	64	
17:24:57	757	72	43	64	
17:24:58	757	73	43	64	
17:24:58	757	73	43	64	
17:24:59	757	74	43	64	
17:24:59	757	74	43	64	
17:25:0	757	75	43	64	
17:25:0	757	76	43	64	
17:25:1	757	77	43	64	
17:25:1	757	78	44	64	
17:25:2	757	79	44	64	
17:25:3	757	79	44	64	

Como se puede evidenciar en las Figuras del experimento 2, se ha corroborado que los valores obtenidos a través del ELM327 almacenados en la SD mantienen un error relativo del 0,55% en un promedio de 5 muestras.

Experimento 3

Para este último experimento se realizará una prueba de ruta con los valores obtenidos de los sensores. Se almacenan los valores obtenidos en la memoria SD para luego obtener diferentes gráficas como se puede evidenciar en la Figura 23. Dando como resultado gráficas que pueden ser utilizadas por los científicos para estudios sobre el rendimiento del vehículo o la caracterización de la ruta.

Figura 21

Conexión del Arduino en ruta



Nota: Se utilizó una laptop para poder visualizar el comportamiento de la adquisición de datos a lo largo de la ruta.

Exportación de datos SD

Se retiró la memoria SD del arduino y se copió el archivo DATALOG dentro de la carpeta que contiene el código en Jupyter Notebook para ser exportado al software de visualización.

Figura 22

Bloc de notas con los datos almacenados en la SD

DATALOG: Bloc de notas

Archivo Edición Formato Ver Ayuda

HORA	RPM	°CMOTOR	KPaADMISION	°CADMISION
17:24:54	0	0	0	0
17:24:54	0	0	0	0
17:24:54	0	0	97	0
17:24:55	0	0	97	0
17:24:56	0	0	87	0
17:24:56	0	0	87	66
17:24:57	0	0	79	66
17:24:58	0	0	79	66
17:24:58	0	0	34	66
17:24:59	0	0	34	64
17:24:59	0	0	33	64
17:25:0	0	0	34	64
17:25:0	757	0	36	64
17:25:1	757	0	43	64
17:25:1	757	0	44	64
17:25:2	757	0	44	64
17:25:3	757	0	44	64
17:25:3	757	0	44	64
17:25:4	757	0	44	64
17:25:5	757	0	44	63
17:25:5	757	0	55	63

Nota: Los datos son extraídos de la SD a un computador y pueden ser visualizados en el archivo de bloc de notas previo a su análisis.

