UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE INGENIERÍA AUTOMOTRIZ

DISEÑO DE UN PANEL PARA VISUALIZAR LOS DATOS DE LA TARJETA MOBYDIC4910 OBTENIDOS A TRAVÉS DEL PROTOCOLO DE COMUNICACIÓN CAN BUS

Trabajo de titulación previo a la obtención del
Título de Ingeniero Automotriz

AUTORES: STIVEN ALEJANDRO NAULA TOAPANTA
MARCOS DANIEL NAVARRETE FIALLOS

TUTOR: CARLOS ALBERTO CARRANCO QUIÑÓNEZ

Quito - Ecuador

2023

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Stiven Alejandro Naula Toapanta con documento de identificación N° 1724367469 y Marcos Daniel Navarrete Fiallos con documento de identificación N° 1726781832, manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 18 de septiembre del año 2023

Atentamente,

Marcos Daniel Navarrete Fiallos
1726781832

Stiven Alejandro Naula Toapanta
1724367469

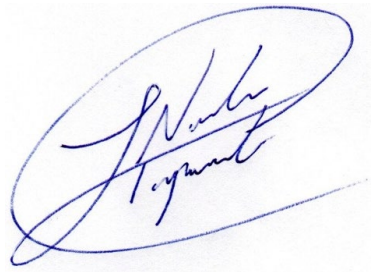
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Stiven Alejandro Naula Toapanta con documento de identificación N° 1724367469 y Marcos Daniel Navarrete Fiallos con documento de identificación N° 1726781832, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Diseño de un panel para visualizar los datos de la tarjeta mobydic4910 obtenidos a través del protocolo de comunicación can bus", el cual ha sido desarrollado para optar por el título de Ingenieros Automotrices, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana

Quito, 18 de septiembre del año 2023

Atentamente,



Marcos Daniel Navarrete Fiallos
1726781832

Stiven Alejandro Naula Toapanta
1724367469

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Carlos Alberto Carranco Quiñónez con documento de identificación N° 1713629564, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO DE UN PANEL PARA VISUALIZAR LOS DATOS DE LA TARJETA MOBYDIC4910 OBTENIDOS A TRAVÉS DEL PROTOCOLO DE COMUNICACIÓN CAN BUS, realizado por Stiven Alejandro Naula Toapanta con documento de identificación N° 1724367469 y Marcos Daniel Navarrete Fiallos con documento de identificación N° 1726781832, obteniendo como resultado final el trabajo de titulación bajo la opción: Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 18 de septiembre del año 2023

Atentamente,

Ing. Carlos Alberto Carranco Quiñónez, Ms.C.

1713629564

DEDICATORIA

Dedico el presente proyecto a mis padres que día con día me han apoyado dándome el estudio, forjando la responsabilidad en mi persona y entregando su esfuerzo para que yo tenga una profesión y pueda salir adelante.

Navarrete Fiallos Marcos Daniel

Dedico con un profundo sentimiento de agradecimiento el presente proyecto a mi familia, la cual me ha brindado la oportunidad de estudiar esta carrera, además de apoyarme día a día en el arduo camino de mi formación académica, además extiendo mi dedicatoria a mi compañero de proyecto de titulación Navarrete Daniel, quién me incluyo en el presente trabajo como su compañero de tesis.

Naula Toapanta Stiven Alejandro

AGRADECIMIENTO

Mis sinceros agradecimientos a mis padres y abuelos que me han acompañado día con día en el proceso de formación profesional, a mis padres les agradezco con toda sinceridad ya que por ellos me encuentro realizando este proyecto, me han dado todo lo necesario para estudiar y cumplir la meta de ser un Ingeniero Automotriz. A mis abuelos agradezco su infinito amor hacia mi persona, dándome todo lo mejor de ellos para seguir adelante.

Navarrete Fiallos Marcos Daniel

Agradezco a mis padres que gracias a su esfuerzo me pusieron en este lugar y momento, dándome todo lo necesario para seguir adelante en la carrera, apoyándome de todas las formas posibles. Agradezco a mis amigos cercanos los cuales me han brindado apoyo y han hecho más sencillo esta etapa de mi vida.

Naula Toapanta Stiven Alejandro

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN

En la actualidad el mundo se ha catapultado gracias a la tecnología, la cual va de la mano con el desarrollo de la industria automotriz, teniendo grandes avances en la tecnología digital que lleva el vehículo, la innovación abre puertas para la mejora continua, con la implementación de motores con mayor tecnología electrónica, aumenta el número de sensores, un mayor número de datos y fallos que se pueden presentar, para lo cual un tablero analógico no puede presentar dichas nuevas variables ya que muestra la información básica del vehículo.

En Ecuador la mayor parte de vehículos posee tecnología análoga en su tacómetro, siendo esta una tecnología que no muestra los datos completos del automotor, por lo cual el enfoque de este proyecto es la construcción de un dashboard digital que permita leer los datos de una tarjeta que simula la ECU (mobydic 4910) de un vehículo, a través de Arduino que se conecta con un módulo ELM327 y trasfiere los datos obtenidos de la ECU a una pantalla Nextion.

El objetivo principal es establecer una comunicación bidireccional entre la placa mobydic 4910 y un microcontrolador utilizando un adaptador ELM327. Este adaptador, conectado al puerto OBDII del vehículo, permite acceder a datos relevantes de sensores de la tarjeta. El código que se utiliza se basa en dos bibliotecas SoftwareSerial y ELMduino para establecer comunicaciones seriales con el adaptador ELM327 y un dispositivo de visualización, en este caso, un panel Nextion. Se crean dos objetos SoftwareSerial, uno para la comunicación con el adaptador ELM327 y otro para la comunicación con el panel Nextion.

Una vez establecida la comunicación, el código realiza consultas a través del adaptador ELM327 para obtener valores específicos de parámetros de la tarjeta, como las RPM del motor, la temperatura del refrigerante, la velocidad del vehículo, el nivel de combustible (tanto para el modo diésel como el modo gasolina) y la detección de códigos de fallo. Estos parámetros se utilizan para monitorear y visualizar en tiempo real los datos de la ECU.

Palabras Claves: ELM327**,** ECU, interfaz, bluetooth, mobydic 4910, Nextion, comandos AT, DTC’s, gasolina, diésel.

ABSTRACT

At present the world has been catapulted thanks to technology, which goes hand in hand with the development of the automotive industry, having great advances in digital technology that carries the vehicle, innovation opens doors for continuous improvement, with the Implementation of engines with more electronic technology, increases the number of sensors, a greater number of data and failures that can occur, for which an analog board cannot present these new variables since it shows the basic information of the vehicle.

In Ecuador, most vehicles have analog technology in their tachometer, this being a technology that does not show the complete data of the vehicle, for which the focus of this project is the construction of a digital dashboard that allows reading the data from a card that simulates the ECU (mobydic 4910) of a vehicle, through Arduino that connects with an ELM327 module and transfers the data obtained from the ECU to a Nextion screen.

The main objective is to establish a bidirectional communication between the mobydic 4910 board and a microcontroller using an ELM327 adapter. This adapter, connected to the vehicle's OBDII port, allows access to relevant data from the card's sensors. The code used is based on two libraries SoftwareSerial and ELMduino to establish serial communications with the ELM327 adapter and a display device, in this case a Nextion panel. Two SoftwareSerial objects are created, one for communication with the ELM327 adapter and one for communication with the Nextion panel.

Once communication is established, the code queries through the ELM327 adapter to obtain specific parameter values from the card, such as engine RPM, coolant temperature, vehicle speed, fuel level (for both mode diesel and gasoline mode) and fault code detection. These parameters are used to monitor and display ECU data in real time.

Keywords: ELM327, ECU, interface, bluetooth, mobydic 4910, Nextion, AT commands, DTC's, gasoline, diesel.

INTRODUCCIÓN

Actualmente, el diseño de sistemas fiables, intuitivos y precisos, para la visualización de datos, ha tomado cada vez más importancia en diferentes industrias y sectores; dentro del ámbito de la industria automotriz, esta necesidad se ve reflejada al momento de interpretar de manera eficiente la gran cantidad de información de todos los sensores y actuadores que conforman los diversos sistemas del vehículo. Por lo cual, en el contexto de la tarjeta Multiple Protocol ECU Simulator OZEN ELEKTRONIK LTD mOByDic 4910 como objeto de estudio, la realización de un dashboard digital para la visualización de datos, que usa como protocolo de comunicación al CAN Bus, se convierte en un elemento fundamental e innovador para aprovechar al máximo la información obtenida a través de la interpretación de las señales que se obtiene con el protocolo antes mencionado.

Una descripción generalizada, así como varias de las características básicas de la tarjeta Multiple Protocol ECU Simulator OZEN ELEKTRONIK LTD mOByDic 4910 se presentan en el Capítulo I, en el que, se destacan varios de los aspectos seleccionados de dicho simulador que posteriormente van a ser representados en el dashborad digital. En el Capítulo II, se presenta una breve descripción de la plataforma de comunicación OBD-II que tiene como objetivo servir como método de contextualización para enmarcar la función del dispositivo ELM327 dentro del desarrollo de la tesis, puesto que, es a través de este dispositivo que se va a realizar la comunicación con el entorno de desarrollo Arduino. En el Capítulo III se desarrollan los recursos utilizados para establecer la conexión entre la tarjeta y Arduino, haciendo énfasis principalmente en los PIDs necesarios para representar cada parámetro de las modalidades diésel y gasolina de la tarjeta.

En el Capítulo IV, se presenta todo el procedimiento metodológico realizado para poder conseguir la interpretación de las señales obtenidas a través del proceso de conexión de todos los dispositivos que representados en el diagrama de la figura 4.1, por lo que, fue necesario desarrollar un subcapítulo centrado en el desarrollo del código de programación que permitió tanto la extracción de los datos de la tarjeta mobydic4910, así como su posterior representación en una interfaz gráfica, además, se realizó una explicación del procedimiento utilizado para la clasificación y transformación de los datos los obtenidos a formato decimal, este paso es de los pilaras más importantes del proyecto ya que al clasificar y realizar la correcta interpretación de la información de los parámetros, se puede representar de manera

fiable la información enviada al variar la posición de cada potenciómetro, dichos datos posteriormente pasan a ser presentados en una interfaz amigable, la que permita visualizar todos los parámetros obtenidos a través de la comunicación de las herramientas de trabajo.

Los resultados obtenidos se describen en el apartado V, donde se realiza una subdivisión del capítulo, en función de cada interfaz realizada, para finalizar, se presentan las conclusiones en base a la realización del proyecto, así como las recomendaciones pertinentes relacionadas a aspectos que se pudieron aplicar durante el desarrollo del dashboard digital, que hubiesen facilitado varios procesos relacionados a la selección de herramientas de trabajo y optimización de tiempo.

PROBLEMA

Son varios los problemas que la tecnología analógica engloba dentro contexto moderno, por lo cual la transición en el reemplazo del uso de sistemas analógicos a digitales es un proceso que se ha visto progresivamente en diferentes tipos de industrias y tecnologías, pues este tipo de sistemas presentan varios inconvenientes que complican los procesos relacionados al almacenamiento, manipulación, comparación, calculo y recuperación de la información con la exactitud necesaria para su correcta interpretación.

Dentro del campo automotriz, el uso de tableros automotrices analógicos ha sido una práctica común dentro de la industria durante décadas, puesto que este tipo de tableros han sido una solución efectiva para proporcionar información importante al conductor durante el proceso de conducción de un vehículo, sin embargo, presentan ciertas desventajas y limitaciones en comparación con los más recientes avances tecnológicos (tableros digitales), de entre los cuales se pueden destacar los siguientes aspectos:

- **Precisión:** Los medidores analógicos pueden no ser tan precisos como los medidores digitales, especialmente después de un uso prolongado. El desgaste de los componentes y las fluctuaciones en la electricidad pueden hacer que los medidores no funcionen correctamente.
- **Falta de funcionalidad:** Los tableros analógicos suelen tener un número limitado de indicadores, lo que significa que no se pueden mostrar datos adicionales importantes, etc., aunque esto depende mayormente del fabricante.
- **Dificultad en la interpretación**: La interpretación de los indicadores analógicos puede ser difícil, especialmente para conductores con poca experiencia. Es difícil establecer de manera precisa y rápida la velocidad exacta del vehículo en comparación con los medidores digitales.

Además, hay que considerar las limitaciones que conllevan el uso de componentes mecánicos, pues en el caso de requerir una mejora y/o actualización, nos encontramos con una nula capacidad para realizar modificaciones. Esto significa que, si el fabricante no ha incluido un indicador específico, es probable que no se pueda agregar en el futuro, yaciendo en este aspecto su más importante desventaja y entre otras consideraciones relacionadas con el uso de elementos mecánicos, con esta estructura planteada se propone desarrollar una

interfaz intuitiva y amigable que presente los datos de manera clara, organizada y personalizable, facilitando así la interpretación de los datos recopilados.

DELIMITACIÓN DEL PROBLEMA

El proyecto se orienta en el diseño de un panel, que permite visualizar los datos de la tarjeta mOByDic4910 obtenidos a través del protocolo CAN BUS sin una restricción geográfica específica. Puede ser aplicado en cualquier ubicación geográfica donde se utilice o necesite la interfaz compatible a la tarjeta por medio del protocolo CAN bus. El trabajo de tesis se desarrollará pensando en la facilidad que brindará al momento de leer datos de los sensores automotrices presentes en los vehículos. No se establece una limitación temporal específica, lo que permite abordar el tema con enfoques y técnicas actuales.

Se enfocará para el uso y manipulación de los estudiantes de ingeniería Automotriz de la Universidad Politécnica Salesiana, pero eso no quiere decir que se pueda ocupar en los sectores económicos, empresariales e industriales que utilizan la tarjeta mOByDic4910 y el protocolo CAN BUS para la recopilación de datos. Esto puede incluir, pero no se limita a, la industria automotriz, la industria de fabricación, la industria de transporte y logística, y otros sectores que dependan del monitoreo y análisis de datos obtenidos a través del protocolo CAN BUS, no se limita a una institución o empresa en particular.

Puede ser aplicado tanto en entornos académicos como en organizaciones industriales y empresariales para recopilar datos. Los resultados y las recomendaciones derivadas de la tesis pueden ser adaptados e implementados por diferentes instituciones interesadas en mejorar el análisis de los datos obtenidos por medio del protocolo de comunicación.

OBJETIVO GENERAL

Diseñar un dashboard para la tarjeta Multiple Protocol OBD ECU Simulator (mOByDic4910), mediante el uso de una tarjeta electrónica para la visualización de señales de comunicación a través del protocolo CAN BUS, con el fin de mejorar la identificación y solución de altercados en los sistemas de control de vehículos.

OBJETIVOS ESPECÍFICOS

- Desarrollar una interfaz amigable para el tablero digital, utilizando elementos gráficos que sean fáciles de leer y que permitan al conductor y técnico automotriz

obtener rápidamente la información necesaria para operar y diagnosticar el vehículo de manera segura y eficiente.

- Obtener los parámetros de un tablero automotriz en una pantalla, a través de una tarjeta electrónica interconectada a la tarjeta Multiple Protocol ECU Simulator de Ozen Elektronik.
- Programar la tarjeta electrónica seleccionada para adquirir y procesar datos relevantes de la tarjeta Multiple Protocol ECU Simulator de Ozen Elektronik, como detención de fallos, lectura de señal de los sensores y otros indicadores de funcionamiento, con el fin de presentar esta información en el tablero digital.
- Realizar pruebas de desempeño del sistema de tablero digital utilizando diferentes escenarios de operación, asegurándose de que el sistema sea confiable, preciso y seguro para el usuario del dispositivo.

MARCO TEÓRICO

ECU

La unida de control electrónico (ECU) es un componente primordial en los vehículos modernos, controla y gestiona diferentes funciones y sistemas del automóvil. Una ECU automotriz es un módulo electrónico programable que supervisa y controla una o varias funciones específicas del vehículo, controla parámetros relacionados con varias funcionalidades referentes al funcionamiento del motor, el sistema de transmisión, el sistema de frenado, el sistema de suspensión, la seguridad, etc. La ECU recopila y procesa datos en tiempo real, toma decisiones basadas en algoritmos y emite comandos para ajustar y optimizar el funcionamiento de los sistemas controlados. interactúan con una variedad de sensores y actuadores. Los sensores proporcionan información sobre el estado y las condiciones del vehículo, los actuadores, por otro lado, reciben señales de la ECU para realizar acciones específicas, como regular la inyección de combustible, controlar válvulas, entre otros (Beniel Dennyson & Dr. C. Jothikumar, 2022).

Los diferentes tipos de ECU's presentes en los vehículos, se comunican entre sí a través de redes de comunicación, como el protocolo CAN Bus, el cual permite la transferencia y lectura de datos, así como la coordinación de las distintas funcionalidades del vehículo.

CAN BUS

El protocolo de comunicación Can Bus dentro del campo automotriz, opera como un estándar de comunicación serial que se basa en un enfoque de bus multi-master, lo que significa que varios dispositivos pueden transmitir y recibir datos simultáneamente en la red. Su importancia, se ve reflejada al funcionar como una de las herramientas con mayor relevancia dentro de la industria automotora; los vehículos actuales están dotados de pequeños sistemas integrados y unidades de control electrónico (ECU) que permiten la comunicación utilizando el protocolo CAN. Este protocolo ha representa un estándar en todos los automóviles y vehículos pesados desde 1996 en vehículos de procedencia norteamericana, pero a través de los años se hizo obligatorio en el año 2008.

El protocolo CAN BUS basa su funcionamiento en una topología de red en bus, donde los elementos se conectan mediante un único cable de comunicación. En esta arquitectura, todos los dispositivos de la red tienen acceso al bus compartido y pueden enviar y recibir mensajes.

OBD II

"El conector OBD II conocido como conector de enlace de diagnóstico II (DLC), el cual realiza la comunicación con la red interna del vehículo" (Beniel Dennyson & Dr. C. Jothikumar, 2022).

Permite la detección y el monitoreo de posibles fallas y malfuncionamientos en los sistemas y componentes del vehículo. Proporciona información detallada sobre el rendimiento del motor, emisiones, sensores, actuadores y otros parámetros críticos.

Utiliza códigos de diagnóstico estandarizados, conocidos como códigos de falla o códigos de problemas (DTC), por donde se identifica y describe posibles fallos en el auto(Yadav et al., 2021).

CONFIGURACIÓN DE PARÁMETROS OBD-II

Mediante el sistema OBD-II, se presenta la capacidad de acceder y modificar parámetros en el sistema de diagnosis a bordo del vehículo. Tales como:

- **Límite de advertencia de falla (DTC):** Ajustar el umbral en el cual se genera una advertencia de falla.
- **Intervalo de mantenimiento:** Se puede configurar el intervalo de mantenimiento, lo que implica programar el momento en el cual el automóvil mostrará una advertencia para realizar el proceso de mantenimiento regular.
- **Monitoreo de sensores:** Es posible habilitar o deshabilitar el monitoreo de sensores específicos con los que se desee trabajar.
- **Calibración del sensor:** En algunas cuestiones, se permite calibrar ciertos sensores, como el sensor de oxígeno, para mejorar la precisión de las mediciones.

PID CODIFICADO POR BITS

- **P:** El control proporcional es el componente básico del control PID. La salida del control proporcional es proporcional al error actual del sistema, es decir, la diferencia entre el valor esperado y el valor real medido.
- **I:** El control integral compensa los errores acumulativos a lo largo del tiempo. Calcula la integral del error a lo largo del tiempo y utiliza este valor acumulado para generar una corrección adicional en la salida del sistema.
- **D:** El control derivativo se basa en una tasa de cambio del error. Calcula la derivada del error con respecto al tiempo y utiliza este valor para predecir la tendencia futura del error (Gülnur_Karanfil, 2021, p. 12).

El identificador de parámetros (PID) identifica dichos parámetros, cada número hexadecimal se convierte en un sistema numérico de 2 bits y se indica en una forma de 4 bits. Cada dígito convertido se comprueba si es 1 o 0. Si 1, entonces se admite el PID especificado por el servicio (Gülnur_Karanfil, 2021).