Figura 23

Importación de datos al software PandasTable

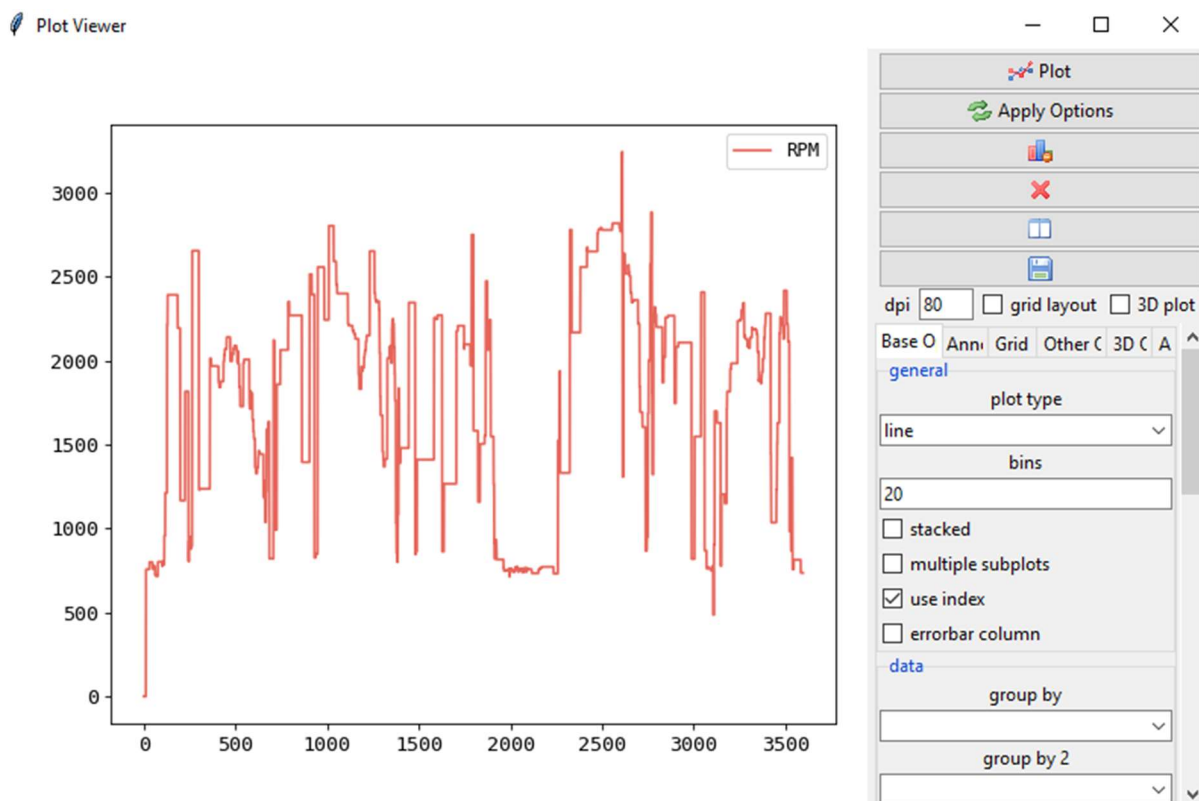
	HORA	RPM	°CMOTOR	KPaADMISI	°CADMISION
1	17:24:54	0	0	0	0
2	17:24:54	0	0	0	0
3	17:24:54	0	0	97	0
4	17:24:55	0	0	97	0
5	17:24:56	0	0	87	0
6	17:24:56	0	0	87	66
7	17:24:57	0	0	79	66
8	17:24:58	0	0	79	66
9	17:24:58	0	0	34	66
10	17:24:59	0	0	34	64
11	17:24:59	0	0	33	64
12	17:25:0	0	0	34	64
13	17:25:0	757	0	36	64
14	17:25:1	757	0	43	64
15	17:25:1	757	0	44	64
16	17:25:2	757	0	44	64

3598 rows x 5 columns

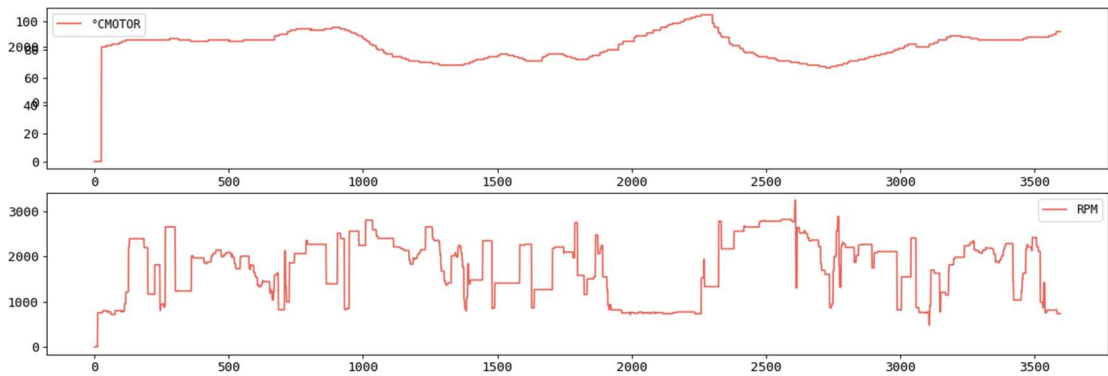
Nota: El prototipo se encarga de almacenar los datos con el nombre que se haya establecido en el código principal. Una vez que este documento sea guardado en la carpeta donde se encuentra el código de Jupyter Notebook. Se procede a correr el programa. Los datos se importarán del documento guardado con el nombre correspondiente.

Figura 24

Gráfica de datos en el software PandasTable

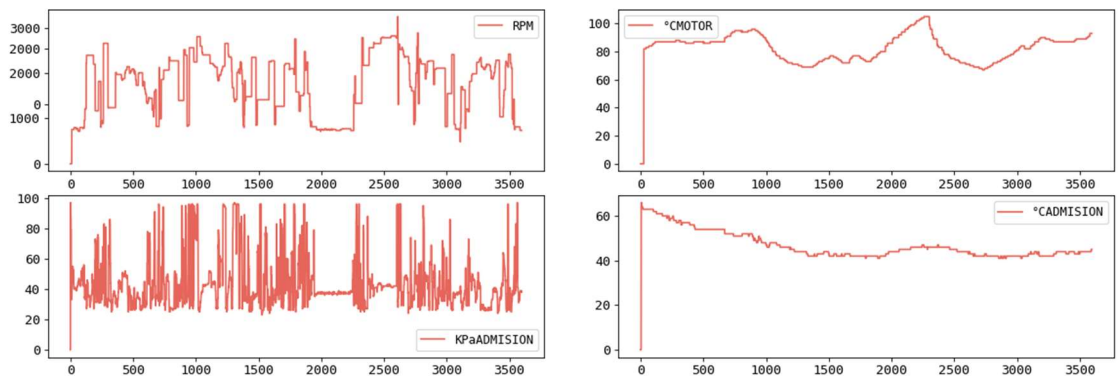


Plot Viewer

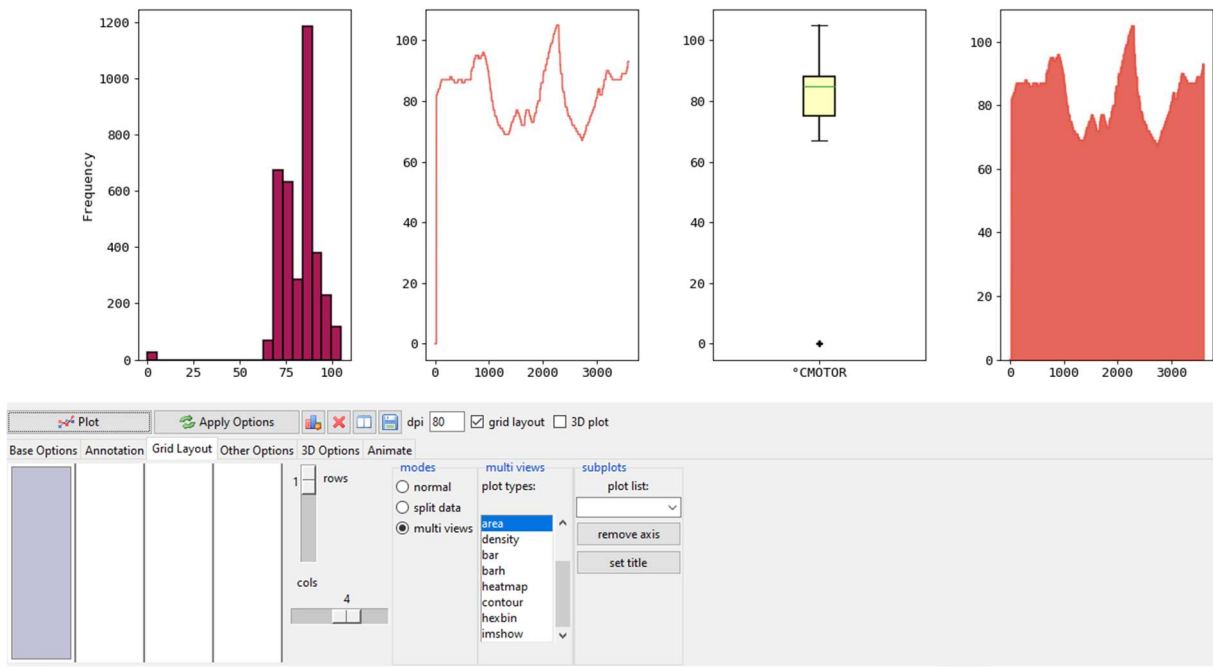


Plot Viewer interface showing options: Plot, Apply Options, dpi 80, grid layout, 3D plot. The interface includes tabs for Base Options, Annotation, Grid Layout, Other Options, 3D Options, and Animate. A 'multi views' dropdown menu is open, listing plot types: histogram, line, scatter, boxplot, dotplot, area, density, and bar. The 'line' option is selected. A 'subplots' section contains a 'plot list' dropdown, 'remove axis', and 'set title' buttons. A grid layout is visible with 4 rows and 3 columns.