LENGUAJE ARDUINO

Apoyado del lenguaje de programación ARDUINO: Arduino puede interactuar con hardware y software de código abierto que se utiliza para la creación de proyectos electrónicos. Es una placa electrónica programable que puede conectarse a actuadores y sensores y diversos componentes electrónicos para interactuar con el mundo físico (García-Tudela & Marín-Marín, 2023).

La tarjeta Multiple Protocol OBD ECU Simulator (mOByDic4910) sera parte de los elementos para elaborar el objetivo planteado. La ECU Ozen Elektronik LTD es capaz de simular diferentes protocolos de comunicación, como CAN, K-Line y LIN, y se puede programar para emular diferentes tipos de ECU's, como ECU de inyección electrónica, ECU de control de transmisión, ECU de control de aire acondicionado, entre otros, también se utiliza para probar y desarrollar software de diagnóstico de vehículos, herramientas de escaneo de diagnóstico y sistemas de calibración de vehículos.

El IDE de Arduino utiliza una variante simplificada del lenguaje para facilitar el aprendizaje y la programación, con una serie de bibliotecas predefinidas que proporcionan funciones y

características adicionales (Kondaveeti et al., 2021). La combinación de Arduino y el protocolo CAN BUS encuentra aplicaciones en diversos campos, como la automoción, con Arduino se podrá leer y analizar datos provenientes de sensores y actuadores CAN BUS de la tarjeta mOByDic 4910, así como para controlar y enviar comandos a dispositivos CAN BUS (García-Tudela & Marín-Marín, 2023).

MOBYDIC 4910

La tarjeta mobydic 4910 es un dispositivo electrónico utilizado en la industria automotriz para la simulación de señales correspondientes a los sensores y actuadores del motor de un vehículo, permite probar y verificar el funcionamiento de la unidad de control del motor (ECU) en diferentes escenarios, incluyendo pruebas de diagnóstico y desarrollo de software.

La tarjeta simula una sola o 4 ECU diferentes (PCM, TCM, HEC, ABS), protocolos que se regulan a través los distintos interruptores dispuesto en el cuerpo de la tarjeta. Además, posee funciones avanzadas de monitoreo, generación de códigos de fallo DTC para cada ECU por separado, envía PID variable con 8 potenciómetros que varían los datos recibidos.

“Es compatible con SAE-J1979 / ISO15031-5, posee una operación de multitrama / multi-mensajes , recibe energía eléctrica a través de un adaptador de corriente de 12 VDC” (OZEN ELEKTRONIK, n.d.).

ARDUINO

“La placa Arduino se describe como una herramienta de desarrollo de código libre diseñada para facilitar la creación de proyectos interactivos en el ámbito científico y tecnológico”.(Enríquez Herrador, 2009). Esta placa se compone de un microcontrolador programable, pines de entrada/salida que permiten la conexión con componentes electrónicos.

El microcontrolador, como el ATmega328P, es el corazón de Arduino y ejecuta instrucciones programadas para controlar diversas tareas. Junto con las conexiones de alimentación, que permiten la conexión de fuentes externas de energía, pines de entrada/salida digital y análoga leen las señales digitales y analógicas respectivamente. (Enríquez Herrador, 2009, p. 8). Esto posibilita la interacción con sensores y actuadores, como luces y motores.

Arduino también ofrece puertos de comunicación, como USB y UART, para facilitar la programación y la comunicación con otros dispositivos. El oscilador garantiza una sincronización adecuada de las operaciones del microcontrolador.

Su entorno de desarrollo integrado (IDE) facilita la programación de Arduino basándose en un lenguaje C/C++. (Enríquez Herrador, 2009, p. 17). Esto ha llevado a la formación de una comunidad activa de desarrolladores que comparten proyectos, tutoriales y bibliotecas de software, lo que ha contribuido a la accesibilidad y popularidad de Arduino.

El módulo Arduino presenta compatibilidad con diferentes tarjetas que manejan el mismo lenguaje de desarrollo; como lo es el módulo ELM327 y el ESP32.

El módulo ELM327 permite el enlace con el sistema OBD-II de la tarjeta mOByDic4910 o a su vez con el de un vehículo, cumpliendo con la función de trasferencia de los bytes con información enviada por las ECU's del dispositivo, los comandos de bytes se encuentran en formato ASCII, un lenguaje que envía dígitos en formato hexadecimal, que posteriormente deben ser convertidos a dígitos decimales siendo los bytes la información trasmitida por la tarjeta estudiada en cuestión. El ELM327 admite varios protocolos de comunicación utilizados en el estándar OBD-II, como el protocolo ISO 9141-2, el protocolo KWP2000, el protocolo J1850 PWM y el protocolo CAN BUS.

Posee un convertidor de señales analógicas a digitales para realizar las mediciones de voltaje, por lo que, varios tipos de dispositivos para el diagnóstico automotriz están equipado con el controlador ELM327 con el fin de detectar instintivamente los protocolos OBD.

Los protocolos admitidos por la ELM327 definen cómo se transmiten y reciben los datos del vehículo, y el ELM327 es capaz de interpretar y comunicarse en estos diferentes protocolos. El ELM327 se utiliza junto con software de diagnóstico compatible para visualizar y analizar los datos del vehículo, tal como Arduino.(Kondaveeti et al., 2021). Existen diferentes aplicaciones móviles y software para equipos de escritorio, que permiten la comunicación con el ELM327 y la interpretación de los datos OBD-II. Estos programas proporcionan funciones como lectura y borrado de códigos de falla, monitoreo en tiempo real de parámetros de la tarjeta mOByDic 4910, registro de datos, entre otros (Eduardo Arévalo Calderón & Angel Geovanny Ortega Ulloa, 2016, pp. 9–10).

BLUETOOTH

El Bluetooth es tecnología inalámbrica de corto alcance que permite el entrelazamiento de módulos electrónicos. Su funcionamiento es a través de ondas de radio, las cuales trasmiten datos de manera segura y confiable. Una de las ventajas del Bluetooth es su excelente rendimiento con un bajo consumo de energía, esto hace que sea una opción ideal para dispositivos que consuman poca energía y sea compactos. La tarjeta ESP32 es una placa de desarrollo de 32 bits con conectividad WiFi y Bluetooth integrada. El ESP32 proporciona una amplia variedad de interfaces y pines de entrada/salida (Zakaria et al., 2023).La conectividad Bluetooth integrada en el ESP32 brinda la posibilidad de establecer conexiones inalámbricas con otros dispositivos compatibles, permitiendo la transferencia de datos y la interacción entre ellos.

DASHBOARD

Un dashboard (tablero) es una representación visual de datos e información, presentada en un formato gráfico y fácil de entender, suele estar compuesto por gráficos, tablas y otros elementos visuales que permiten al usuario obtener una vista general de la información de manera sencilla, rápida e intuitiva. Estos elementos pueden ser personalizados para mostrar datos específicos de los sensores que están presentes en la tarjeta mobydic 4910, también puede permitir al usuario interactuar con ellos variando los potenciómetros que a su vez envían señales que son atrapadas con el código generado en Arduino, se puede filtrar o clasificar los datos, proporciona una visión holística y consolidada de los datos, lo que facilita la comparación y el análisis de diferentes métricas y variables (Gülnur_Karanfil, 2021, pp. 85–90).

El uso de gráficos y elementos visuales en un dashboard también puede mejorar la comprensión y comunicación de los datos al presentar la información de manera visualmente eficiente, pues se facilita la interpretación y la toma de decisiones basadas en los parámetros dispuestos en la interfaz.

DISPLAY -NEXTION ARDUINO

“Display compatible para Arduino” es un dispositivo que combina una pantalla TFT (Thin Film Transistor) a color y una interfaz que permitirá la visualización de los datos de la tarjeta mobydic 4910 (Cavazos Luis Enrique et al., 2022).

La pantalla Nextion para Arduino es un dispositivo inteligente de pantalla táctil programable diseñado específicamente para su integración en proyectos basados en Arduino. Estas pantallas se caracterizan por la facilidad de uso permitiendo realizar una interfaz de usuario interactiva para el dashboard.

Nextion son pantallas de formato de matriz activa de transistor de película delgada de alta calidad, que ofrecen una resolución nítida y colores vibrantes. Lo que distingue a estas pantallas es su capacidad de operar de manera independiente del Arduino, gracias a la inclusión de un microcontrolador y memoria interna propios (Cavazos Luis Enrique et al., 2022). Esta característica reduce la carga de trabajo del Arduino al procesar los gráficos y almacenar datos relacionados con la interfaz gráfica.

Su funcionamiento se basa en la conexión con una placa Arduino u otro microcontrolador compatible. “Utilizando bibliotecas de programación específicas, se puede controlar la visualización de la información en la pantalla y capturar eventos táctiles para realizar acciones correspondientes” (Cavazos Luis Enrique et al., 2022, pp. 2–3).

“Se puede manejar como interfaz de usuario para controlar y monitorear dispositivos, de cualquier índole siempre y cuando sean compatibles con la programación base que es Arduino, entre otros” (Gülnur_Karanfil, 2021). Además, el “display TFT táctil Arduino” puede utilizarse para visualizar datos en tiempo real del módulo ELM 327, presentando gráficos interactivos de la variación de los distintos sensores existentes en la modydic 4910.

NEXTION HMI

NEXTION HMI (Human Machine Interface) es un programa y una serie de pantallas táctiles diseñadas para la creación de interfaces para proyectos de electrónica con cualquier enfoque. El programa NEXTION HMI proporciona una forma sencilla de diseñar y desarrollar interfaces gráficas interactivas sin requerir conocimientos avanzados de programación. (FÉLIX ANDRÉS ANDRADE SÁNCHEZ, 2015)

El software NEXTION Editor e permite a los usuarios diseñar las interfaces de usuario de forma visual. Con una interfaz intuitiva basada en arrastrar y soltar, los usuarios pueden crear botones, gráficos, indicadores y otros elementos de la interfaz. Además, pueden establecer acciones para realizar cuando se interactúa con estos elementos, como mostrar información, cambiar el estado de un dispositivo o enviar comandos a otros componentes ((Kondaveeti et al., 2021).

INKSCAPE

"Inkscape es una aplicación de software de gráficos vectoriales de código abierto y multiplataforma. Proporciona un conjunto de herramientas completo y versátil para la creación y edición de gráficos vectoriales" (inkscape.es, 2023).

A diferencia de las imágenes de mapa de bits, los gráficos vectoriales basan su desarrollo en ecuaciones matemáticas y descripciones geométricas para definir formas, líneas y colores. Esto permite que los gráficos sean escalables sin pérdida de calidad, lo que los hace ideales para diseños que necesitan ser redimensionados o impresos en diferentes tamaños, proporciona opciones de edición avanzadas, (inkscape.es, 2023) como la manipulación de nodos, combinación de objetos, capas y transformaciones. Además, admite el uso de filtros y efectos para añadir estilo y mejorar las imágenes.

CAPÍTULO I

1. SIMULADORES DE COMPUTADORAS AUTOMOTRICES

Por motivos inherentes a la aplicación de los conocimientos adquiridos durante el proceso de formación académica de la carrera de Ingeniería Automotriz, el presente trabajo de titulación, requiere de investigación y clasificación de datos, en conjunto de la ardua labor de programación en Arduino con el propósito de lograr el cumplimiento de los objetivos propuestos para la realización de este proyecto; por lo cual en base a todo el procedimiento metodológico que se realizará, se tiene previsto, culminar con el desarrollo de un dashboard digital que cumpla con las condiciones necesarias para brindar al usuario un monitoreo efectivo del rendimiento de la tarjeta que emula una ECU.(Smith, 2006)

En este primer capítulo, se explorará la importancia, características principales y, el papel de los simuladores de ECU en el campo del desarrollo y diagnóstico automotriz. Se analizarán los diversos usos y beneficios que ofrecen estos dispositivos, así como su aplicación en el desarrollo de software, validación y verificación, diagnóstico y resolución de problemas, para su aplicación en programas de formación y educación.

- **Desarrollo y pruebas de software de control del motor:** Los simuladores de ECU cumplen un papel fundamental en el desarrollo y pruebas de software de control para los vehículos. Proporcionan a los ingenieros de software un entorno controlado para probar y depurar el código de la ECU antes de su implementación en un vehículo real. Esto ayuda a reducir los costos y tiempos relacionados con el desarrollo y pruebas del dispositivo, al mismo tiempo que asegura un funcionamiento óptimo del software de control. (Zambrano et al., 2015).
- **Validación y verificación del software de la ECU:** La validación y verificación del software de la ECU son etapas críticas en el proceso de desarrollo. Los simuladores de ECU permiten recrear diferentes escenarios de funcionamiento del vehículo y verificar si el software de control responde de manera adecuada en cada situación. Esto contribuye a garantizar la confiabilidad y seguridad del software antes de su implementación en un vehículo real, evitando posibles fallas y problemas en el campo.
- **Diagnóstico y resolución de problemas de las ECUs:** Los simuladores de ECU son herramientas valiosas en el diagnóstico y resolución de problemas de las ECU’s. Al

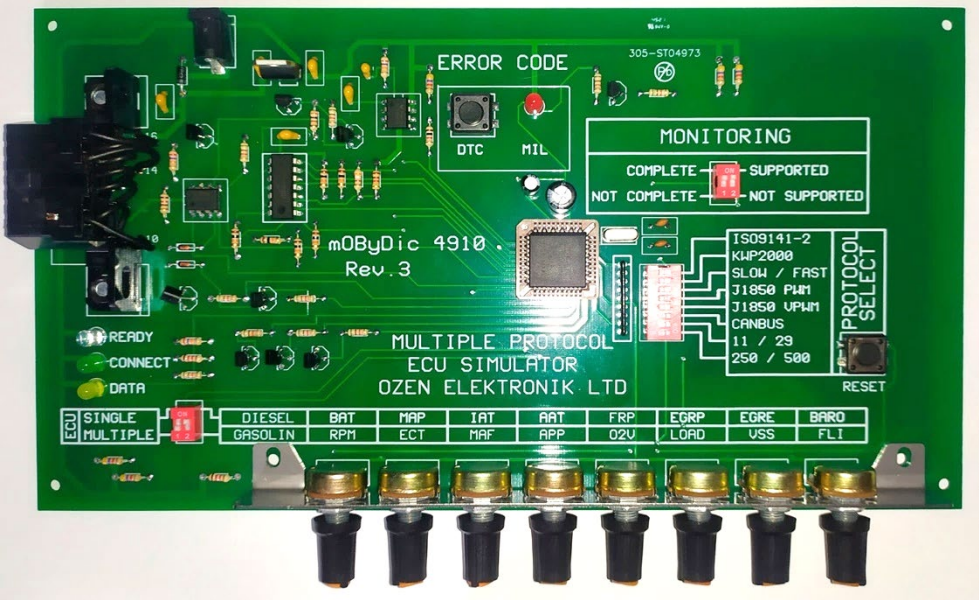
simular diferentes señales y condiciones de funcionamiento, ayudan a identificar posibles fallas o problemas en la ECU. Esto permite a los técnicos realizar diagnósticos precisos y resolver problemas de manera más eficiente, minimizando el tiempo de inactividad del vehículo y optimizando el rendimiento de la ECU.

- **Aplicación en programas de formación y educación automotriz**: Los simuladores de ECU también cumplen un papel importante en los programas de formación y educación dentro del campo automotriz. Proporcionan a los estudiantes una plataforma segura y controlada para aprender aspectos relacionados al funcionamiento de las ECU's, a su vez que ayudan a practicar poner en práctica sus habilidades de diagnóstico para la resolución de problemas. Esto permite a los futuros técnicos automotrices adquirir experiencia práctica sin la necesidad de acceder a vehículos reales, lo que facilita su aprendizaje y desarrollo profesional.

Es por los puntos expuesto con anterioridad, que se seleccionó la tarjeta Multiple Protocol ECU Simulator OZEN ELEKTRONIK LTD mOByDic 4910 como objeto de estudio para la realización de este proyecto técnico; en los siguientes subapartados correspondientes al Capítulo I se detallaran varios aspectos referentes a las características técnicas y de funcionamiento de la tarjeta Multiple Protocol ECU Simulator OZEN ELEKTRONIK LTD mOByDic 4910, a la que con fines de simplificar el desarrollo del contenido se referirá únicamente como "mobydic 4910".

1.1. MULTIPLE PROTOCOL OBD ECU SIMULATOR MOBYDIC 4910

Figura 1.1: Tarjeta mobydic 4910.



Tarjeta Multiple Protocol ECU Simulator OZEN ELEKTRONIK LTD mOByDic 4910.

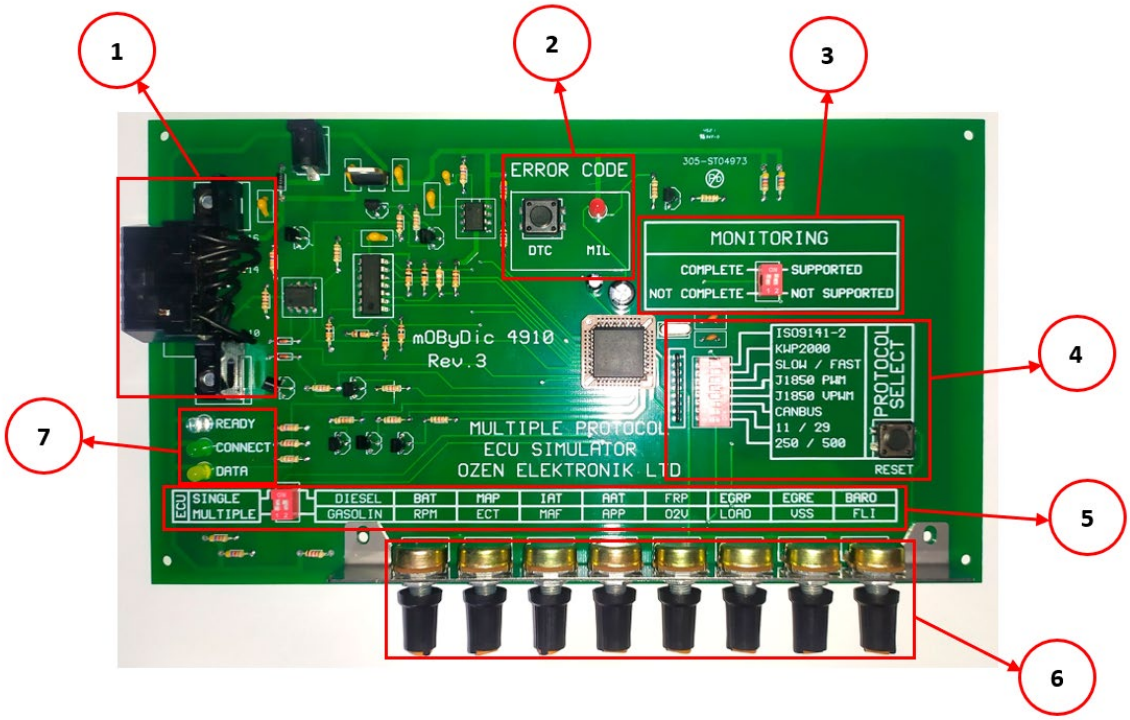
Fuente: Autores.

La tarjeta mobydic 4910, es un dispositivo que simula el comportamiento de las computadoras automotrices presentes en los vehículos automóviles, dentro del ámbito automotriz, es utilizada para el diagnóstico y la simulación de unidades de control electrónico (By ALLDATASHEETCOM, 2002). Referente al contexto de la diagnosis de sistemas eléctricos y electrónicos de vehículo automóviles, funge como una herramienta sumamente versátil, puesto que permite a los estudiantes, técnicos y expertos en automoción acceder y comunicarse con las ECUs simuladas presentes en la tarjeta.

Dentro de las características principales que posee a la tarjeta mobydic 4910, se destacan sus múltiples funcionalidades referentes a la simulación de múltiples protocolos de comunicación, hasta 4 tipos de ECUs automotrices, 8 PIDs regulados por potenciómetros que varían su función dependiendo del tipo de motor a simular(OZEN ELEKTRONIK, n.d.), además de que cuenta con una función para generar códigos de fallo (DTC); con la finalidad

de facilitar la clasificación de los elementos que compone esta tarjeta, se realizó el siguiente esquema:

Figura 1.2: Partes de la tarjeta mobydic 4910.



Partes principales de la tarjeta mobydic 4910, Fuente: Autores.

1. **Conector OBD-II:** Es la interfaz mediante la que se establecerá la comunicación con el dispositivo de diagnóstico automotriz.
2. **Panel de código de errores:** Cuenta con un botón “DTC” que al ser presionado generará diferentes códigos de errores en las ECU, el led “MIL” se activa al presionar el botón.
3. **Panel de monitoreo:**
4. **Panel de selección de protocolo:** Cuenta con un botón de “RESET” que sirve para el reinicio de la tarjeta en caso de querer cambiar ciertas características operativas, además, en caso de querer seleccionar un protocolo de comunicación específico para la lectura de datos, posee un interruptor DIP de 8 palancas que regula las siguientes funcionalidades:
    - **Palanca 1:** Activa o desactiva el protocolo de comunicación ISO9141-2.
    - **Palanca 2:** Activa o desactiva el protocolo de comunicación KWP2000.

- **Palanca 3:** Configura la velocidad de comunicación entre lenta o rápida (SLOW/FAST).
- **Palanca 4:** Activa o desactiva el protocolo de comunicación J1850 PWM.
- **Palanca 5:** Activa o desactiva el protocolo de comunicación J1850 VPWM.
- **Palanca 6:** Activa o desactiva el protocolo de comunicación CAN-BUS.
- **Palanca 7:** Establece si la comunicación del protocolo CAN-BUS será con 11 o 29 bits.
- **Palanca 8:** Sirve para configurar la velocidad de transmisión entre 250 o 500 baudios para el protocolo CAN-BUS.

5. **Panel ECU:** Posee un interruptor DIP de 2 palancas:
    - **Palanca 1:** Sirve para seleccionar la opción "SINGLE" donde únicamente responderá el PCM, o la opción "MULTIPLE" donde responderán las 4 ECUs.
    - **Palanca 2:** Sirve para seleccionar el modo diésel o gasolina.
6. **Conjunto de potenciómetros:** Son un conjunto de 8 potenciómetros que sirven para variar los valores de cada sensor simulado por la tarjeta mobydic 4910.
7. **Leds indicadores:** Son un conjunto de 3 leds, cuyo propósito es el de indicar el estado de funcionamiento de la tarjeta:
    - **Led READY:** Sirve para determinar si la tarjeta está lista para su uso, se activa cuando energizamos la tarjeta.
    - **Led CONNECT:** Se activa cuando se establece la conexión entre la tarjeta y el dispositivo de diagnóstico automotriz.
    - **Led DATA:** Se activa cuando la lectura de datos comienza, brillará intermitentemente durante dicho proceso.

1.1.1. ECU'S SIMULADAS

Una ECU (Unidad de Control Electrónico) automotriz es un componente esencial en los vehículos modernos que se encarga de controlar y gestionar diversos sistemas y funciones del automóvil. (Yadav et al., 2021)La ECU es un tipo de computadora que recopila datos de diferentes sensores del vehículo y toma decisiones en tiempo real; para controlar el funcionamiento de los sistemas, la tarjeta mobybic 4910 según (OZEN ELEKTRONIK, n.d.), permite la simulación de 4 ECU's:

- **PCM (Módulo de Control del Tren de Potencia):** La PCM controla el motor y otros sistemas relacionados con la propulsión, como la inyección de combustible, la ignición, la gestión de la transmisión y la monitorización de las emisiones.
- **TCM (Módulo de Control de la Transmisión):** El TCM se encarga del control y gestión de la transmisión del vehículo, incluyendo el cambio de marchas, el embrague (en caso de transmisiones automáticas) y el monitoreo de los sensores de velocidad y posición.
- **HEC (Módulo de Control del Entorno del Habitáculo):** El HEC se ocupa de controlar y supervisar las funciones relacionadas con el habitáculo, como la iluminación, la climatización, los sistemas de entretenimiento y otros sistemas eléctricos del vehículo.
- **ABS (Sistema de Frenos Antibloqueo):** El ABS es responsable de controlar y gestionar el sistema de frenos de antibloqueo, proporcionando un frenado seguro al evitar el bloqueo de las ruedas cuando se presentan situaciones de frenado difíciles.

1.1.2. NORMAS EN LA MOBYDIC 4910

1.1.2.1.SAE-J1979

Es un modelo de comunicación que se ocupa en la industria automotriz para el diagnóstico a bordo de vehículos. Fue hecho por la Society of Automotive Engineers (SAE) y establece un conjunto común de reglas y protocolos para permitir la comunicación de los dos sistemas de diagnóstico y los módulos electrónicos de un automotor.

El estándar SAE-J1979 define un protocolo de comunicación basado en mensajes y códigos que permite acceder y extraer información de los sistemas electrónicos del vehículo. Estos sistemas incluyen la ECU, la Transmisión Automática (TCM), el Sistema de Frenos Antibloqueo (ABS) y otros módulos relacionados con el rendimiento y la seguridad del vehículo.

SAE-J1979 establece la comunicación por medio de del conector OBD-II en la cabina del vehículo, que brinda acceso a los datos generados por los sistemas electrónicos. Además, define códigos de diagnóstico de problemas (DTC) estandarizados, los cuales identifican y comunican fallas específicas. Los códigos permiten a los profesionales y sistemas de

diagnóstico realizar una evaluación precisa y eficiente de los problemas del vehículo según, (Society of Automotive Engineers, 2002).

Los PID's que proporcionan información en tiempo real sobre las condiciones y operación de un vehículo. Estos PIDs se agrupan en conjuntos llamados servicios o modos OBD.

- Modo 1. Identificación de Parámetro (PID), es el acceso a datos en vivo de valores analógicos o digitales de salidas y entradas a la ECU.
- Modo 2. Acceso a Cuadro de Datos Congelados (Freeze Frame Data).
- Ésta es una función necesaria del OBD-II debido a que la ECU escoge una muestra de los valores relacionados con las emisiones, en el momento exacto de ocurrir un fallo.
- Modo 3. Extrae de la memoria de la ECU los códigos de fallo (DTC) guardados.

Tabla 1.1: Tabla de modos de prueba.

Condición de prueba	Tipo de mensaje bit 6	Norma
00 – 0F	Requerimiento bit 6=0	SAE J1979
10 - 1F		SAE J2190
20 - 2F		
30 - 3F		

Parámetros de bits de las normas SAE. Fuente: (Society of Automotive Engineers, 2002).

SAE-J1979 establece la comunicación a través del puerto OBD-II, que brinda acceso a los datos generados por los sistemas electrónicos. Además, define códigos de diagnóstico de problemas (DTC) estandarizados, los cuales identifican y comunican fallas específicas. Los códigos permiten a los técnicos y sistemas de diagnóstico realizar una evaluación precisa y eficiente de los problemas del vehículo.

1.1.2.2. ISO 150231–5

SO 15031-5 define un conjunto de protocolos de comunicación que permiten a los equipos de diagnóstico externos acceder a las funciones y datos de los sistemas de las ECU's presentes en el automotor. Estos protocolos de comunicación proporcionan una interfaz estandarizada.

La norma ISO 15031-5 establece los requisitos técnicos para la implementación de protocolos de comunicación, como el acceso a los datos de diagnóstico, la identificación de los módulos electrónicos del vehículo, la recuperación de códigos de diagnóstico y la interacción con los sistemas de control, además, ISO 15031-5 también aborda aspectos relacionados a la reserva de los datos de diagnóstico. Establece medidas para garantizar la confidencialidad de datos.(ISO, 2011).

1.1.3. PROTOCOLOS DE COMUNICACIÓN DE LA TARJETA MOBYDIC 4910

Los protocolos de comunicación automotriz forman el conjunto de reglas y estándares que definen la forma en que los diferentes componentes electrónicos de un vehículo se comunican entre sí. Estos protocolos establecen el formato de los mensajes, la tasa de la velocidad de transmisión, los tipos de datos y las funciones que se pueden realizar a través de la comunicación. Los protocolos de comunicación automotriz permiten que las múltiples ECU's que controlan el funcionamiento de los sistemas de un vehículo, intercambien información y controlen sus operaciones de manera coordinada:

- **ISO 9141-2:** Este protocolo comúnmente se utiliza en automotores europeos y asiáticos. Permite el enlace bidireccional entre la ECU y el scanner de diagnóstico, utilizando una velocidad de transmisión de 10.4 kbps.
- **KWP2000 (Keyword Protocol 2000):** Es un protocolo que permite la comunicación, utilizado en gran parte de los vehículos europeos. Ayuda a la comunicación bidireccional y ofrece velocidades de transmisión de hasta 125 kbps.
- **J1850 PWM (Pulse Width Modulation):** Es un protocolo utilizado en vehículos americanos y pocos europeos. Utiliza una señal de modulación por ancho de pulso para transmitir datos y la velocidad de transmisión de 41.6 kbps.
- **J1850 VPW (Variable Pulse Width):** La diferencia con el protocolo J1850 PWM, es que utiliza una codificación de pulso variable y con una velocidad de transmisión de 10.4 kbps, también se utiliza en automotores americanos y algunos europeos.

- **CANBUS (Controller Area Network):** Protocolo de comunicación moderno y utilizado en la automoción. Permite la comunicación bidireccional y ofrece velocidades de transmisión de hasta 1 Mbps. CANBUS se ocupa el control del motor, ABS, airbags y otros.

1.1.3.1. ISO 15765-4 CAN (29 bit ID, 500 kbaud)

Durante el proceso de formación académica de la carrera de Ingeniería Automotriz, el protocolo de comunicación más explorado fue el CAN Bus, motivo por el cual ha sido seleccionado como protocolo de comunicación a usar en la elaboración de este proyecto. En lo que respecta a la tarjeta mobydic 4910, esta cuenta con 4 protocolos de comunicación CAN Bus:

- ISO 15765-4 CAN (11 bit ID, 250 kbaud)
- ISO 15765-4 CAN (11 bit ID, 250 kbaud)
- ISO 15765-4 CAN (29 bit ID, 500 kbaud)
- ISO 15765-4 CAN (29 bit ID, 500 kbaud)

Y, en base al procedimiento iterativo aplicado para la recolección de datos, se decidió optar por el uso del protocolo ISO 15765-4 CAN (29 bit ID, 500 Kbaud), aunque en general, en comparación con protocolos CAN Bus diferentes en el enlace de la tarjeta, no existen diferencias significativas durante la comunicación. Dando continuidad, se presentarán las especificaciones principales de este protocolo:

Tabla 1.2: Tabla características ISO 15765-4 CAN (29 bit ID, 500 kbaud)

Característica	Valor
Norma	ISO 15765-4
Tipo de red	CAN (Controller Area Network)

Característica	Valor
Identificación del mensaje	29 bits
Velocidad de transmisión	500 Kbaud
Modo de transmisión	Half-Duplex (unidireccional)
Tipo de trama	Estándar (11 bits) y extendida (29 bits)
Prioridad del mensaje	No hay prioridades definidas
Tamaño máximo del paquete de datos	8 bytes
Mecanismo de detección de errores	Control de errores CAN
Soporte de mensajes multiplexados	Sí

Característica	Valor
Mecanismo de direccionamiento	ID de 29 bits
Capacidad de direccionamiento	Hasta 536,870,912 direcciones
Aplicaciones típicas	Comunicación de datos en vehículos, diagnóstico automotriz

Descripción protocolo de comunicación a usar en la elaboración de este proyecto, Fuente: Autores.

1.1.4. MODALIDADES DE FUNCIONAMIENTO

Dependiendo del tipo de datos que se requiera visualizar, la tarjeta mobydic 4910, cuenta con dos interruptores que regulan el modo de funcionamiento correspondiente a que motor se va a representar, siendo estos: diésel y gasolina. Además, cuenta con una funcionalidad, que se activa a través de un botón, que permite la generación de códigos de fallo (DTC).

1.1.4.1. MODO DIÉSEL

Para configurar la modalidad diésel en la tarjeta mobydic 4910, se tiene que posicionar la palanca 2 del interruptor DIP de dos posiciones del panel ECU en el modo diésel, con lo cual se podrá visualizar los siguientes 8 parámetros:

- **BAT (Battery Voltage):** Mide la tensión o voltaje de la batería de un vehículo. Suministra información respecto al estado de carga de la batería, que es vital para el adecuado trabajo del sistema eléctrico del vehículo.
- **MAP (Presión Absoluta del Colector):** Mide la presión absoluta en el colector de admisión.

- **IAT (Temperatura del Aire de Admisión):** Mide la temperatura del aire que ingresa al sistema de admisión.
- **AAT (Temperatura del Aire Ambiente):** Mide la temperatura del aire en el entorno.
- **FRP (Presión de Combustible de Riel):** Mide la presión del combustible en el riel de inyección.
- **EGRP (Válvula de Recirculación de Gases de Escape):** Simula el funcionamiento de la válvula de recirculación de gases de escape.
- **EGRE (Válvula de Recirculación de Gases de Escape):** Simula el funcionamiento de la válvula de escape de gases de escape en modo error.
- **BARO (Presión Barométrica):** Mide la presión atmosférica en el entorno del vehículo

Como se manifiesta en el manual de (OZEN ELEKTRONIK, n.d.).

1.1.4.2. MODO GASOLINA

Para configurar la modalidad gasolina en la tarjeta mobydic 4910, se tiene que posicionar la palanca 2 del interruptor DIP de dos posiciones del panel ECU en el modo gasolina, con lo cual se podrá visualizar los siguientes parámetros:

- **RPM (Revoluciones por minuto):** Representa la velocidad de rotación del motor en rpm's.
- **LOAD (Carga del motor):** Indica la carga del motor en porcentaje.
- **VSS (Velocidad del vehículo):** Muestra la velocidad del vehículo en kilómetros/hora.
- **FLI (Indicador de nivel de combustible):** Representa el nivel de combustible en el tanque del vehículo.
- **ECT (Temperatura del refrigerante del motor):** Representa la temperatura actual del refrigerante del motor.
- **MAF (Flujo de aire masivo):** Representa el flujo de aire masivo que ingresa al motor.
- **APP (Posición del pedal del acelerador):** Muestra la posición del pedal del acelerador en porcentaje.

- **O2V (Voltaje del sensor de oxígeno):** Indica el voltaje del sensor de oxígeno del sistema de escape

Como se indica en el manual de (OZEN ELEKTRONIK, n.d.).

Figura 1.3: Modos DIESEL/GASOLINA de la tarjeta mobydic 4910.

Modo de selección de gasolina y diésel. Fuente: Autores.

1.1.4.3. MODO DTCs

Para activar la configuración de generación de fallos (DTC), únicamente será necesario presionar el botón DTC, presente en el panel ERROR CODE, con lo que se podrá apreciar los siguientes fallos, respecto a cada ECU, según (OZEN ELEKTRONIK, n.d.):

En el ECU PCM se presentarán los siguientes fallos:

- **P0100:** Indica un problema en el circuito del caudalímetro de masa de aire. Puede ser causado por un sensor MAF defectuoso, un cableado dañado o una conexión suelta.
- **P0101:** Se genera cuando el sensor MAF no está proporcionando una lectura de la cantidad (flujo) de aire adecuada a la ECM que es el módulo de control del motor. Puede deberse a un sensor MAF sucio, dañado o mal conectado, o también puede indicar baja de presión en la admisión.
- **P0102:** Este código se produce cuando el voltaje de salida del sensor MAF está por debajo del rango esperado. Puede ser causado por un sensor MAF defectuoso, un circuito de señal abierto o una mala conexión eléctrica.

En la ECU TCM, se generarán los siguientes fallos:

- **B0200:** Falla en el circuito del sensor de impacto frontal del puesto del conductor. Este código indica que existe un fallo con el circuito eléctrico o el sensor de impacto

frontal posicionado en el lado del conductor del vehículo. Puede ser causado por un cableado defectuoso, una conexión suelta, un sensor de impacto dañado o problemas en el módulo de la bolsa de aire.

- **B0201:** Falla en el circuito del sensor de impacto frontal del lado del pasajero. Este código indica que existe un fallo en el circuito eléctrico o el sensor de impacto frontal ubicado en el lado del pasajero del vehículo. Al igual que en el caso anterior, puede ser causado por un cableado defectuoso, una conexión suelta, un sensor de impacto dañado o problemas en el módulo de la bolsa de aire.

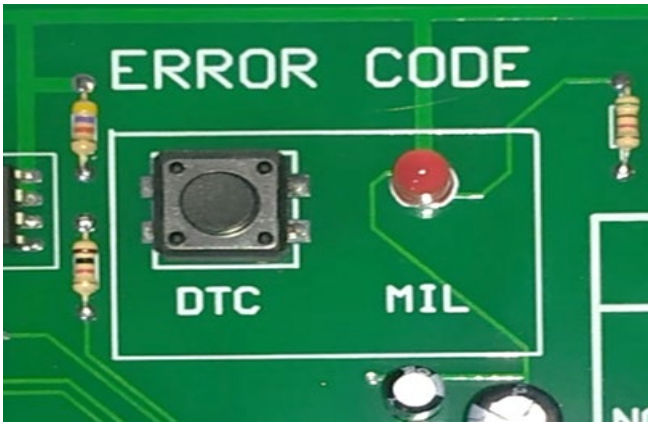
En la ECU ABS se generará únicamente un fallo:

- **C0300:** Se refiere a una dificultad detectado en el sistema de encendido del automotor. Específicamente, el código C0300 indica un fallo en la bobina de encendido o en alguno de los cilindros del motor.

En la ECU HEC, únicamente se generará un fallo:

- **U0400:** Se da debido a un problema de comunicación con la red de control del vehículo. Este código muestra que la comunicación con el módulo de control electrónico (ECM) se ha perdido con uno o más módulos de control adicionales.

Figura 1.4: Panel ERROR CODE



Panel de códigos de generación de códigos de errores de la tarjeta mobydic 4910.

Fuente: Autores.

CAPÍTULO II

2. PLATAFORMA DE COMUNICACIÓN OBD-II

La plataforma de comunicación OBD-II es la actualización del sistema de diagnóstico a bordo y sucesora del sistema OBD-I, entre las características de este "nueva" plataforma, se presentan la opción de detectar varios tipos de fallos que el sistema OBD-I no permitía.

Para ofrecer la máxima información posible para el técnico automotriz, el sistema OBD-II tiene la posibilidad de guardar un registro de fallos y las circunstancias en las que se presentó (modo 2), cada fallo tiene un código asignado. Para poder acceder a los registros y otra funciones del OBD-II, el mecánico debe contar con un dispositivo de diagnóstico automotriz que envíe comandos al sistema OBD-II denominados PID's, los que permiten obtener una respuesta en base a lo que se desea analizar, este tipo de dispositivos son denominados scanners automotrices.

Figura 2.1: Conector OBD-II.

Conector OBD-II de un vehículo. Fuente: (ELM327, 2017).

La ubicación del conector OBD-II, generalmente se ubica debajo de la consola central del automóvil. Actualmente las formas en las que se puede establecer una conexión entre la plataforma OBD-II y el dispositivo de diagnosis, puede ser mediante Bluetooth, WiFi y USB, por lo que, establecer una conexión a través de conectores como el puerto en serie RS232 va disminuyendo progresivamente. Dicha conexión entre el sistema OBD-II y el dispositivo de diagnosis, debe ser visualizada mediante un software, que se ejecuta desde una PC o un dispositivo móvil, permitiendo la monitoreo en tiempos reales de diversos

parámetros que poseen las diferentes ECU's del automóvil, además de los códigos de falla (DTC), si es que el vehículo tuviese algún problema.

Cuando un individuo que no es un especialista en el área automotriz quiere monitorizar a su vehículo sin recurrir a un taller automotriz o gastar en los costosos dispositivos de monitoreo profesionales, entre las opciones económicas más populares en el mercado, el controlador ELM327 es el más extendido para establecer el enlace necesario para monitorear las diferentes ECU's del automóvil.