Plot Viewer



Plot Viewer interface showing options: Plot, Apply Options, dpi 80, grid layout, 3D plot. The interface includes tabs for Base Options, Annotation, Grid Layout, Other Options, 3D Options, and Animate. A 'multi views' dropdown menu is open, listing plot types: histogram, line, scatter, boxplot, dotplot, area, density, and bar. The 'line' option is selected. A 'subplots' section contains a 'plot list' dropdown, 'remove axis', and 'set title' buttons. A grid layout is visible with 4 rows and 4 columns.



Nota: Como se puede observar este software permite graficar los datos que se hayan obtenido en función de lo que se desee interpretar. Se puede graficar los datos individualmente, en conjunto o todos a la vez. Adicionalmente se tiene la opción de aplicar distintos gráficos que permitan una mejor visualización de los mismos.

Discusión

Una vez que los datos son extraídos de la SD se procedió a graficar las RPM, temperatura y presión de admisión. Tal como se puede observar en la Figura 21 dando como resultado la obtención de gráficas en función de las necesidades para el desarrollo de pruebas de ruta. El prototipo logró una adquisición, almacenamiento y análisis completo de los datos de ruta sin la necesidad de transcribir los valores a una hoja de cálculo y un procesamiento manual de los valores. Como se mencionó en el apartado de introducción Fabián Arévalo (2016) realizó la adquisición de datos empleando una metodología diferente. Consistía en extraer los valores del OBD-II a través de un adaptador que se encontraba conectado al computador. Como la adquisición de datos se realiza en pruebas de ruta, tener cables conectados puede provocar ruido eléctrico en el caso de que llegaran a moverse o desconectarse. Siendo esto una desventaja frente a nuestro prototipo, ya que este permite obtener los valores registrados en una SD y su adquisición de datos lo realiza mediante el protocolo de comunicación Bluetooth IEEE 802.15.1.

Un punto débil que se puede evidenciar en el prototipo es que el manejo del software requiere un poco de práctica para lograr obtener el máximo provecho de cada una de las herramientas y aplicaciones que permite interactuar con la base de datos. Sin embargo, una vez que se ha familiarizado con la interfaz el poder de análisis mejorará de manera significativa. Por último, es necesario mencionar que tanto los sensores del vehículo como los valores físicos obtenidos han dado como resultado un error relativo del 0.55 % en un promedio de 5 muestras, siendo estos resultados confiables para el desarrollo de prueba de rutas.

Conclusiones

Se logró realizar los diagramas de conexión eléctrica de manera satisfactoria mediante el uso de la plataforma Fritzing tal como se muestra en la Figura 5 donde se puede apreciar de mejor manera la conexión del cableado.

Se concluyó que el protocolo de comunicación Bluetooth IEEE 802.15.1. permitió establecer la comunicación entre el OBD-II y el microcontrolador de manera eficaz tal como se puede apreciar en la Figura 16.

El diseño mecánico a través de impresión 3D permite realizar mejoras al prototipo para que pueda ser utilizado de manera segura. Se concluye que al emplear este recurso tecnológico permitió asegurar la integridad del circuito evitando desconexiones o ruido en la recepción de datos.

El desarrollo del software de adquisición de datos permitió obtener los valores de los sensores del vehículo en tiempo real, y estos ser almacenados dentro de una memoria SD para su posterior análisis mediante el software de graficación. Se concluye que esta herramienta puede ser utilizada por los científicos para el desarrollo de pruebas de rutas ya que permite interactuar entre distintos modelos de gráficas y variables. Teniendo en cuenta la posibilidad de escoger a conveniencia el tipo de gráfico que se requiere visualizar.

Se puede concluir que el proyecto puede ser aplicado a modelos de vehículo que utilicen el puerto OBD-II correspondiente al ELM327 ya que al ser códigos PDI estos se encuentran ya desarrollados en la librería del propio módulo. Sin embargo, es necesario aclarar que debe establecer una comunicación entre este dispositivo, el módulo bluetooth y el arduino a la misma velocidad con el fin de obtener una lectura correcta de los datos del sensor.