2.1. DISTRIBUCIÓN DE PINES

Existen dos tipos de conectores OBD-II presentes en los distintos tipos de vehículos, que indistintamente de su funcionalidad y/o tipo de vehículo para el que fue diseñado, cuenta con la siguiente distribución de pines:

Figura 2.2: Tipos de puertos OBD-II.

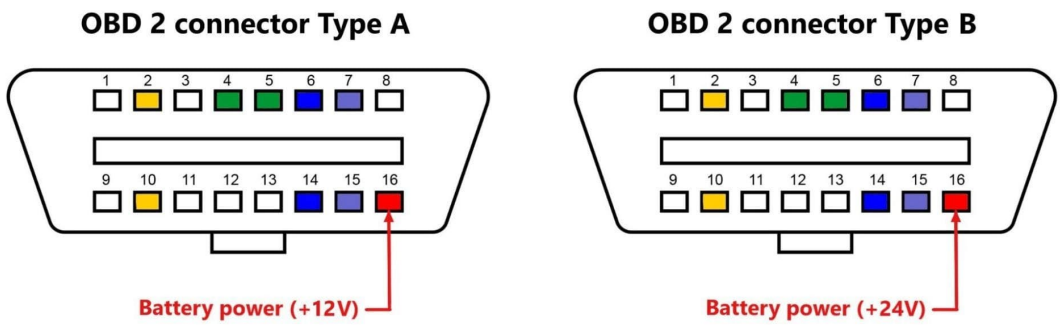


Diagrama de los puertos A y B OBD-II, Fuente: (Weis, 2023).

Indistintamente del tipo de OBD-II, la distribución de pines se da de la siguiente manera:

Tabla 2.1: Distribución de pines del DLC OBD-II.

N° de pin	Nombre del Pin	Descripción
1	Discrecional del fabricante	Este pin no es estándar y depende del fabricante del vehículo. No es indispensable para la comunicación normal.
2	Auto SAE J1850+	
3	Discrecional del fabricante	Este pin no es estándar y depende del fabricante del vehículo. No es indispensable para la comunicación normal.
4	Tierra del chasis	La tierra del Sistema del automóvil (incluido el chasis).
5	Tierra de señal	La tierra del Sistema del automóvil (incluido el chasis).
6	CAN Alta ISO 15765-4, SAE J2284	PIN CAN alto. Utiliza un protocolo CAN de 2 hilos a una velocidad de 1 Mbps.
7	ISO 9141-2 / ISO 14230 – 4 Linea K	Alfiler de línea K. Sigue un protocolo de comunicación serie asíncrono.
8	Discrecional del fabricante	Este pin no es estándar y depende del fabricante del vehículo. No es indispensable para la comunicación normal.
9	Discrecional del fabricante	Este pin no es estándar y depende del fabricante del vehículo. No es indispensable para la comunicación normal.
10	SAE J1850 Bus	
11	Discrecional del fabricante	Este pin no es estándar y depende del fabricante del vehículo. No es indispensable para la comunicación normal.
12	Discrecional del fabricante	Este pin no es estándar y depende del fabricante del vehículo. No es indispensable para la comunicación normal.
13	Discrecional del fabricante	Este pin no es estándar y depende del fabricante del vehículo. No es indispensable para la comunicación normal.
14	CAN Baja ISO 15765-4, SAE J2284	PIN CAN bajo. Utiliza un protocolo CAN de 2 hilos a una velocidad de 1 Mbps.
15	ISO 9141-2 / ISO 14230 – 4 Línea K (opcional)	PIN DE LINEA K. Utiliza el protocolo de comunicación serie asincrónico.
16	Energía de la batería del vehículo	Se conecta a la batería del vehículo para alimentar las herramientas del escaneo. Tipo "A" 12v/4ª, Tipo "B" 24v/2ª

Describe la función de cada pin que posee el OBD-II, Fuente: (Weis, 2023).

2.2. MODOS DE DIAGNÓSTICO OBD-II

De acuerdo con (SAE, 2002), el OBD-II (On-Board Diagnostics) tiene nueve modos de diagnóstico estandarizados, cada uno de los cuales brinda información específica sobre el estado y los componentes del vehículo. Estos modos de diagnóstico son:

- **Modo 1. Identificación de Parámetro (PID, del inglés Process Identification):** Permite acceder a los datos en tiempo real de los valores analógicos o digitales que emiten las diferentes ECU's del vehículo.
- **Modo 2. Acceso a Cuadro de Datos Congelados (del inglés Freeze Frame Data):** Permite a la ECU tomar una muestra referente a los valores actuales con las condiciones presentes en cuanto se detectó una falla.
- **Modo 3:** Esta modalidad permite la extracción todos los códigos de fallo (DTC) almacenados en la memoria de la ECU.

- **Modo 4:** Permite eliminar los códigos de diagnóstico de fallas guardados en la memoria del vehículo y reiniciar los sistemas de monitoreo.
- **Modo 5:** Proporciona información acerca de los resultados de las pruebas de monitoreo y verificación para los componentes y sistemas del vehículo.
- **Modo 6:** Permite acceder a los resultados de las pruebas de componentes específicos realizadas por los sistemas de control del automotor.
- **Modo 7:** Lee todos los DTC's pendientes que se encuentran en la memoria de la ECU.
- **Modo 8:** Con esta función, el técnico puede habilitar y deshabilitar actuadores.
- **Modo 9:** Presenta los datos referentes al número de serie del motor o VIN por sus siglas en inglés.

El estándar J1979 enlistados con anterioridad, generalmente son los más ocupados en la industria automotriz comercial. Sin embargo, la SAE J2190 define otros modos del 10 hacia arriba, los que cuentan con funcionalidades más avanzadas que usualmente no son accesibles desde los dispositivos de diagnóstico tradicionales, debido a que permiten realizar configuraciones avanzadas como la modificación de varios parámetros de la ECU, para fines que como, por ejemplo: aumentan el rendimiento del motor, sin considerar el incremento de las emisiones contaminantes que estos cambios producen, lo cual debido a las normativas establecidas por los organismos que regulan las emisiones contaminantes, pueden ser ilegales para ciertas legislaciones.

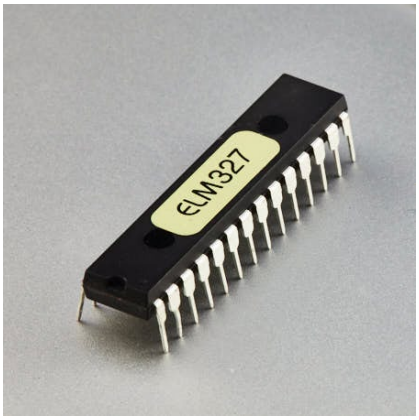
2.3. CIRCUITOS INTEGRADOS DE INTERCCIÓN OBD

Los circuitos integrados de interpretación OBD (On-Board Diagnostics) son dispositivos electrónicos diseñados específicamente para interpretar y procesar los datos transmitidos a través del puerto OBD de un vehículo. Estos circuitos integrados se utilizan en herramientas de diagnóstico automotriz, escáneres y otros dispositivos de lectura de códigos de falla.

El propósito principal de los circuitos integrados de interpretación OBD es comunicarse con la computadora del vehículo (ECU) y extraer datos sobre el estado y los componentes del vehículo. Esto incluye leer y borrar códigos de falla, acceso a lectura de datos en vivo, diagnósticos, y visualizar la información en una interfaz de usuario.

2.3.1. ELM327

Figura 2.3: Chip ELM327.



Fotografía del chip ELM327, Fuente: (Pinout, 2002).

El ELM327 es un circuito integrado (IC) utilizado en dispositivos de diagnóstico de vehículos compatibles con OBD-II (On-Board Diagnostics). Es especialmente conocido por su uso en adaptadores de diagnóstico OBD-II que permiten a los usuarios eliminar y leer los fallos del vehículo relacionados a códigos, monitorear los parámetros en tiempo real y el acceso a la información de diagnóstico, su conexión es a través del puerto OBD-II del vehículo, que se encuentra en la mayoría de los automóviles fabricados después de 1996. Actúa como un intermediario entre el vehículo y un dispositivo externo, permitiendo la comunicación y lectura de datos mediante software especializado, ya sean aplicaciones o programas desarrollas para softwares como Windows, MacOS o Linux.

ELM327 es compatible con varios protocolos de comunicación utilizados en OBD-II, como CAN (Controller Area Network), ISO9141, J1850 PWM (Pulse Width Modulation) y J1850 VPW (Variable Pulse Width). Esto permite que los dispositivos equipados con el ELM327 se conecten y comuniquen con una amplia gama de vehículos. Ofrece una interfaz de comunicación serial, generalmente por USB, Bluetooth o WI-FI, que permite la conexión con dispositivos externos.

De acuerdo con (ELMElectronics, 2023), los protocolos soportados por el circuito integrado ELM327 son:

- SAE J1850 PWM (41.6 kbaud)
- SAE J1850 VPW (10.4 kbaud)

- ISO 9141-2 (5 baud init, 10.4 kbaud)
- ISO 14230-4 KWP (5 baud init, 10.4 kbaud)
- ISO 14230-4 KWP (fast init, 10.4 kbaud)
- ISO 15765-4 CAN (11 bit ID, 500 kbaud)
- ISO 15765-4 CAN (29 bit ID, 500 kbaud
- ISO 15765-4 CAN (11 bit ID, 250 kbaud)
- ISO 15765-4 CAN (29 bit ID, 250 kbaud)
- SAE J1939 (250 kbaud)
- SAE J1939 (500 kbaud)

El ELM327 es ocupado en diversos dispositivos y aplicaciones relacionadas con el diagnóstico automotriz, para objeto de este estudio principalmente se puede desatacar su aplicación en dos casos:

- **Scanners automotrices profesionales:** Muchos escáneres automotrices profesionales, utilizan el ELM327 como el chip principal para la comunicación e interpretación de la interfaz OBD-II de los vehículos.

Figura 2.4: Scanner Automotriz Autel.

Scanner automotriz profesional Autel MaxiSYS MS906BT. Fuente: (AUTEL, 2022).

- **Adaptadores OBD-II económicos:** Hay una amplia gama de adaptadores OBD-II en el mercado que utilizan el ELM327, que cuenta con la característica de ser económicos y fáciles de usar; para estos casos la comunicación con los vehículos se da a través de diversos dispositivos, como teléfonos inteligentes, tabletas, computadoras, etc.

Figura 2.5: ELM 327.

ELM327 de la marca KONNWEI modelo KW902. Fuente: (KONNWEI, 2020).

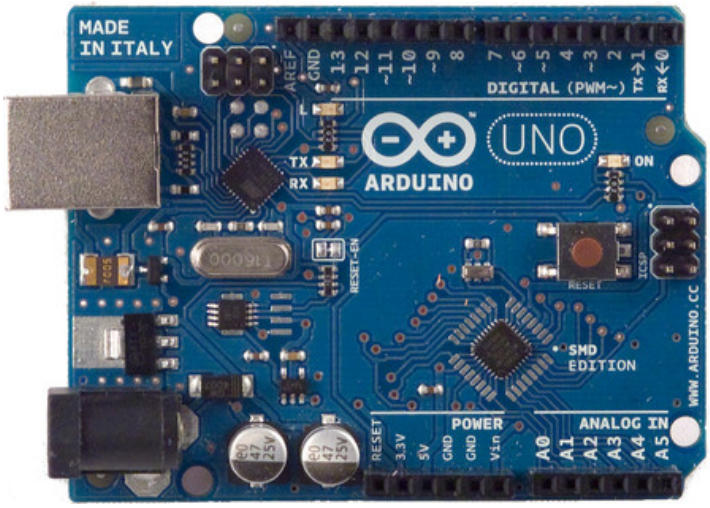
Se opto por el uso de un dispositivo ELM327 para la comunicación entre la tarjeta mobydic 4910 y Arduino, debido a la facilidad para su adquisición dentro del mercado ecuatoriano, además de su bajo costo. En específico, el ELM327 seleccionado para este proyecto, es mostrado en la figura 2.5, que pertenece a la marca KONNWEI modelo KW902.

CAPÍTULO III

3. ARDUINO

Arduino es una herramienta sumamente útil por su plataforma de desarrollo, dentro de la electrónica y la programación al ser una solución accesible y sencilla para la realización de proyectos de dispositivos interactivos; el enfoque en la facilidad de uso y su sencilles de programación ha sido un punto crucial para ser elegido el dispositivo para la recolección de datos dentro del proyecto de la mobydic 4910, ya que permite el uso de un mismo ecosistema de hardware y software.

Figura 3.1: Placa Arduino Uno.



Placa que permite la programación con un lenguaje basado en C++. Fuente: (Enríquez Herrador, 2009).

En términos de hardware, utiliza placas electrónicas equipadas con microcontroladores programables que actúan como el núcleo de los proyectos. La simplicidad en el diseño del hardware de Arduino facilita la implementación de circuitos electrónicos y reduce las barreras para aquellos sin experiencia previa en electrónica.

El software de Arduino es una plataforma de desarrollo integrada (IDE) tiene un entorno de programación sencillo y útil. Apoyado en C/C++, el IDE de Arduino simplifica el proceso de codificación al proporcionar una biblioteca extensa que simplifica la interacción con los componentes electrónicos conectados a la placa. Esta característica permite a los usuarios programar sus proyectos sin requerir conocimientos avanzados en programación, lo que amplía aún más el acceso y la participación en el campo de la electrónica.

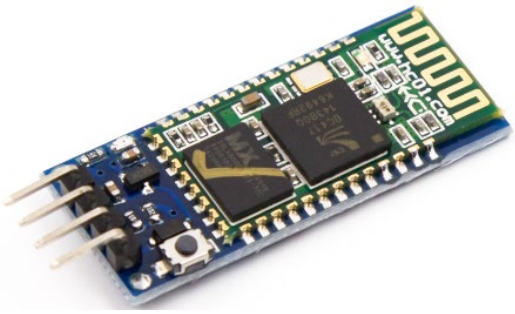
Las características de Arduino son variadas, ya que cuentan con una gran variedad de placas, tales como Arduino Uno, mega, nano, entre otros. Estos tipos de Arduino son compatibles con la programación que se ocupara para la comunicación de la tarjeta mobydic 4910 mediante el módulo bluetooth que a su vez se comunica con la tarjeta ELM327. (Zakaria et al., 2023).

- Pines de E/S: Las placas Arduino tienen pines de entrada/salida digital y análogas. Los pines digitales pueden ser ocupados como entradas o salidas binarias, mientras que los análogos pueden leer y generar señales análogas.
- Posee microcontroladores AVR, son programables y tienen capacidades de entrada/ salida digital y analógica para que se pueda interactiva con componentes electrónicos externos.
- Bibliotecas y ejemplos: Arduino se apoya con librerías de funciones predefinidas que facilitan la interacción con los componentes electrónicos comunes. Además, el IDE incluye ejemplos de código que muestran cómo utilizar diferentes funciones y módulos, lo que acelera el proceso de desarrollo.

3.1. MÓDULO BLUETOOTH HC-05

El módulo Bluetooth que es compatible con el lenguaje Arduino, se define como un dispositivo de comunicación inalámbrico que permite la trasferencia de datos entre un microcontrolador Arduino y otros dispositivos compatibles con Bluetooth, como teléfonos móviles, tabletas y computadoras. Este módulo utiliza la tecnología Bluetooth para establecer una conexión de corto alcance y enviar y recibir información de manera inalámbrica.

Figura 3.2: Módulo bluetooth HC-05.



Módulo que permite la transferencia de datos de manera inalámbrica. Fuente: (Enríquez Herrador, 2009).

Una vez que se establece la conexión Bluetooth, dicho módulo admite la transferencia de data en ambos sentidos. Lo que indica que a través del módulo Bluetooth se puede enviar datos a otros dispositivos, como comandos de control o información de sensores. Al mismo tiempo, el módulo Bluetooth puede recibir datos de otros dispositivos y transmitirlos al microcontrolador Arduino para su procesamiento o visualización, según (FÉLIX ANDRÉS ANDRADE SÁNCHEZ, 2015).

- El módulo HC-05 tiene una interfaz serial UART que permite la comunicación con el Arduino por medio de los pines RX (receptor) y TX (transmisor). Esto permite que el Arduino envíe comandos y datos al módulo HC-05 para su transmisión inalámbrica y reciba datos del módulo HC-05 desde dispositivos emparejados tales como el ELM327.

- El HC-05 admite una velocidad de transmisión ajustable, lo que permite una transferencia de datos eficiente según las necesidades del proyecto.

- Para utilizar el módulo HC-05 con Arduino, es necesario establecer una conexión física entre los pines RX y TX del módulo y los pines correspondientes del Arduino, además se debe configurar adecuadamente la comunicación serial en el código a utilizar.

- "El módulo Bluetooth de Arduino también puede admitir diferentes perfiles de Bluetooth, como el perfil SPP (Serial Port Profile), que emula una conexión serial

virtual para facilitar la comunicación con otros dispositivos"(Cavazos Luis Enrique et al., 2022).

Tabla 3.1: Tabla de descripción del módulo HC-05.

Funcionalidad	Descripción
Trasmisión de datos	Facilita la transferencia de datos entre dispositivos a través de la conexión Bluetooth.
Conexión serial	Proporciona una interfaz serial, enviando y recibiendo datos utilizando protocolos comunicación como UART.
Configuración sencilla	Permite una configuración fácil y rápida a través de comandos AT, lo que simplifica el proceso de conexión y ajuste de parámetros.

Describe las funciones que posee el módulo bluetooth que se ocupa en el proyecto. Fuente: Autores.

3.2.CONFIGURACIÓN ESCLAVO Y MAESTRO

El dispositivo configurado como "esclavo" en una conexión Bluetooth actúa como un receptor pasivo, a la orden de solicitudes de conexión que otros dispositivos lo demanden. El esclavo responde a la iniciativa tomada por un dispositivo configurado como "maestro" y establece la comunicación según las instrucciones del maestro. El esclavo cumple principalmente la función de recibir y ejecutar comandos (Álvaro Alberto Giner & Armesto Ángel, n.d.).

Por otro lado, el dispositivo configurado como "maestro" en una conexión Bluetooth toma la iniciativa de establecer la conexión y asume un rol más activo en la comunicación, el cual es parte fundamental para el desarrollo de la comunicación entre los módulos HC-05 y el ELM327. El maestro busca el dispositivo esclavo el cual será el ELM327. Una vez establecida la conexión, el maestro puede enviar y recibir datos al dispositivo esclavo, controlando así la secuencia y el flujo de la comunicación con la tarjeta mobydic 4910.

La funcionalidad de esclavo y maestro permite una comunicación bidireccional y establece una jerarquía en la relación entre dispositivos.

A continuación, se presentará una tabla con las características del modo esclavo y modo maestro de Arduino, según (García-Tuleda & Marín,2023).

Tabla 3.2: Tabla esclavo y maestro.

Características	Modo Esclavo	Modo Maestro
Funcionalidad	Receptor pasivo	Iniciador activo
Conexión	Responde a solicitudes de conexión	Inicia la conexión
Rol en la comunicación	Principalmente recibe y ejecuta comandos	Controla la secuencia y el flujo de la comunicación
Configuración	Espera la iniciativa del maestro para establecer la conexión	Busca y se conecta a dispositivos esclavos
Iniciativa	Responde a la iniciativa del maestro	Toma la iniciativa de establecer la conexión
Comandos	Recibe comandos del maestro para su ejecución	Envía comandos al esclavo para controlarlo
Transferencia de datos	Bidireccional, recibe y envía datos según las instrucciones del maestro	Bidireccional, envía y recibe datos del esclavo
Control de flujo	Controlado por el maestro	Controla el flujo de datos hacia el esclavo
Configuración del módulo	Puede ser configurado como esclavo	Puede ser configurado como maestro

Describe las características del modo esclavo y maestro de Arduino. Fuente: Autores.

3.3. COMANDOS AT

Son un tipo de comandos básicos utilizados para permitir la comunicación con el dispositivo bluetooth utilizado. Al enviar este comando, el módulo debe responder con un mensaje de confirmación, lo que indica una conexión exitosa. Los comandos AT (Comandos de control AT) configuran y controlan módulos de comunicación, como los módulos Bluetooth y los módulos GSM, mediante una interfaz UART (Universal Asynchronous Receiver-Transmitter) en Arduino. Estos comandos permiten establecer parámetros de comunicación, configurar opciones de funcionamiento y realizar diversas operaciones. En la tabla 3, se presentan algunos comandos AT comunes utilizados en Arduino (ELMElectronics, 2023).

Tabla 3.3: Tabla de comandos AT.

Comando AT	Descripción
AT	Comando básico para verificar la comunicación con el dispositivo.
AT+BAUD	Configura la velocidad de transmisión (baud rate) del módulo.
AT+NAME	Establece el nombre del dispositivo en los módulos Bluetooth.
AT+PIN	Establece el código PIN para emparejamiento en los módulos Bluetooth.
AT+ROLE	Configura el rol del dispositivo en los módulos Bluetooth (Maestro o Esclavo).
AT+RESET	Reinicia el módulo a su estado predeterminado.
AT+CMGF	Establece el modo de mensaje para los módulos GSM (texto o PDU).
AT+CNUM	Obtiene el número de teléfono asociado al módulo GSM.
AT+CSQ	Obtiene la calidad de señal del módulo GSM.
AT+CMGS	Envía un mensaje de texto a un número de teléfono específico en los módulos GSM.

Comando AT	Descripción
AT+BIND	Comando específico utilizado en módulos Bluetooth para emparejar dispositivos Bluetooth.

Describe y explica los comandos AT para Arduino. Fuente: Autores.

El comando AT+BIND juega parte fundamental para el desarrollo del código en Arduino, ya que el código; AT+BIND=<dispositivo_remoto> representa la dirección MAC (identificador único del dispositivo bluetooth) el cual se enlaza al módulo ELM327.

3.4. LIBRERÍAS UTILIZADAS

Las librerías de Arduino proporcionan funciones que ayudan a simplificar el desarrollo de aplicaciones. En el presente proyecto se utilizan dos librerías modificadas acorde a las necesidades de comunicación que se requiere para que se pueda interpretar los datos de los actuadores (potenciómetros) de la tarjeta mobydic 4910.

3.4.1. SoftwareSerial.h

La librería SoftwareSerial.h es una librería estándar de Arduino que permite la comunicación serial para pines digitales (por ejemplo, RX y TX en Arduino Uno). Proporciona una forma sencilla de configurar y utilizar puertos de comunicación serial adicionales en el Arduino.

La librería SoftwareSerial.h es útil cuando se necesita establecer una comunicación serial con dispositivos externos que utilizan otros pines digitales diferentes a los pines RX y TX predeterminados. Permite crear múltiples instancias de objetos de comunicación serial en diferentes pines digitales, lo que amplía las capacidades de comunicación del Arduino.

Algunas características y funcionalidades clave de la librería SoftwareSerial.h son las siguientes:

- **Comunicación serial en pines digitales:** La librería permite la comunicación serial de datos en pines digitales seleccionados, lo que ofrece flexibilidad en la elección de los pines de comunicación serial.
- **Configuración de baud rate:** Permite configurar el valor del baud rate para adaptarse a las necesidades del dispositivo externo con el que se desea comunicar.

- **Métodos de lectura y escritura:** Proporciona métodos para la lectura y escritura de datos a través de la comunicación serial, como read(), write(), print() y println().
- **Control de flujo:** La librería no admite el control de flujo por hardware (RTS/CTS), pero proporciona la opción de habilitar el control de flujo por software (XON/XOFF) que evita la pérdida de información en caso de comunicaciones rápidas.
- **Limitaciones de velocidad y precisión:** Debido a su implementación en software, la librería SoftwareSerial.h tiene algunas limitaciones en términos de velocidad y precisión en comparación con la comunicación serial hardware. Por lo tanto, es importante tener en cuenta las limitaciones de velocidad y no exceder los valores recomendados para una comunicación confiable.

La librería SoftwareSerial.h es ampliamente utilizada en proyectos de Arduino que requieren comunicación serial adicional en pines digitales. Es especialmente útil cuando se necesita interactuar con varios dispositivos que utilizan la comunicación serial, como módulos GPS, módulos Bluetooth, módems GSM, entre otros.

3.4.2. ELMduino.h

La librería ELMduino.h es una librería para Arduino que permite la comunicación con dispositivos ELM327 basados en el estándar OBD-II. Esta librería simplifica la comunicación entre el dispositivo Arduino seleccionado y el dispositivo ELM327, lo que facilita la lectura y escritura de datos del vehículo a través de la interfaz OBD-II.

La librería ELMduino.h proporciona una serie de funciones y métodos que permiten realizar operaciones comunes de diagnóstico automotriz, como leer códigos de falla, obtener valores de sensores, enviar comandos personalizados, entre otros. Está diseñada para trabajar con el módulo ELM327 y utiliza la comunicación serial para establecer la conexión con Arduino.

Algunas de los aspectos principales de esta librería incluyen:

- **Inicialización y configuración del dispositivo ELM327:** Permite establecer la velocidad de comunicación, el protocolo de comunicación y otras configuraciones necesarias para el funcionamiento del ELM327.
- **Lectura de códigos de falla:** Permite leer y decodificar los códigos de falla almacenados en la memoria de la ECU del vehículo.

- **Obtención de datos de sensores:** Permite leer los valores de los diversos sensores y actuadores del vehículo.
- **Envío de comandos personalizados:** Permite enviar comandos personalizados al vehículo a través del dispositivo ELM327 para obtener información específica o realizar acciones específicas.

La librería ELMduino.h simplifica la programación del Arduino al proporcionar funciones y métodos específicos para la comunicación con dispositivos ELM327. Esto facilita el desarrollo de aplicaciones de diagnóstico automotriz utilizando Arduino y permite acceder a los datos del vehículo de manera rápida y sencilla.

3.5. PIDs A INTERPRETAR

Los PIDs (Parameter Identification) algoritmos de control utilizados para mantener o ajustar automáticamente una variable o proceso en un sistema, para los sensores automotrices en cuestión se relaciona con códigos o identificadores utilizados en el protocolo de comunicación OBD-II (On-Board Diagnostics) para acceder a los datos específicos de los sensores y sistemas de la ECU's que se estan manejando.

Como se especificó en el apartado 1.2.4. la tarjeta mobydic 4910 cuenta con tres funcionalidades:

- Modo diésel
- Modo gasolina
- Modo de generación de fallos (DTC)

Para obtener los datos correspondientes a cada potenciómetro en las modalidades diésel y gasolina, se necesitan establecer los modos de diagnóstico OB2-II y determinar los PIDs correspondientes a cada parámetro simulado, además, puesto que los datos a recolectar están en formato hexadecimal, es necesario disponer de las fórmulas para convertir los datos a formato decimal, en base a (Resource, 2008), se desarrollaron las siguientes tablas:

Tabla 3.4: Tabla PID's sensores modalidad diésel.

Modalidad diésel				
Sensor/Actuador	**Modo OBD-II**	**PID (hex)**	**PID (Dec)**	**Fórmula**

BAT	1	42	66	$\frac{256A+B}{1000}$
MAP	1	0B	11	A
IAT	1	0F	15	$\mathrm{A}-40$
AAT	1	46	70	$\mathrm{A}-40$
FRP	1	23	35	$0.079(256A+B)$
EGRP	1	2C	44	$\frac{100}{255}\mathrm{A}$
EGRE	1	2D	45	$\frac{100}{128}\mathrm{A}-100$
BARO	1	33	51	A

Se indican los PID's de la modalidad diésel en la tarjeta mobydic 4910. Fuente: Autores.

Tabla 3.5: Tabla PID's sensores modalidad gasolina.

Modalidad gasolina				
RPM	1	0C	12	$\frac{256A+B}{4}$
ECT	1	05	5	$\mathrm{A}-40$
MAF	1	10	16	$\frac{256A+B}{100}$
APP	1	11	17	$\frac{100}{255}A$
O2V	1	14	20	$\frac{100}{128}B-100$
LOAD	1	04	4	$\frac{100}{255}A$
VSS	1	0D	13	A
FLI	1	2F	47	$\frac{100}{255}A$

Se indican los PID's de la modalidad gasolina en la tarjeta mobydic 4910. Fuente: Autores.

Respecto a la Modalidad generación de fallos (DTC), únicamente es necesario acceder al modo 3 de diagnóstico OB2-II:

Tabla 3.6: Tabla de DTC's.

Modalidad generación de fallos (DTC)	
Modo	**Errores**
3	P0100
3	P0101
3	P0102
3	B0200
3	B0201
3	C0300
3	U0400

Enlista los fallos que se generan a través de la tarjeta mobydic 4910. Fuente: Autores.

CAPÍTULO IV

4. DISEÑO DEL PROYECTO

4.1. DIAGRAMA DE CONEXIÓN Y GENERALIDADES

Durante el desarrollo del dashboard se presentaron situaciones complicadas debido a que, la tarjeta de Ozen Elektronik presentaba un complejo bloqueo de trasferencia de datos, esto debido a condiciones particulares preestablecidas por el fabricante, lo cual no permitía establecer una conexión sencilla a través de los elementos que se tenían previstos utilizar en primera instancia. Por lo cual, al final, los componentes elegidos para la realización del proyecto se presentan a continuación mediante un diagrama simplificado del circuito:

Figura 4.1: Diagrama de conexión entre los dispositivos utilizados para la realización de la interfaz.

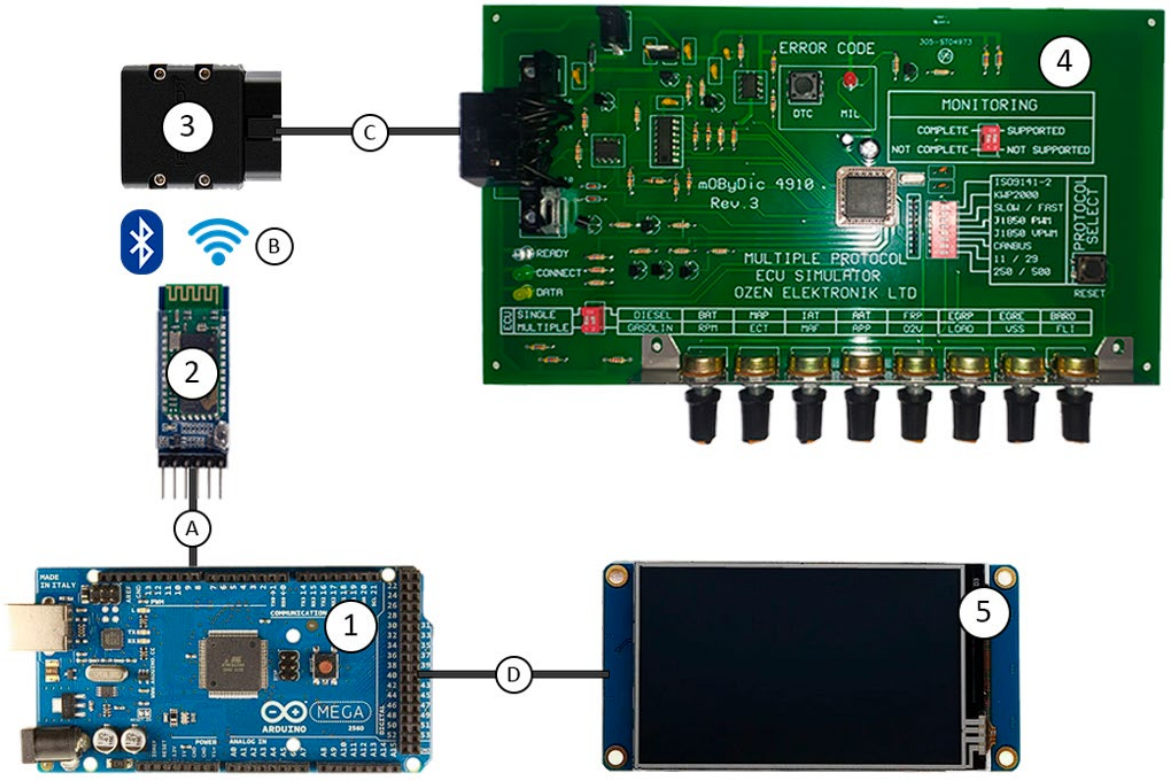


Diagrama que enumera todos los elementos utilizados en el proyecto. Fuente: Autores.

Los elementos que conforman el diagrama de la figura 4.1 son los siguientes:

1. Arduino MEGA

2. Módulo Bluetooth HC-05
3. KONNWEI KW902 (ELM327)
4. Tarjeta Multiple Protocol ECU Simulator de Ozen Elektronik mOByDic4910
5. Nextion Display 3.5″

Las conexiones entre los elementos se dan de la siguiente manera:

1. **Conexión A:** Se da entre el Arduino MEGA y el módulo bluetooth HC-05, a través de los pines 10 y 11.
2. **Conexión B:** Se da entre el módulo Bluetooth HC-05 y el dispositivo ELM327 KONNWEI KW902, a través de comunicación Bluetooth.
3. **Conexión C:** Se da entre el dispositivo ELM327 KONNWEI KW902 y la tarjeta Multiple Protocol ECU Simulator de Ozen Elektronik mOByDic4910, mediante el puerto OBD-II.
4. **Conexión D:** Se da entre el Arduino MEGA y la pantalla Nextion de 3.5”, mediante los pines 4 y 5.

4.2. METODOLOGÍA

El proceso metodológico para el desarrollo del proyecto se realizó a través del respectivo trabajo de investigación y el desarrollo de nuevas alternativas destinadas a solucionar los problemas que se presentaron durante la elaboración del anteproyecto, por lo cual los elementos utilizados y la secuencia de pasos que se realizaron para el desarrollo del dashboard digital se presentan en los siguientes subapartados.

4.2.1. RECOLECCIÓN DE DATOS DE LA TARJETA MOBYDIC 4910 A TRAVÉS DEL MÓDULO KONNWEI KW902 (ELM327)

Figura 4.2: Diagrama de conexión de la tarjeta mobydic 4910 y el ELM327.

Muestra la conexión entre la tarjeta mobydic y el módulo ELM327 a través de puerto CAN bus. Fuente: Autores.

- El dispositivo ELM327 se encarga de extraer la información del simulador de ECU conectándose al puerto OBD-II de la tarjeta o en su defecto del vehículo del que se desee extraer los datos.
- Una vez conectado al puerto OBD-II, el adaptador se alimenta directamente de la energización de la tarjeta.
- El KONNWEI KW902 se comunica de forma inalámbrica con dispositivos compatibles, a través de la tecnología Bluetooth. Para utilizarlo, es necesario emparejar el adaptador KW902 con el módulo Bluetooth deseado, que en el caso del proyecto es el dispositivo HC-05.
- A través de la respectiva programación del Arduino, se debe establecer la conexión con el dispositivo KONNWEI KW902.
- Una vez que el adaptador KW902 está conectado y la aplicación de escaneo está funcionando, se pueden realizar varias funciones de diagnóstico. Estas funciones incluyen la lectura de códigos de diagnóstico que emite el simulador de ECU, además de la extracción de datos de parámetros en tiempo real que varían al manipular los potenciómetros que simulan el funcionamiento, de los sensores que existen en los vehículos tanto para diésel como para gasolina.

4.2.2. CONFIGURACIÓN DEL MÓDULO BLUETOOTH HC-05

Figura 4.3: Diagrama de conexión entre los módulos bluetooth.

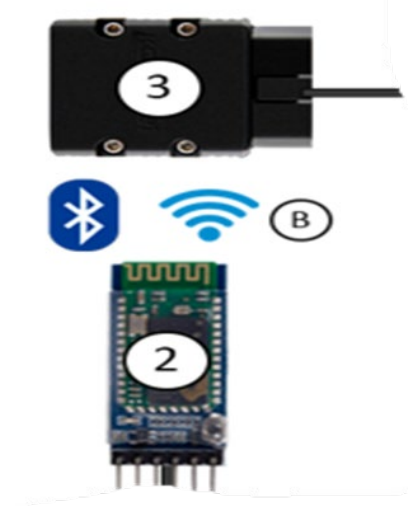
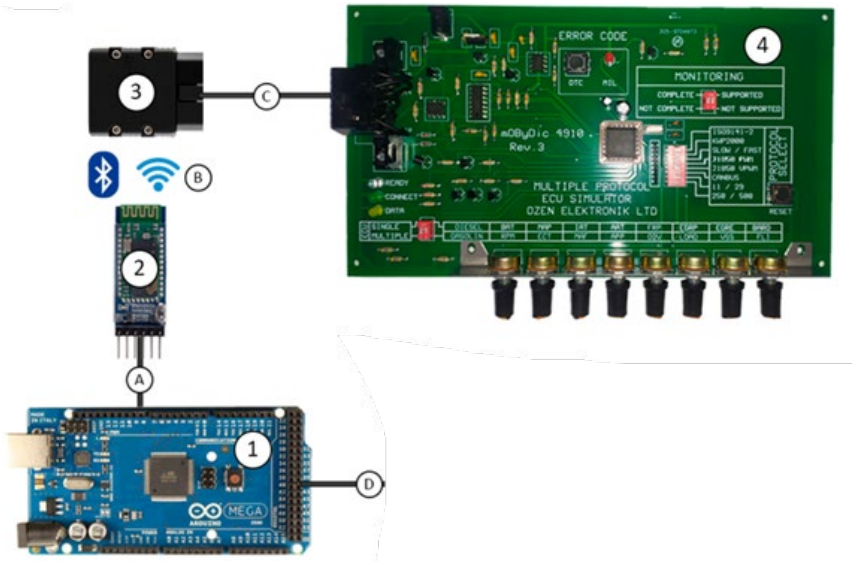


Diagrama que muestra la conexión entre el módulo HC-05 y el ELM327. Fuente: Autores.

- La configuración del módulo bluetooth necesita de la conexión al pin KEY del módulo a tierra (GND) mediante un cable o pulsador durante el proceso de encendido o reinicio.
- La configuración y conexión serial se la realiza mediante comandos AT, los cuales deben ser conectados a los pines de comunicación serial de Arduino (RX y TX), en su defecto se utiliza los pines 10 y 11 de Arduino, en conjunto con el trabajo de la librería SoftwareSerialh.
- Posterior a lo antes ya descrito se necesita cargar el código al Arduino para que los comandos AT puedan ser enviados al módulo HC-05. Para enviar los comandos AT al módulo HC-05 se tiene que configurar como maestro mediante el uso de: AT+ROLE=1\r\n // Establecer el módulo como maestro AT+RESET\r\n // Reiniciar el módulo después de la configuración.
- Una vez establecido el bluetooth maestro y enviados los comandos AT se procede a anclar los dispositivos ELM327 y el módulo maestro (HC-05).

4.2.3. CONEXIÓN DE LA PLACA ARDUINO CON LOS COMPONENTES ELM 327, MODULO BLUETOOTH Y TARJETA MOBYDIC 4910

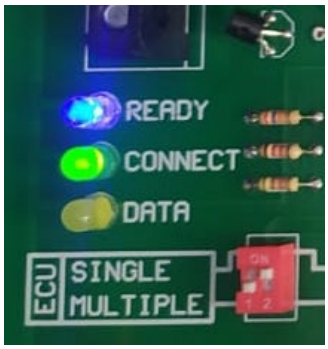
Figura 4.4: Diagrama de conexión de la tarjeta mobydic 4910 y el módulo.



Muestra la conexión de Arduino con el módulo bluetooth que a su vez se en lanza al chip ELM327. Fuente: Autores.

- Con los elementos conectados, se procede a emparejar el módulo bluetooth con la placa Arduino para que este empiece a reconocer los datos de la ECU programable que son extraídos a través del ELM327. Para reconocer si se estableció la conexión entre Arduino y la tarjeta mobydic 4910, el led CONNET deberá activarse.

Figura 4.5: Activación del led CONNECT.



Al establecerse la conexión el CONNECT se activa. Fuente: Autores.