Recomendaciones

Se recomienda analizar los códigos PDI de la librería ELMduino y compararlos con los del OBD-II del vehículo, ya que el consumo de combustible arroja un valor de 0. Una de las posibilidades es que el comando no esté definido correctamente o este pertenezca a otro sensor. Este estudio podrá determinar el error que existe en la librería. Es necesario mencionar que la versión utilizada es la 2.6 por lo que en un futuro pueda ser lanzado un parche que solucione este problema.

Para futuras investigaciones se recomienda exportar los datos directamente a la nube para que sean almacenados en una base de datos en línea. Esto mejorará en gran medida el proyecto ya que no se requerirá de un lector SD para que pueda ser almacenado en la PC. Y además tendrá un mejor alcance al adaptarse a la industria 4.0 que tiene como principal objetivo manejar todo desde la nube. Adicionalmente se puede reemplazar el arduino por un MKR GSM 1400 que permite conectarse a la red del celular para tener conectividad al internet y transferir los datos.

Referencias

- Acosta, O., & González, J. (2007). Globalización: una aproximación desde la evolución biológica y los sistemas complejos auto-organizativos. *Revista Unal*, 101-121.
- Adafruit. (19 de Abril de 2018). *Adafruit Assembled Data Logging shield for Arduino*. Obtenido de <https://www.adafruit.com/product/1141>
- Aliexpress. (2018). *ELM327-herramienta de diagnóstico de coche, lector de código para Android/IOS, OBD2, WIFI, escáner ELM327, elm 327 V 1,5 OBD 2*. Obtenido de https://es.aliexpress.com/item/4000387155062.html?aff_fcid=114490291e08437aa84528b0d0361218-1625250366371-02093-_pxfxXeC&aff_fsk=_pxfxXeC&af=2347803&aff_platform=api-new-link-generate&sk=_pxfxXeC&aff_trace_key=114490291e08437aa84528b0d0361218-162525036637
- Alfaro, D. C. (s.f.). Diagnóstico a bordo. *Comisión Nacional para el uso eficiente de energía conuee*, 1-4.
- Borja Pintos, G. d. (2011). *Desarrollo de una metodología para generación de ciclos de conducción representativos del tráfico real urbano. Aplicación para medida de emisiones en banco de rodillos*. Madrid
- Bosch, R. (2005). *Manual de la Técnica del Automóvil*. Plochingen.
- Calderón, F. E. (2016). *Desarrollo de una interfaz para la visualización y adquisición de datos provenientes de la ECU a través de OBD-II mediante un dispositivo de comunicación serial y del analizador de gases QROTECH 6000*. Cuenca: Artículo. Obtenido de <https://dspace.ups.edu.ec/bitstream/123456789/12029/1/UPS-CT005836.pdf>

- Carrizo, J. S. (2017). *Simulador de una ECU y diagnóstico mediante CAN y OBD-II*.
<https://ruidera.uclm.es/xmlui/bitstream/handle/10578/15618/TFG%20Jorge%20Sanchez%20Carrizo.pdf?sequence=1&isAllowed=y>: Artículo.
- DiselNet. (2011). *Disel Net*. Obtenido de: <https://www.Dieselnet.com/standards/cycles/ftp75.php>
- Domínguez, M. (2021). La contaminación ambiental, un tema con compromiso social. *Producción, Limpia*, 9-21.
- El comercio. (2019). Parque automotor de Ecuador creció en 1,4 millones de vehículos en una década. *El comercio*, 1.
- Fabián Eduardo Arévalo Calderón, A. G. (2016). Desarrollo de una interfaz para la visualización y adquisición de datos provenientes de la ECU a través de OBD-II mediante un dispositivo de comunicación serial y del analizador de gases QROTECH 6000. *UNIVERSIDAD POLITÉCNICA SALESIANA SEDE MATRIZ CUENCA* ,
<https://dspace.ups.edu.ec/bitstream/123456789/12029/1/UPS-CT005836.pdf>.
- Gómez, J. (21 de Febrero de 2020). *Diario motor*. Obtenido de <https://www.diarimotor.com/reportajes/diesel-gasolina-que-contamina-mas/>
- Hurtado Gómez, A. (2014). *Desarrollo de ciclos de conducción para el área metropolitana centro occidente-Amco*. Obtenido de <https://dspace.ups.edu.ec/bitstream/123456789/15032/1/UPS-CT007421.pdf>
- Jarrín, B. E. (2019). *Evaluación del Consumo de Combustible en Vehículos a 2385 msnm en los Modos de*. QUITO: UISEK.
- Jhonson, T., Formenti, D., Gary, R., & Paterson, W. (1975). *Measurement of Motor Vehicle Operation Pertinent to Fuel Economy*, "SAE Technical Paper 750003.