- El código con los comandos para el bluetooth maestro, comandado por AT y dirigido por dos librerías adaptadas a las necesidades de la lectura de los datos de la ECU se carga dentro del entorno de desarrollo de Arduino (IDE), posteriormente se comienza incluyendo las librerías necesarias para la realización del programa:

Figura 4.6: Inclusión librerías.

```
#include <SoftwareSerial.h>
#include "ELMduino.h"
```

Comando de código para incluir: SoftwareSerial.h y ELMduino.h. Fuente: Autores.

- Se debe declarar los pines mediante los cuales se está realizando la conexión, tanto para el módulo bluetooth HC-05, como para la pantalla Nextion, además se define el ELM_PORT.

Figura 4.7: Declaración de los pines de conexión.

```
SoftwareSerial mySerial(10, 11); // RX, TX
SoftwareSerial myNextion(4, 5); // RX, TX

#define ELM_PORT mySerial
```

Líneas de comandos para establecer los pines de conexión. Fuente: Autores.

- Es necesario configurar la velocidad de comunicación para el puerto serial "Serial.begin()" y para la conexión bluetooth del ELM327 "ELM_PORT.begin()", que para los dos casos debe ser de 38400 baudios.

Figura 4.8: Velocidades de conexión para los puertos.

```
Serial.begin(38400);
ELM_PORT.begin(38400);
```

Líneas de código de comando para la velocidad establecer la velocidad de conexión,

Fuente: Autores.

- Cuando el código se encuentra cargado se procede a realizar la obtención de datos de la tarjeta mobydic 4910, que, debido a la naturaleza de los PIDs, se presentan en formato hexadecimal para cada uno de los 16 sensores, y dado que el aspecto más relevante a tratar fue la obtención de los datos de la tarjeta, la explicación del código, principalmente se centrará en las líneas destinadas a la obtención, el procesamiento y la transformación de datos.

Figura 4.9: Ejemplo de código para las RPM's.

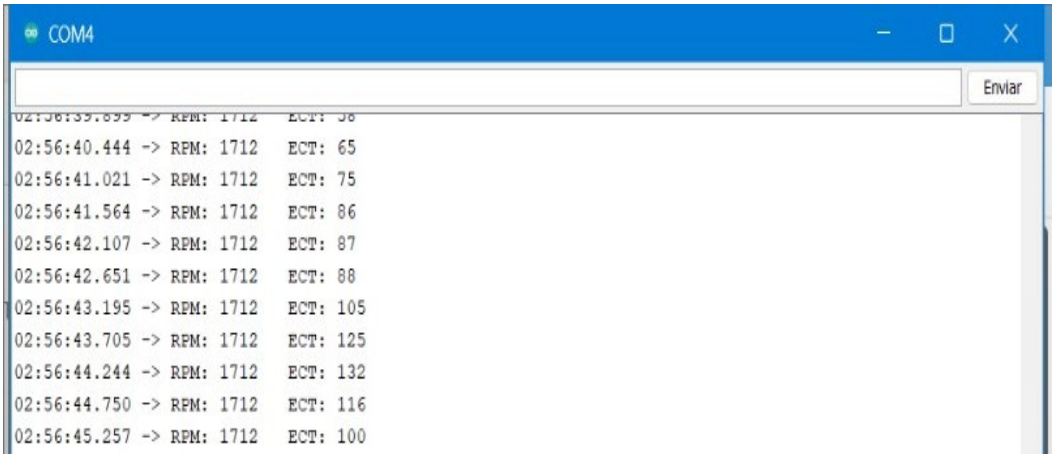
```
// RPM: servicio 1 y PID 12
if(myELM327.queryPID(1,12)){

  if(myELM327.status==ELM_SUCCESS){
    testrpm=myELM327.payload;
    charA[0]=myELM327.payload[4];
    charA[1]=myELM327.payload[5];
    charA[2]=0;

    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
    charB[2]=0;


    int A=(strtoul(charA,NULL,16));
    int B=(strtoul(charB,NULL,16));

    rpm=(64*A+0.25*B);
    //Serial.print("RPM: ");
    //Serial.print(rpm);
```

Ejemplo de líneas de código para la obtención y transformación del PID de RPM. Fuente: Autores.

- Con los PID's identificados, se procede a realizar la clasificación de los datos para convertirlos a formato decimal mediante las fórmulas descritas en las tablas 3.4 y 3.5, a continuación, se presentará la explicación detallada de lo que hace el fragmento del código de la figura 4.9, que consiste en un ejemplo de conversión para obtener la información sobre las revoluciones del motor del vehículo:
- if(myELM327.queryPID(1,12)): Esta línea envía una solicitud al dispositivo ELM327 para obtener el valor del PID de las RPM's, para lo cual se introduce, los dígitos (1, 12), que corresponden al modo 1 OBD-II y al PID 12 que representa a las RPMs en formato decimal. Si la solicitud es exitosa, continúa ejecutando el siguiente bloque de código.
- if(myELM327.status==ELM_SUCCESS): Verifica si la respuesta del dispositivo ELM327 indica que la solicitud fue exitosa. La constante ELM_SUCCESS representa un estado de éxito en la comunicación con el dispositivo.
- testrpm=myELM327.payload;: Guarda el valor de la carga útil (payload) recibida del dispositivo ELM327 en la variable testrpm. La carga útil contiene los datos de respuesta del PID solicitado.
- charA[0]=myELM327.payload[4]; ,charA[1]=myELM327.payload[5];, charA[2]=0;: Extrae los caracteres correspondientes a revoluciones del motor del vehículo de la carga útil recibida y los guarda en un arreglo de caracteres charA. Estos caracteres están ubicados en las posiciones 4 y 5 del arreglo payload.
- charB[0]=myELM327.payload[6];, charB[1]=myELM327.payload[7];, charB[2]=0;: Similar al paso anterior, extrae los caracteres adicionales de las RPM y los guarda en otro arreglo de caracteres charB. Estos caracteres están ubicados en las posiciones 6 y 7 del arreglo payload.
- int A=(strtoul(charA,NULL,16));, int B=(strtoul(charB,NULL,16));: Convierte los valores de los arreglos de caracteres charA y charB en enteros. Se utiliza la función strtoul para convertir los caracteres hexadecimales en enteros decimales. El argumento NULL se utiliza para ignorar cualquier prefijo de formato. El valor 16 se utiliza para indicar que los caracteres están en formato hexadecimal.

- rpm=(64*A+0.25*B);: Calcula las RPM del motor utilizando los valores convertidos de A y B. Se realiza una operación matemática donde se aplican las fórmulas descritas en la tabla 3.5.
- El procedimiento para la obtención de los demás PIDs es el mismo para todos los sensores, únicamente se tiene que introducir el valor PID correspondiente a cada sensor.
- Una vez que se realizó la conversión a formato decimal de todos los PIDs, estos valores pasan a ser representados en la pantalla serial de Arduino IDE; es necesario verificar que todos los sensores simulados por la tarjeta se encuentren dentro de los intervalos establecidos de funcionamiento para cada PID:

Figura 4.10: Valores de los PID's RPM y ECT.



Visualización en el serial monitor de los valores de PID's, en el serial monitor. Fuente: Autores.

Tabla 4.1: Tabla de valores mínimos y máximos para el modo diésel.

Diésel			
Sensor/Actuador	**Valor mínimo**	**Valor máximo**	**Unidades**
BAT	0	65.535	V
MAP	0	255	kPa

IAT	-40	215	°C
AAT	-40	215	°C
FRP	0	5177.265	kPa
EGRP	0	100	%
EGRE	-100	99.2	%
BARO	0	255	kPa

Describe el intervalo de funcionamiento de los sensores diésel en la tarjeta mobydic 4910.

Fuente: Autores.

Tabla 4.2: Tabla de valores mínimos y máximos para el modo gasolina.

Gasolina			
Sensor/Actuador	**Valor mínimo**	**Valor máximo**	**Unidades**
RPM	0	16383,75	rpm
ECT	-40	215	°C
MAF	0	655,35	g/s
APP	0	100	%

O2V	-100	99,2	%
LOAD	0	100	%
VSS	0	255	km/h
FLI	0	100	%

Describe el intervalo de funcionamiento de los sensores de gasolina en la tarjeta mobydic 4910. Fuente: Autores.

- Una vez comprobado que los datos obtenidos se encuentran en el rango de datos esperado, se procede a realizar la interfaz gráfica.

4.2.4. DISEÑO DE LA INTERFAZ PARA LA PANTALLA NEXTION ARDUINO

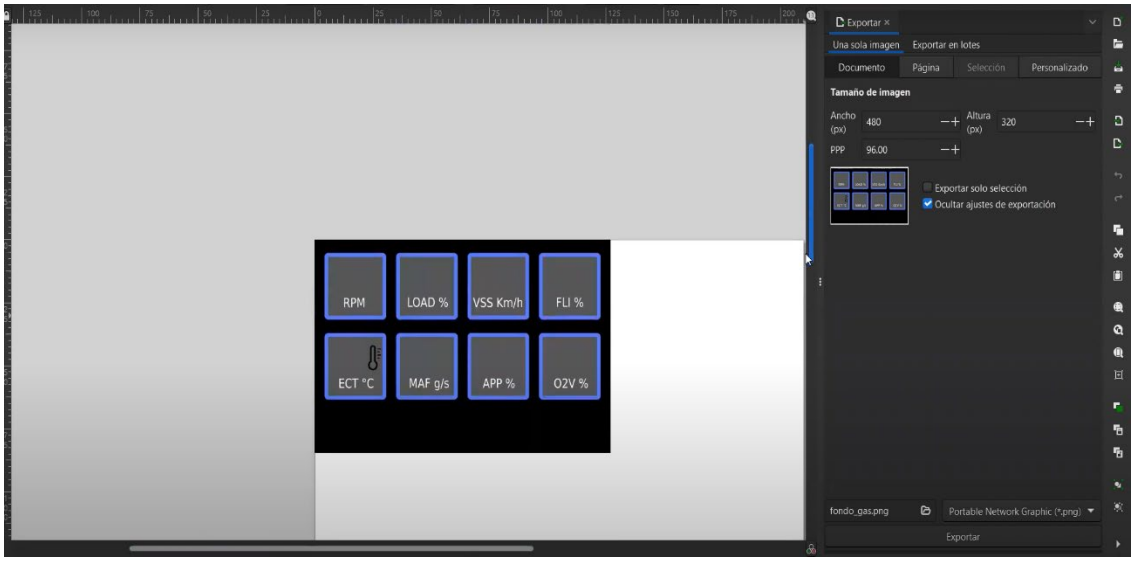
Figura 4.11: Pantalla Nextion de Arduino.

Pantalla Nextion TFT 3.5". Fuente: Autores.

- En el desarrollo de la interfaz gráfica como primer punto se realiza el fondo en la aplicación Inkscape, véase en la figura 4.12, se toma como referencia las dimensiones de la pantalla que se ocupara, las cuáles son; ancho 480 x altura 320 (px). Las medidas son importantes ya que al momento de importar los datos al programa NEXTION HMI las mismas deben coincidir para que se pueda presentar en la pantalla TFT. Se realizaron tres fondos destinados a usarse para el desarrollo

de las interfases diésel, gasolina y DTC's, como se puede observar en las figuras 4.13, 4.15 y 4.16.

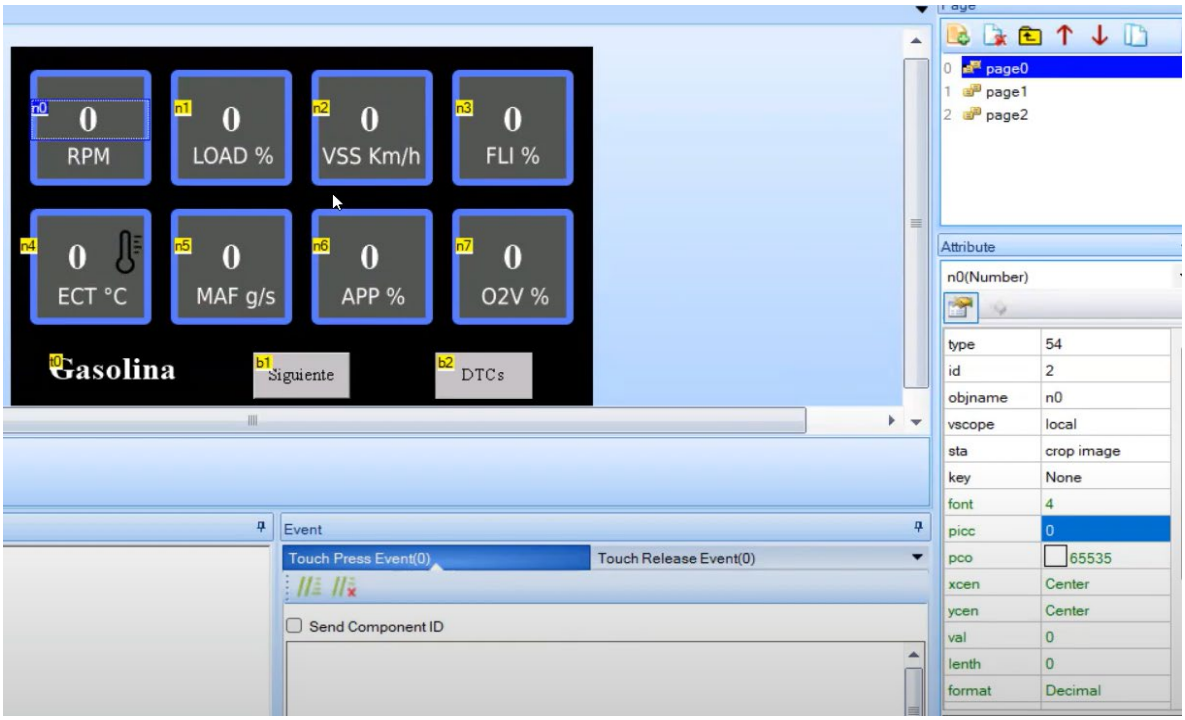
Figura 4.12: Modelo de interfaz para el dashboard.



Se muestra el modelo creado en el programa Inkcape que se utilizó para la presentación de los datos en la pantalla Nextion. Fuente: Autores.

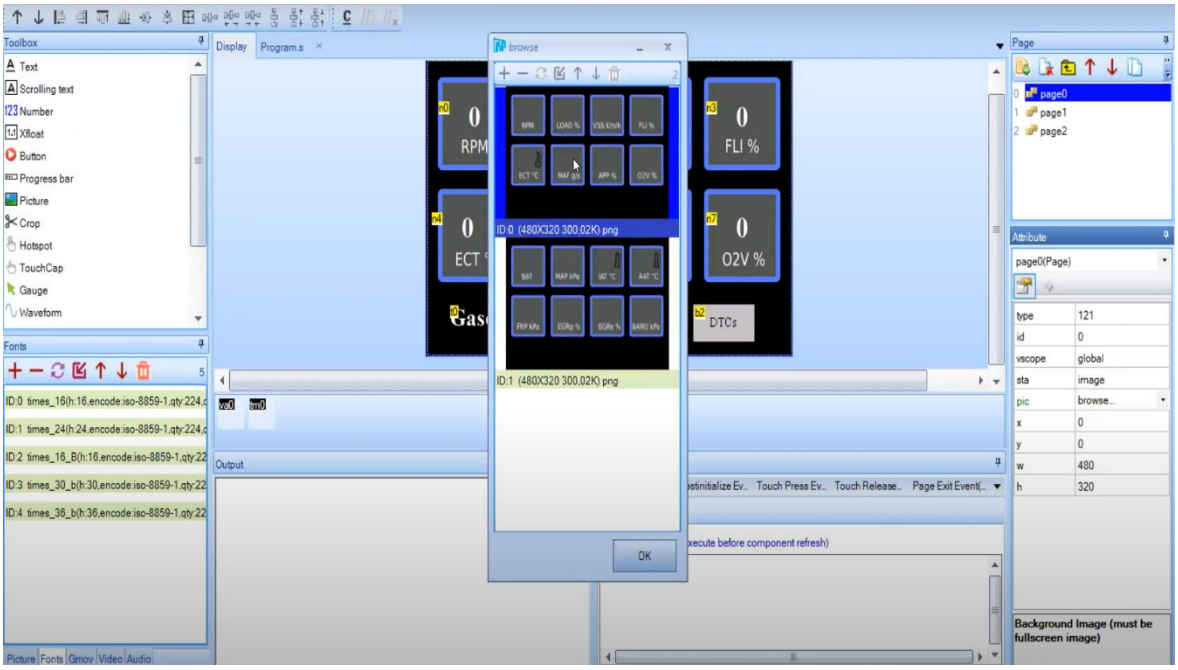
- Dentro del programa de NEXTION se procede agregar las pantallas que se presentaran en la interfaz, véase en la figura 4.13, en la primera página (page 0) se llama a través del nombre con el que se guardó el fondo de gasolina

Figura 4.13: Fondo de interfaz para el modo gasolina



Fondo programado en Nextion HMI, utilizado para la visualización de los datos de los sensores gasolina. Fuente: Autores.

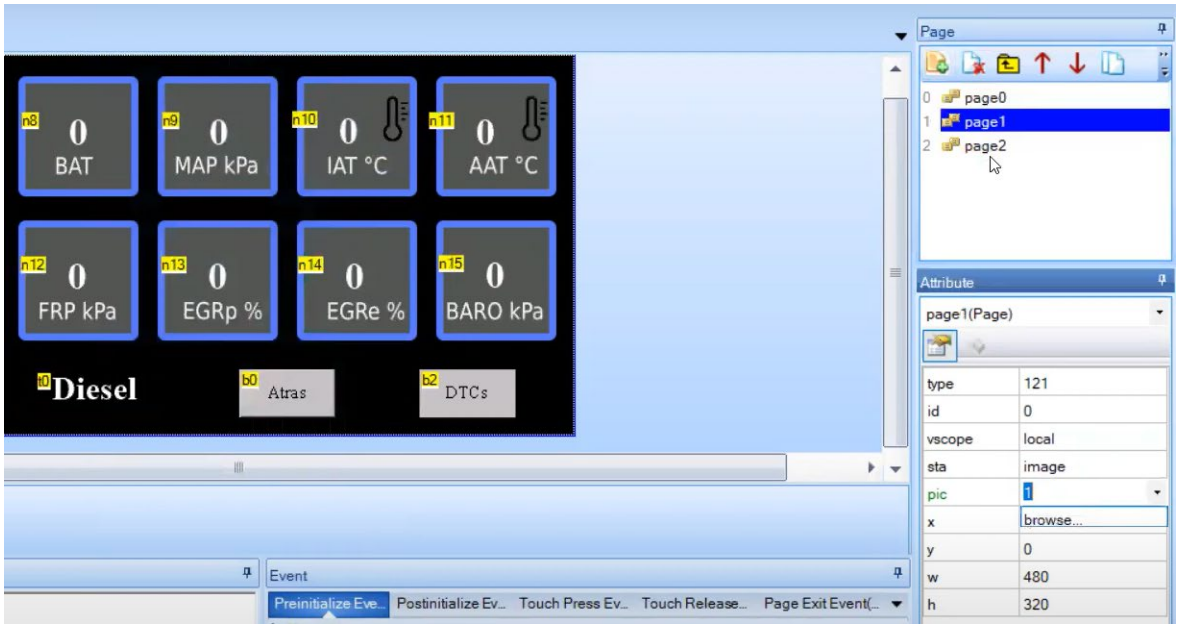
Figura 4.14: Modelo de interfaz para los modos de gasolina y diésel.



Se muestra los modelos de interfaz exportados a el programa Nextion HMI. Fuente: Autores.

- En la segunda página (page 1) se llama al fondo con los parámetros diésel, véase en la figura 4.15.

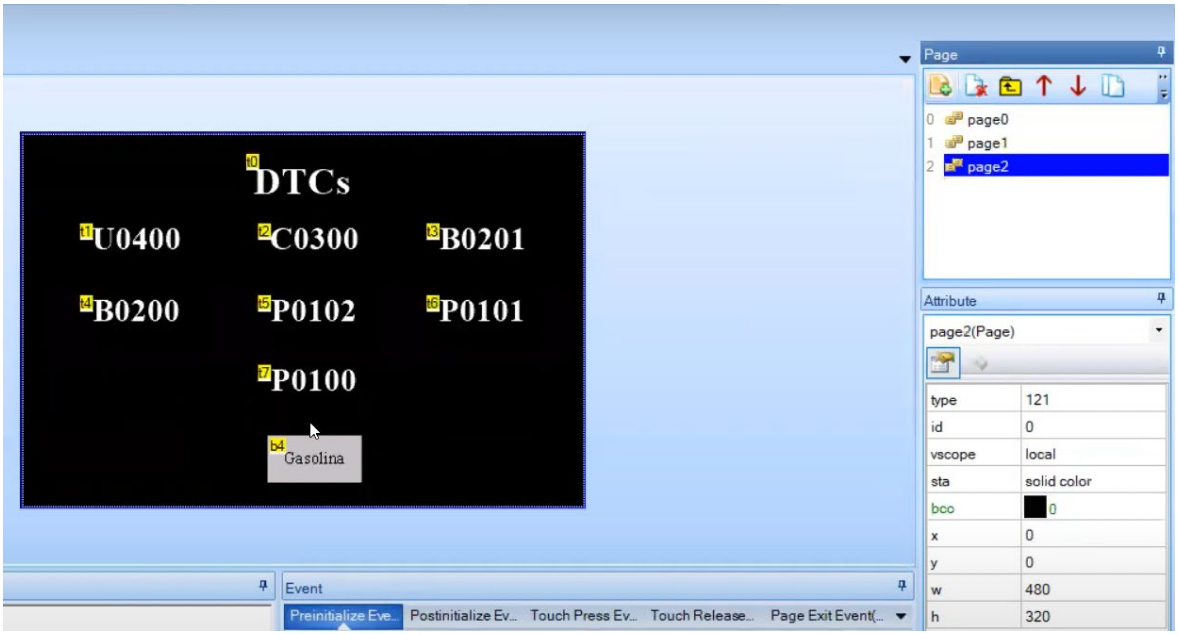
Figura 4.15: Fondo de interfaz para el modo diésel.



Fondo programado en Nextion HMI, utilizado para la visualización de los datos de los sensores diésel. Fuente: Autores.

- En la tercera página (page 2) se llama el fondo con los DTC's, véase en la figura 4.16.

Figura 4.16: Fondo de interfaz para los DTC's.



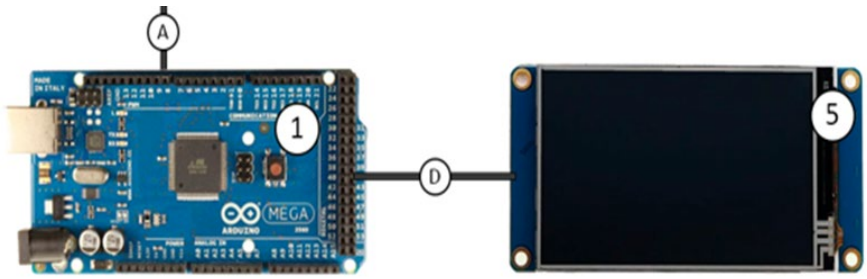
Fondo programado en Nextion HMI, utilizado para la visualización de los fallos (DTC's). Fuente: Autores.

- Posterior a la inclusión de los tres fondos se configura los NUMBERS, en donde se comienza a configurar el STA, el cual debe colocarse como (crop image) esto permite colocar los valores y así poder visualizarlos.
- Los botones se configuran de acuerdo con el propósito que van a cumplir, en este caso, sirven para la navegación entre pantallas.
- A continuación, en el código de Arduino, se define la velocidad a la cual se comunica para configuración de la pantalla NEXTION coincida con dicha velocidad. Esto se lo realiza por medio de la función Serial.begin() en Arduino, especificando la velocidad de transmisión adecuada, como 9600 baudios.
- Una vez establecida la comunicación serie, se envía comandos y datos desde Arduino a la pantalla NEXTION utilizando la función Serial.write(). Estos comandos permiten controlar diversas acciones, tales como indicar u esconder opciones en la interfaz, actualizar valores y manejar eventos táctiles de acuerdo con los datos de los sensores que se desean visualizar.

La pantalla NEXTION envía datos o respuestas al Arduino utilizado con el fin de responder a los eventos táctiles detectados. El Arduino debe estar preparado para recibir estos datos para lo cual se ocupa la función Serial.read() y procesarlos de acuerdo con los requisitos del proyecto.

4.2.5. COMUNICACIÓN CON LA PANTALLA NEXTION ARDUINO

Figura 4.17: Diagrama de conexión entre Arduino y la pantalla Nextion.



Conexión entre Arduino MEGA y la pantalla Nextion de 3.5". Fuente: Autores.

- La interfaz entre la placa Arduino y la pantalla Nextion permite una comunicación bidireccional entre el Arduino Mega utilizado y la pantalla TFT táctil para facilitar la interacción y el control de la interfaz gráfica desarrollada para la presentación de los datos arrojados en los modos: gasolina, diésel y DTC.
- La conexión física se establece mediante cables al serial de la placa Arduino con los pines correspondientes de la pantalla Nextion utilizada. Además de los pines de comunicación, se deben conectar los pines de alimentación y tierra para tener un suministro de energía.
- Para programar la interacción con la pantalla Nextion con los elementos del proyecto, se utiliza la librería específica <Nextion.h> en el entorno de desarrollo de Arduino. Esta librería proporciona funciones y comandos para controlar los elementos gráficos de la pantalla, como botones, textos e imágenes.
- La comunicación entre la placa Arduino y la pantalla Nextion se plantea con la recepción y envió de comandos. Desde el Arduino, se envían comandos para actualizar la interfaz gráfica, mostrar los datos seleccionados de los dos modos que posee la tarjeta mobydic 4910 (diésel y gasolina) haciendo que el usuario pueda cambiar entre los modos establecidos.

4.3. PRESUPUESTO

Tabla 4.3: Presupuesto estimado.

Elemento	Valor estimado
Pantalla Nextion 3.5”	$65,00
KONNWEI KW902	$50,00
Tarjeta mobydic 4910	$700,00
Arduino MEGA	$25,00
Módulo HC-05	$8,00

Total	$848,00

Presupuesto estimado del valor de todos los elementos usados para el proyecto. Fuente: Autores.

CAPÍTULO V

6. ANALISIS E INTERPRETACIÓN DE RESULTADOS

Finalizado el desarrollo se presenta la realización del dashboard para la tarjeta mobydic 4910, es necesario analizar los resultados obtenidos, con el fin de evaluar el cumplimiento del objetivo principal del proyecto.

En aspectos generales, el dashboard elaborado cuenta con 3 ventanas, dispuestas de la siguiente manera:

1. Interfaz modo gasolina
2. Interfaz modo diésel
3. Interfaz de generación de códigos de fallo (DTC)

6.1.INTERFAZ MODO GASOLINA

Al accionar los módulos que están en la figura 4.1, se tiene que esperar a que se establezca la conexión, la primera interfaz que se puede visualizar, es la correspondiente a la modalidad gasolina; por lo cual se podrán visualizar el conjunto de los 8 sensores descritos en el apartado 1.1.4.2.

Esta interfaz cuenta con:

1. Interfaz modo gasolina.
2. Ocho recuadros distribuidos en una matriz 2x4, donde se puede visualizar el nombre de los sensores, sus valores, y las unidades correspondientes.
3. Un botón para acceder a la siguiente interfaz.

Figura 5.1: Interfaz modo gasolina.

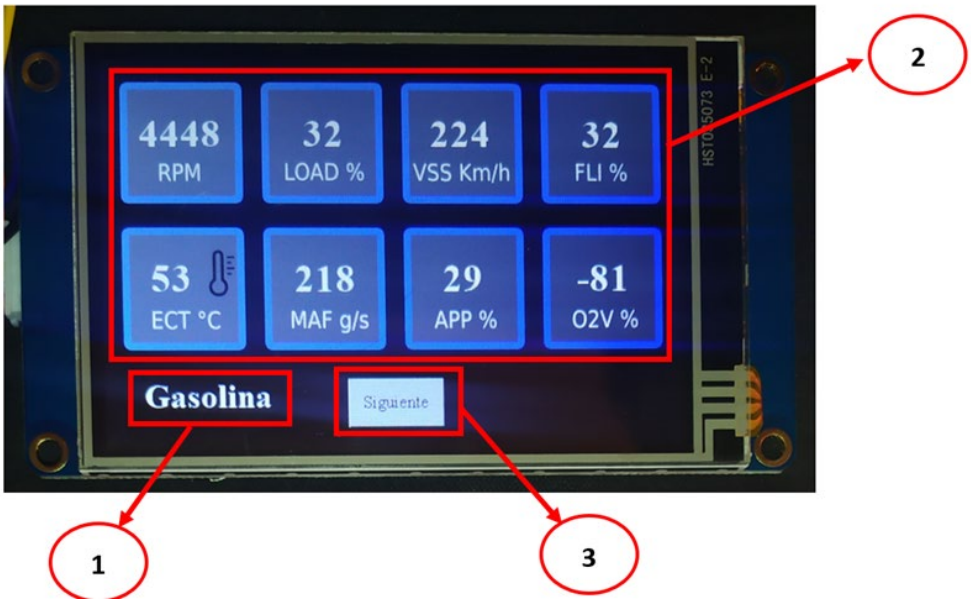
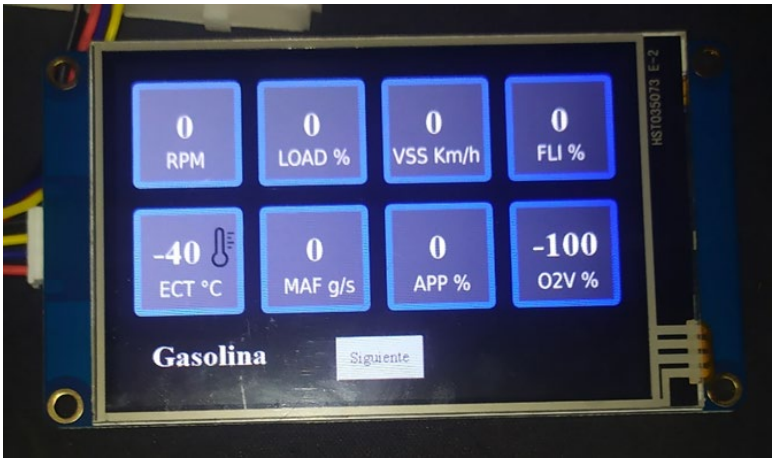


Diagrama de los elementos que conforman la interfaz gasolina. Fuente: Autores.

Para poder determinar si los valores representados en la interfaz gráfica son correctos, es necesario posicionar los potenciómetros en las posiciones de mínima y máxima resistencia, y así comprobar si los valores representados en la pantalla coinciden con los valores de la tabla 4.2.

- Cuando el potenciómetro está colocado en la posición de mínima resistencia, los valores obtenidos son los siguientes:

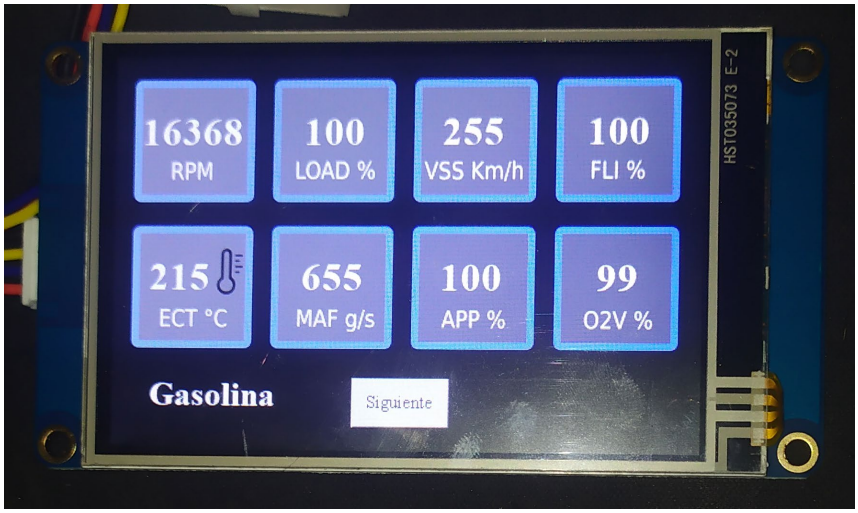
Figura 5.2: Valores mínimos modalidad gasolina.



Parámetros mínimos de la interfaz en modo gasolina. Fuente: Autores.

- Cuando el potenciómetro está colocado en la posición de máxima resistencia, los valores obtenidos son los siguientes:

Figura 5.3: Valores máximos modalidad gasolina.



Parámetros máximos de la interfaz en modo gasolina. Fuente: Autores.

Para los dos casos, se puede observar que la interfaz cumple con el intervalo establecido en la tabla 4.2.

6.2. INTERFAZ MODO DIÉSEL

Para acceder a la interfaz del modo diésel es necesario presionar el botón “siguiente”. Esta interfaz tiene:

1. La interfaz para modalidad diésel.
2. Ocho recuadros distribuidos en una matriz 2x4, donde se puede visualizar el nombre de los sensores, sus valores, y las unidades correspondientes.
3. Un botón para acceder a la interfaz anterior.

Figura 5.4: Interfaz modo diésel.

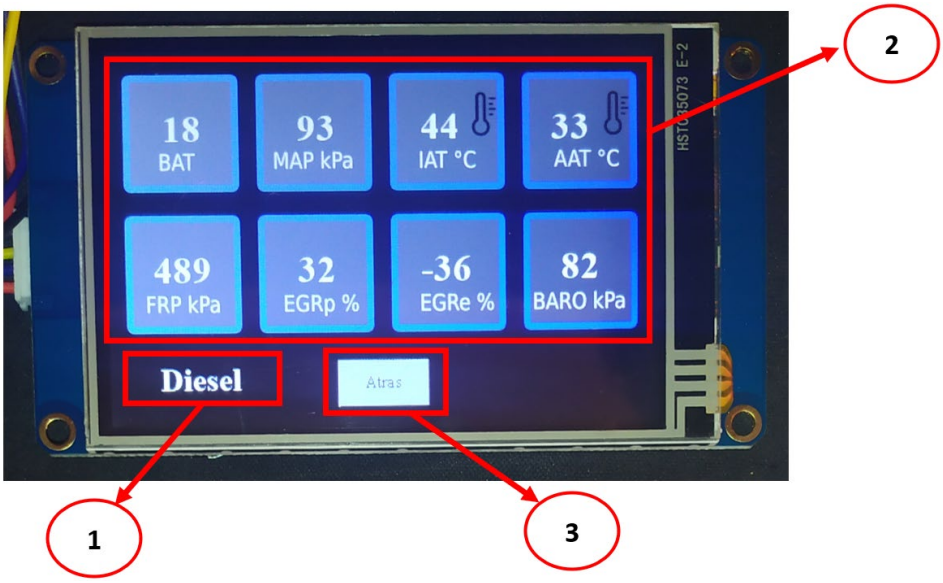
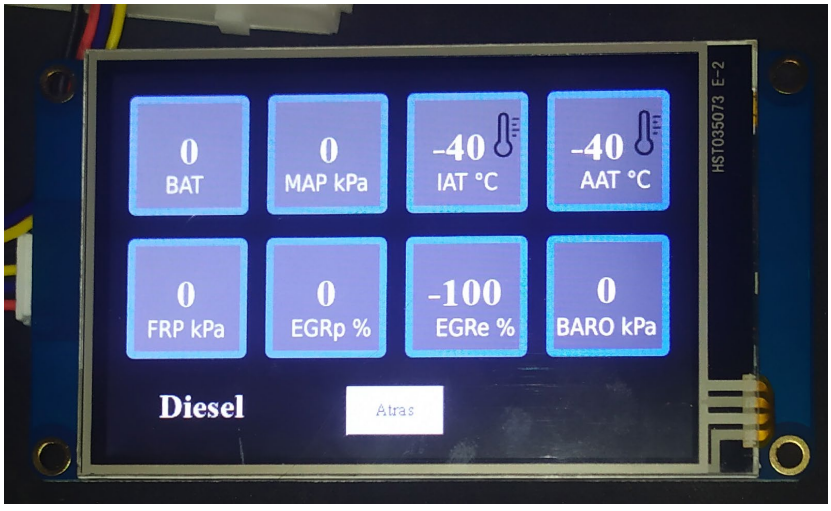


Diagrama de los elementos que conforman la interfaz diésel. Fuente: Autores.

Para poder determinar si los valores representados en la interfaz gráfica del modo diésel son correctos, se tiene que repetir procedimiento descrito en el anterior apartado.

- Cuando el potenciómetro está colocado en la posición de mínima resistencia, los valores obtenidos son los siguientes:

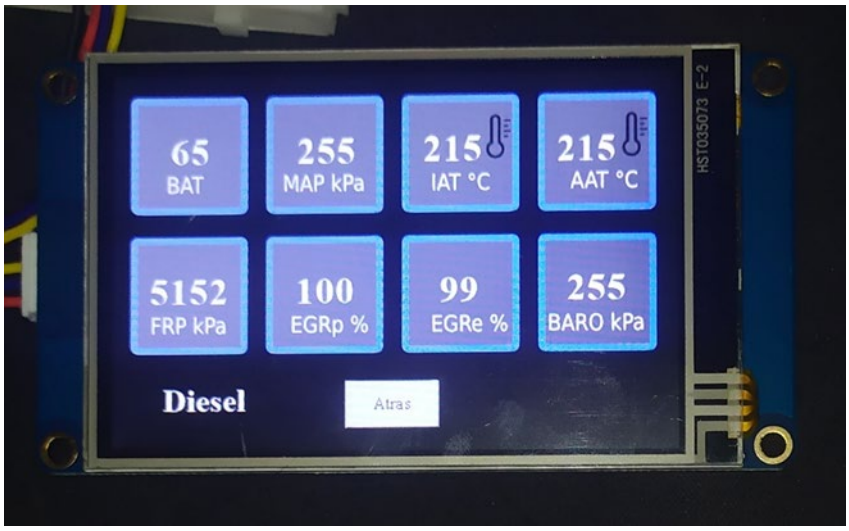
Figura 5.5: Valores mínimos modalidad diésel.



Parámetros mínimos de la interfaz en modo diésel. Fuente: Autores.

- Cuando el potenciómetro está colocado en la posición de máxima resistencia, los valores obtenidos son los siguientes:

Figura 5.6: Valores máximos modalidad diésel.



Parámetros máximos de la interfaz en modo diésel. Fuente: Autores.

Para los dos casos, se puede observar que la interfaz cumple con el intervalo establecido en la tabla 4.1.

6.3. INTERFAZ DTC

Tanto la interfaz de modo gasolina como la interfaz de modo diésel, permanecen activas de forma permanente, por lo cual se va a poder visualizar los datos en todo momento; en lo que respecta interfaz de generación de códigos de fallo (DTC), su acceso es a través de un botón denominado “DTCs” que aparece en la parte inferior de las otras interfases y, se activa al presionar el botón DTC del panel de EERROR CODE de la tarjeta mobydic 4910.

Figura 5.7: Boton DTCs.

Botón DTCs que permite el acceso a la interfaz de códigos de fallo. Fuente: Autores.

En la interfaz referente al modo DTC, se aprecia:

1. El distintivo de la interfaz que se está visualizando.
2. Los siete códigos de fallo que generados por la tarjeta mobydic 4910.
3. Un botón para acceder a la interfaz gasolina.

Figura 5.8: Interfaz modo DTC.