Jupyter ORG. (2021). *Jupyter*. Obtenido de <https://jupyter.org/>

Naylampmechatronics. (28 de Diciembre de 2016). *CONFIGURACIÓN DEL MÓDULO BLUETOOTH HC-05 USANDO COMANDOS A*. Obtenido de Comandos AT HC-05 : https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usando-comandos-at.html

Ortiz, M. (2010). Reducción de las emisiones de CO₂ en vehículos de transporte: combustibles alternativos. *Revista Profesional, Técnica y Cultural de los Ingenieros Técnicos de Minas*,, 28-33.

Quinchimbla Pisuñam, F. E., & Solís Santamaría, J.M. (2017). *Desarrollo de ciclos de conducción en ciudad, carretera y combinado para evaluar el rendimiento real del combustible de un vehículo con motor de ciclo Otto en el distrito Metropolitano de Quito*, Quito.

Panadero, J. (3 de Julio de 2012). *Tecmovia*. Obtenido de Volvo: <https://www.diariomotor.com/tecmovia/2012/07/03/ecu-que-es-y-el-porque-de-su-existencia/>

Paz, J. I. (2005). La globalización: más que una amenaza es una oportunidad. *Revista EIA*, 21-34.

Pérez Llanos Pablo Santiago, Q. S. (2016). Determinación de los ciclos de conducción de un vehículo categoría M1 para la ciudad de cuenca. *Universidad politécnica saleciana sede matriz cuenca* , <https://dspace.ups.edu.ec/bitstream/123456789/15032/1/UPS-CT007421.pdf>.

Pineda, B., Muñoz, C., & Gil, H. (2018). Aspectos relevantes de la movilidad y su relación con el medio ambiente en el Valle de Aburrá: una revisión. *Ingeniería y desarrollo*, 489-508.

PowerBroker2. (6 de Mayo de 2019). *Github*. Obtenido de <https://github.com/PowerBroker2/ELMduino>

Rodríguez, I. C. (2011). Los sensores en el automóvil. *Tu taller mecánico*, 1-10.

Romero, M., Olite, D., Álvarez, F., & Toste, M. (2006). La contaminación del aire: su repercusión como problema de salud. *Revista Cubana de Higiene y Epidemiología*, 44.

Sanchez, F. A. (2015). Análisis, Diseño e implementación de un scanner automotriz para vehículos volkswagen gol, programado con arduino para visualizar en android. . *Universidad Tecnológica Equinoccial* , http://repositorio.ute.edu.ec/bitstream/123456789/4841/1/59406_1.pdf.

scientists, U. o. (23 de Julio de 2017). *Union of concerned scientists*. Obtenido de <https://es.ucsusa.org/recursos/carros-camiones-buses-contaminacion>

Wordpress. (22 de Octubre de 2017). *Aprendiendo Arduino*. Obtenido de <https://aprendiendoarduino.wordpress.com/2015/03/30/entradas-y-salidas-analogicas-pwm/>

ANEXO 1

En el siguiente anexo se puede evidenciar el código que se utilizó para configurar el módulo HC-05 mediante los códigos proporcionados por el fabricante. Aquí se establece los parámetros de velocidad de comunicación y al módulo como esclavo.

```
#include <SoftwareSerial.h> // Incluimos la librería SoftwareSerial
SoftwareSerial BT(10,11); // Definimos los pines RX y TX del Arduino conectados al Bluetooth

void setup()
{
  BT.begin(9600); // Inicializamos el puerto serie BT (Para Modo AT 2)
  Serial.begin(9600); // Inicializamos el puerto serie
}

void loop()
{
  if(BT.available()) // Si llega un dato por el puerto BT se envía al monitor serial
  {
    Serial.write(BT.read());
  }

  if(Serial.available()) // Si llega un dato por el monitor serial se envía al puerto BT
  {
    BT.write(Serial.read());
  }
}
```

ANEXO 2

Comandos utilizados por la librería ELMduino para obtener los valores de los sensores a través del OBD-II del vehículo.

```
uint32_t supportedPIDs_1_20();

uint32_t monitorStatus();
uint16_t freezeDTC();
uint16_t fuelSystemStatus();
float engineLoad();
float engineCoolantTemp();
float shortTermFuelTrimBank_1();
float longTermFuelTrimBank_1();
float shortTermFuelTrimBank_2();
float longTermFuelTrimBank_2();
float fuelPressure();
uint8_t manifoldPressure();
float rpm();
int32_t kph();
float mph();
float timingAdvance();
float intakeAirTemp();
float mafRate();
float throttle();
uint8_t commandedSecAirStatus();
uint8_t oxygenSensorsPresent_2banks();
uint8_t obdStandards();
uint8_t oxygenSensorsPresent_4banks();
bool auxInputStatus();
uint16_t runTime();
```

```
uint32_t supportedPIDs_21_40();

uint16_t distTravelWithMIL();
float fuelRailPressure();
float fuelRailGaugePressure();
float commandedEGR();
float egrError();
float commandedEvapPurge();
float fuelLevel();
uint8_t warmUpsSinceCodesCleared();
uint16_t distSinceCodesCleared();
float evapSysVapPressure();
uint8_t absBaroPressure();
float catTempB1S1();
float catTempB2S1();
float catTempB1S2();
float catTempB2S2();
```

```
uint32_t supportedPIDs_41_60();

uint32_t monitorDriveCycleStatus();
float ctrlModVoltage();
float absLoad();
float commandedAirFuelRatio();
float relativeThrottle();
float ambientAirTemp();
float absThrottlePosB();
float absThrottlePosC();
float absThrottlePosD();
float absThrottlePosE();
float absThrottlePosF();
float commandedThrottleActuator();
uint16_t timeRunWithMIL();
uint16_t timeSinceCodesCleared();
float maxMafRate();
uint8_t fuelType();
float ethonolPercent();
float absEvapSysVapPressure();
float evapSysVapPressure2();
float absFuelRailPressure();
float relativePedalPos();
float hybridBatLife();
float oilTemp();
float fuelInjectTiming();
float fuelRate();
uint8_t emissionRqmts();
```

ANEXO 3

El presente código es el ejemplo de la librería ELMduino para mostrar los valores de las RPM del vehículo a través del monitor serial. Empleando los comandos del Anexo 2 es posible acceder al resto de valores de los sensores.

```
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);

  Serial.begin(115200);
  ELM_PORT.begin(115200);

  Serial.println("Attempting to connect to ELM327...");

  if (!myELM327.begin(ELM_PORT))
  {
    Serial.println("Couldn't connect to OBD scanner");
    while (1);
  }

  Serial.println("Connected to ELM327");
}

void loop()
{
  float tempRPM = myELM327.rpm();

  if (myELM327.status == ELM_SUCCESS)
  {
    rpm = (uint32_t)tempRPM;
    Serial.print("RPM: "); Serial.println(rpm);
  }
  else
  {
    Serial.print(F("\tERROR: "));
    Serial.println(myELM327.status);
    delay(100);
  }
}
```

ANEXO 4

El siguiente código se emplea para generar la interfaz gráfica e importación de datos.

Para esto se utiliza la librería Pandastable que genera un Dataframe.

```
In [ ]: # creando la interfaz gráfica a partir de la librería
from tkinter import *
from pandastable import Table, TableModel

import pandas as pd #Librería para tabulación de datos a partir de un archivo TXT

class TestApp(Frame):

    def __init__(self, parent=None):
        self.parent = parent
        Frame.__init__(self)
        self.main = self.master
        self.main.geometry('600x400+200+100')
        self.main.title('DATALLOGGER')
        f = Frame(self.main)
        f.pack(fill=BOTH, expand=1)
        df = pd.read_csv('DATALOG.txt', sep=" ", header=0)
        self.table = pt = Table(f, dataframe=df,
                               showtoolbar=True, showstatusbar=True)

        pt.show()
        return

app = TestApp()
#Launch the app
app.mainloop()
```

ANEXO 5

El presente código se encarga de inicializar la conexión entre el OBD-II, SD, almacenar, filtrar y exportar los datos que son generados por el vehículo.

```
1 //LIBRERIAS
2 #include <Wire.h>
3 #include <SPI.h>
4 #include <SD.h>
5 #include "RTClib.h"
6 #include <SoftwareSerial.h>
7 #include "ELMduino.h"
8 #define ELM_PORT mySerial
9 RTC_DS1307 rtc;
10 #define ELM_PORT mySerial
11 SoftwareSerial mySerial(5, 6); // RX, TX
12 // VARIABLES
13 ELM327 myELM327;
14 uint32_t rpm = 0;
15 uint32_t temp = 0;
16 uint32_t pres = 0;
17 uint32_t temp2 = 0;
18 const int chipSelect = 10;
19 File dataFile;
20
21 void setup()
22 {
23 // INICIALIZACIÓN DE PUERTOS
24   Serial.begin(57600);
25   ELM_PORT.begin(57600);
26 // CONEXIÓN CON EL ELM327
27   Serial.println("Attempting to connect to ELM327...");
```

```

28
29 while (!myELM327.begin(ELM_PORT))
30 {
31     Serial.println("Intentando conectar");
32     delay(50);
33     ELM_PORT.begin(57600);
34
35 }
36 // CONEXIÓN EXITOSA, INTENTAR CONECTAR LA SD
37 Serial.println("Connected to ELM327");
38 Serial.print("Inicializando SD card...");
39
40 pinMode(chipSelect, OUTPUT);
41
42 while (!SD.begin(chipSelect)) {
43     Serial.println("SD no disponible o desconectada");
44
45 }
46 Serial.println("SD INICIADA.");
47
48 dataFile = SD.open("datalog.txt", FILE_WRITE);
49
50 while (!dataFile) {
51     Serial.println("error opening datalog.txt");
52     dataFile = SD.open("datalog.txt", FILE_WRITE);
53 }

```

```
54
55 while (!rtc.begin()) {
56     Serial.println("Couldn't find RTC");
57
58     }
59 String dataString1 = "";
60 dataString1 += String("HORA");
61 dataString1 += String(" ");
62 dataString1 += String("RPM");
63 dataString1 += String(" ");
64 dataString1 += String("°C MOTOR");
65 dataString1 += String(" ");
66 dataString1 += String("PRESION");
67 dataString1 += String(" ");
68 dataString1 += String("C");
69
70 dataFile.println(dataString1);
71 Serial.println(dataString1);
72 }
73
74
```



```

75 void loop()
76 {
77     float tempRPM = myELM327.rpm();
78     float tempTEMP = myELM327.engineCoolantTemp();
79     float tempPRES = myELM327.manifoldPressure();
80     float tempTEMP2 = myELM327.intakeAirTemp();
81
82     if (myELM327.status == ELM_SUCCESS)
83     {
84         if (0 < tempTEMP && 150 > tempTEMP){
85             temp = tempTEMP;
86         }
87         if (0 < (uint32_t)tempRPM && 60000 > (uint32_t)tempRPM){
88             rpm = (uint32_t)tempRPM;
89         }
90         if (0 < (uint32_t)tempPRES && 150 > (uint32_t)tempPRES){
91
92             pres = (uint32_t)tempPRES;
93         }
94         if (0 < tempTEMP2 && 150 > tempTEMP2){
95
96             temp2 = tempTEMP2;
97         }
98

```

```

98
99     Serial.print(rpm);
100    Serial.print(  "  ");
101    Serial.print(temp);
102    Serial.print(  "  ");
103    Serial.print(pres);
104    Serial.print(  "  ");
105    Serial.println(temp2);
106
107    DateTime now = rtc.now();
108    String dataString = "";
109
110    dataString += String(now.hour(), DEC);
111    dataString += String(':');
112    dataString += String(now.minute(), DEC);
113    dataString += String(':');
114    dataString += String(now.second(), DEC);
115    dataString += String(" ");
116    dataString += String(rpm);
117    dataString += String(" ");
118    dataString += String(temp);
119    dataString += String(" ");
120    dataString += String(pres);
121    dataString += String(" ");
122    dataString += String(temp2);
123
123
124    dataFile.println(dataString);
125    dataFile.flush();
126
127
128    }
129    else
130    {
131        Serial.print(F("\tERROR: "));
132        Serial.println(myELM327.status);
133        delay(1000);
134    }
135 }

```