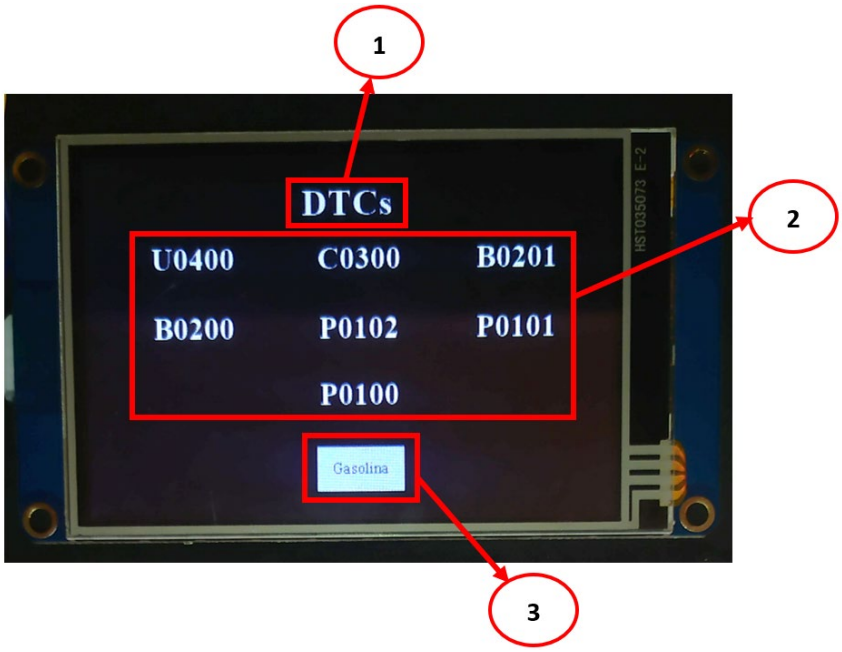


Diagrama de los elementos que conforman la interfaz DTC's. Fuente: Autores.

Al revisar los códigos de falla presentes en la interfaz y compararlos con el apartado 1.1.4.3, se puede apreciar que consta de los 7 fallos producidos por la tarjeta mobydic 4910, por lo cual los resultados obtenidos por esta interfaz son correctos.

CONCLUSIONES

- En base al análisis realizado en el Capítulo V, se determina que, la interfaz desarrollada para cada modo de la tarjeta mobydic 4910 cumplió con el principal objetivo de mostrar los valores de los sensores tanto de diésel como de gasolina, así como para los códigos de fallo DTC's, de forma fiable. Además, el dashboard desarrollado, cuenta con una interfaz gráfica amigable que presenta los datos recopilados de manera clara y organizada, mejorando así la comprensión de datos para los usuarios en general.
- El desarrollo de un panel de visualización de datos para la tarjeta mobydic4910 y el protocolo CAN bus ha sido un objetivo fundamental en este proyecto, ya que permite aprovechar al máximo la información obtenida a través de la interpretación de las señales para que puedan ser visualizadas en la pantalla Nextion.
- Respecto a todos los protocolos CAN bus que dispone la tarjeta descritos en el apartado 1.1.3.1, para fines prácticos del proyecto, no presentan diferencias significativas en lo que respecta a la comunicación, de ser necesario, es factible optar por el uso de otro protocolo que no sea el ISO 15765-4 CAN (29 bit ID, 500 kbaud).
- El uso de componentes electrónicos digitales para el desarrollo de este proyecto refleja las obvias ventajas que representa los sistemas digitales frente a los analógicos, puesto que, el desarrollo metodológico del proyecto, principalmente se centró en la programación. Y en caso de haber optado por el desarrollo de un dashboard tradicional, se hubiesen involucrado procedimientos más pragmáticos relacionados al uso de elementos mecánicos para la representación de los parámetros de la mobydic 4910, que en términos generales hubiesen representado una mayor inversión de recursos económicos, así como en la disponibilidad de tiempo.
- La interfaz gráfica desarrollada en este proyecto proporciona una solución eficiente para la lectura de señales de la mobydic 4910, mejorando la experiencia del usuario y permitiendo una mejor interpretación e información recopilada, puesto que, los datos visualizados en el dashboard no se limitan únicamente a los parámetros representados típicamente por un tablero automotriz analógico. En cada una de las interfaces realizadas se analizó detalladamente los resultados obtenidos, considerando aspectos como la precisión de los datos mostrados, la usabilidad de la

interfaz, la capacidad de personalización, la interacción con los controles y la respuesta en tiempo real.

RECOMENDACIONES

- Se recomienda realizar una investigación exhaustiva sobre las necesidades y requisitos específicos de la manipulación de la tarjeta mobydic 4910 en cuanto a la recolección de datos y accesibilidad que permite la ECU. Esto permitirá comprender mejor los desafíos y las expectativas del proyecto, así como evitar pérdidas de tiempo.
- Se debe presentar los comandos AT de Arduino desde el inicio de la realización del proyecto para que se envite pérdidas de tiempo con pruebas con otros códigos que no presenten los AT's como método de comunicación a través de bluetooth.
- Se sugiere que desde el inicio se trabaje con el módulo bluetooth en modalidad maestro para que se pueda conectar con los comandos AT's teniendo como consecuencia una conexión estable y directa con las variables presentes en el proyecto.
- Se recomienda familiarizarse con el entorno que ofrece el CAN bus y el funcionamiento de la tarjeta mobydic4910. Esto incluye comprender cómo se transmiten y reciben los datos a través del protocolo y cómo se pueden interpretar y utilizar en el diseño del panel de visualización.
- Planificar cuidadosamente el diseño y la arquitectura del panel de visualización de datos. Esto implica determinar qué información se mostrará, cómo se organizará la interfaz y qué funciones y características adicionales se implementarán para mejorar la experiencia del usuario.
- Asegurarse de contar con las herramientas adecuadas para la comunicación y extracción de datos de la tarjeta mobydic4910. Esto implica la configuración de la conexión adecuada con la tarjeta de desarrollo a usar y el sistema de visualización NEXTION, así como el desarrollo del código necesario para extraer y clasificar los datos correctamente.
- Prestar especial atención a la usabilidad y la experiencia del usuario al diseñar la interfaz gráfica con la ayuda de NEXTION HMI. Se recomienda realizar una interfaz intuitiva, clara y fácil de usar, facilitando, además, considerar la personalización para los tres tipos de pantallas que se está ofreciendo al momento de la presentación de datos.

- Considerar la escalabilidad y la posibilidad de futuras actualizaciones en el diseño del panel de visualización. Esto implica utilizar tecnologías y herramientas flexibles que permitan la incorporación de nuevas funcionalidades y la adaptación a cambios futuros en los requerimientos del proyecto.

REFERENCIAS BIBLIOGRÁFICAS.

AUTEL. (2022). *AUTEL*. Obtenido de https://www.autel.com/mk3/3219.jhtml

ELM327, O. (2017). *OBDii ELM327*. Obtenido de OBDii ELM327: https://obd2-elm327.es/ubicacion-aspecto-conector-obd2-esquema

ELMElectronics. (2023). *ELM Electronics.* Obtenido de ELM Electronics: https://j5d2v7d7.stackpathcdn.com/wp-content/uploads/2022/09/ELM327.pdf

KONNWEI. (2020). *KONNWEI*. Obtenido de http://www.konnwei.com/product/418.html

Pinout. (11 de Septiembre de 2002). *DatasheetCafe*. Obtenido de DatasheetCafe: http://www.datasheetcafe.com/elm327-datasheet-microcontroller/

SAE. (2002). *SAE J1979: E/E Diagnostic Test Modes.* Whashington D. C.: Society of Automotive Engineers.

Weis, O. (12 de Abril de 2023). *FLEXIHUB*. Obtenido de https://www.flexihub.com/es/oobd2-pinout/

Automotriz, M., & Eduardo Arévalo Calderón Angel Geovanny Ortega Ulloa Director, F. (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"*.

Beniel Dennyson, & Dr. C. Jothikumar. (2022). *Vista de UNA REVISIÓN DE LA RED DE ÁREA DEL CONTROLADOR Y LA UNIDAD DE CONTROL ELECTRÓNICO EN EL ENTORNO AUTOMOTRIZ*. 1–9.

By ALLDATASHEETCOM, P. (2002). *SN65HVD251_06 TI | Alldatasheet*. www.ti.com

Cavazos Luis Enrique, N., Razo Carlos Hernán, P., Gómez Rosa Ivette, S., & Sánchez Rogelio, C. (2022). *Diseño y construcción de prototipo portátil para exhibición de señal analógica*. 1–6. www.jóvenesenlaciencia.ugto.mx

DEC AUTOMOTIVE DIAGNOSIS TOOLS. (1995). *OBD II*.

Eduardo Arévalo Calderón, & Angel Geovanny Ortega Ulloa. (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"*.

Enríquez Herrador, R. (2009). *Guía de Usuario de Arduino*. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.uco.es/aulasoftwarelibre/wp-content/uploads/2010/05/Arduino_user_manual_es.pdf\

FÉLIX ANDRÉS ANDRADE SÁNCHEZ. (2015). *ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SCANNER AUTOMOTRIZ PARA VEHÍCULOS*

VOLKSWAGEN GOL, PROGRAMADO CON ARDUINO PARA VISUALIZAR EN ANDROID.

García-Tudela, P. A., & Marín-Marín, J. A. (2023). Use of Arduino in Primary Education: A Systematic Review. In *Education Sciences* (Vol. 13, Issue 2, pp. 1–3). MDPI. https://doi.org/10.3390/educsci13020134

Gülnur_Karanfil. (2021). *Uso de Protocolos OBD II Diseñar un software de interfaz*. 1–118.

inkscape.es. (2023). *Inkscape - editor de gráficos vectoriales*. https://inkscape.es/

ISO. (2011, April). *Norma ISO 15031-5:2011*.

Kondaveeti, H. K., Kumaravelu, N. K., Vanambathina, S. D., Mathe, S. E., & Vappangi, S. (2021). A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations. In *Computer Science Review* (Vol. 40). Elsevier Ireland Ltd. https://doi.org/10.1016/j.cosrev.2021.100364

OZEN ELEKTRONIK. (n.d.). *mobydic4910-1*.

POR Álvaro Alberto Giner, R., & Armesto Ángel, L. (n.d.). *DISEÑO, PROGRAMACIÓN Y CONSTRUCCIÓN DE UN ROBOT HEXÁPODO BASADO EN ARDUINO Y CONTROLADO A TRAVÉS DEL MÓDULO BLUETOOTH MEDIANTE UNA APLICACIÓN DE TELÉFONO MÓVIL*. Retrieved 28 June 2023, from chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://riunet.upv.es/bitstream/handle/10251/148343/Alberto%20-%20Dise%c3%b1o%2c%20programaci%c3%b3n%20y%20construcci%c3%b3n%20de%20un%20robot%20hex%c3%a1podo%20basado%20en%20Arduino%20y%20controlado....pdf?sequence=1&isAllowed=y

Smith, C. (2006). *THE CAR HACKER'S HANDBOOK*.

Smith, C. (2016). *THE CAR HACKER'S HANDBOOK*.

Society of Automotive Engineers. (2002). *SAE J1979: E/E Diagnostic Test Modes*.

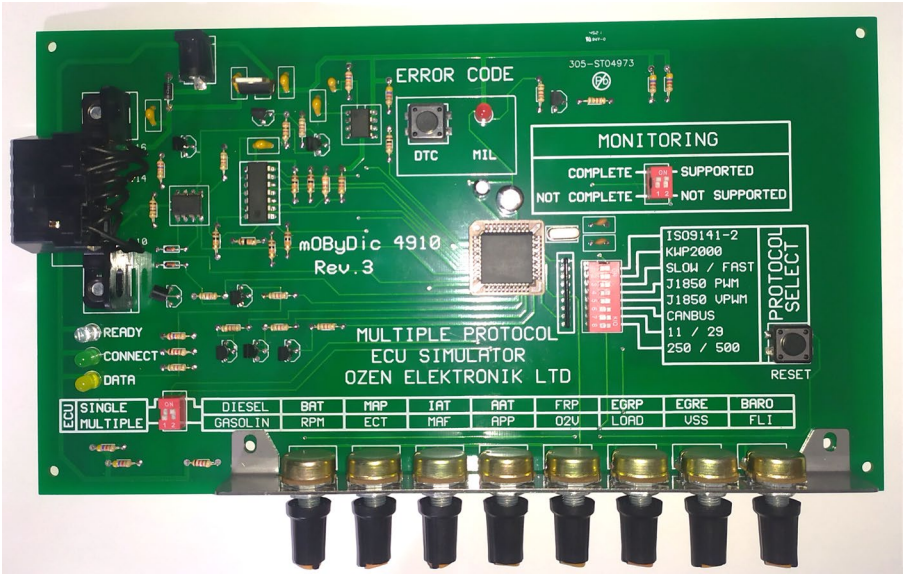
Yadav, P., Peeyush, M., & Pathak, K. (2021). On Board Diagnostics (OBD): A Review On Monitoring Vehicle Operations. In *Organized by: International Journal of Research*.

Zakaria, S., Mativenga, P., & Ariff, E. A. R. E. (2023). An Investigation of Energy Consumption in Fused Deposition Modelling using ESP32 IoT Monitoring System. *Procedia CIRP*, *116*, 263–268. https://doi.org/10.1016/j.procir.2023.02.045

Zambrano, J. B., García, F. B., & Sánchez García, J. (2015). *Desarrollo de un simulador electrónico de una ECU y su diagnóstico sobre CAN y OBD-II*

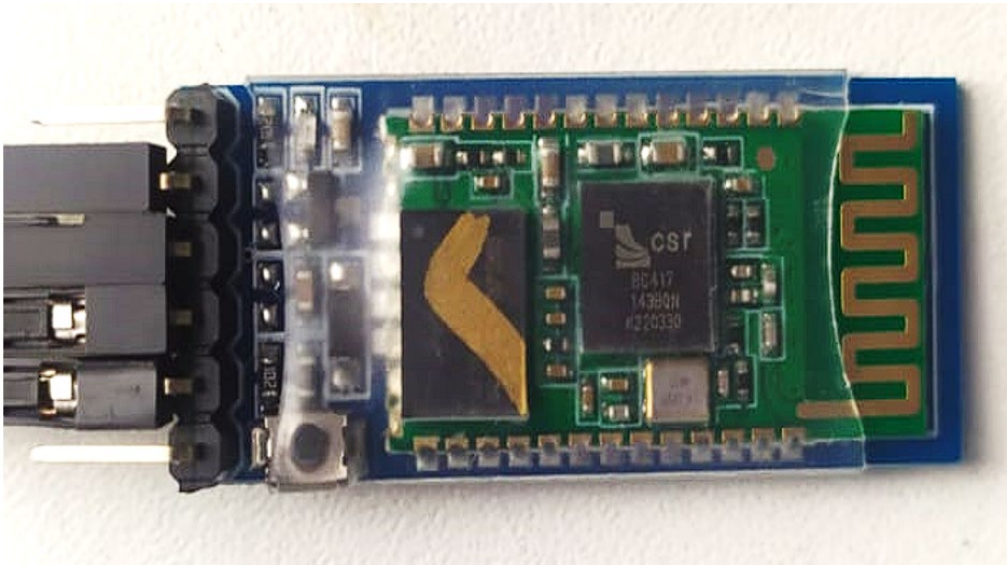
ANEXOS.

Anexo 1. Tarjeta mobydic 4910.

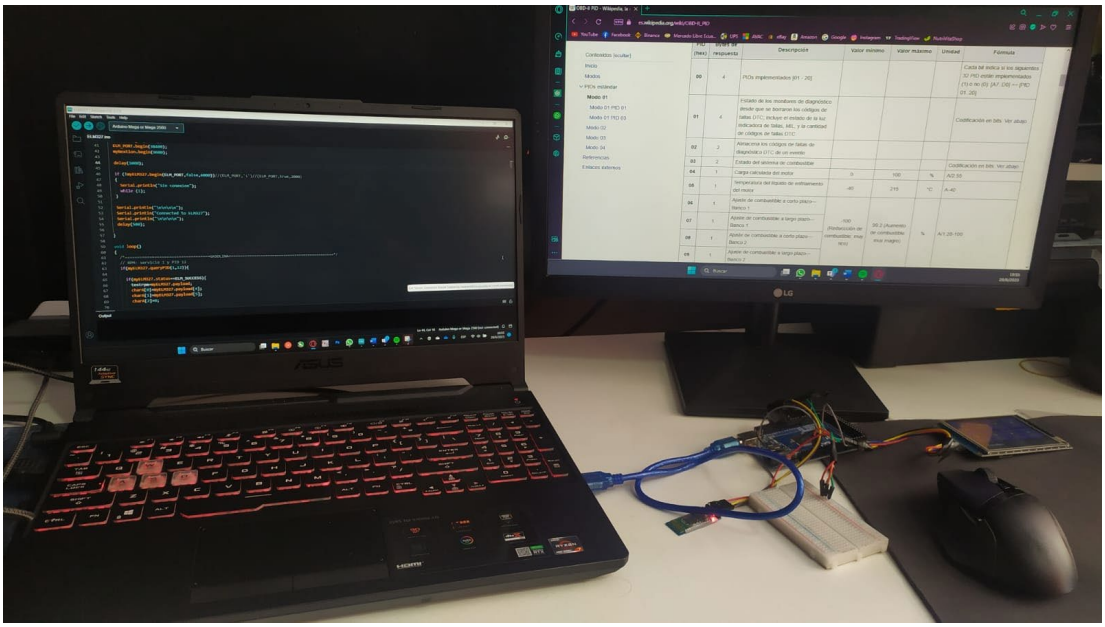


Anexo 2. KONNWEI KW902.

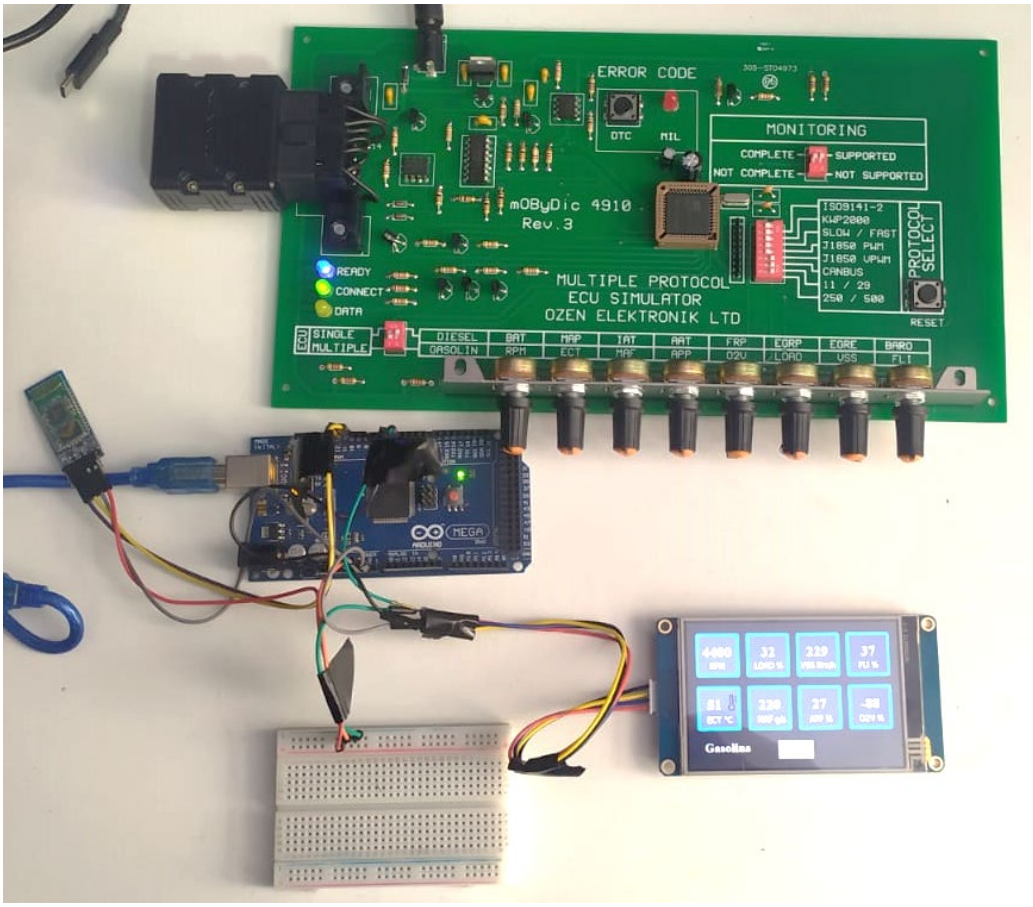
Anexo 3. Módulo bluetooth HC-05.



Anexo 4. Programación de los elementos Arduino.



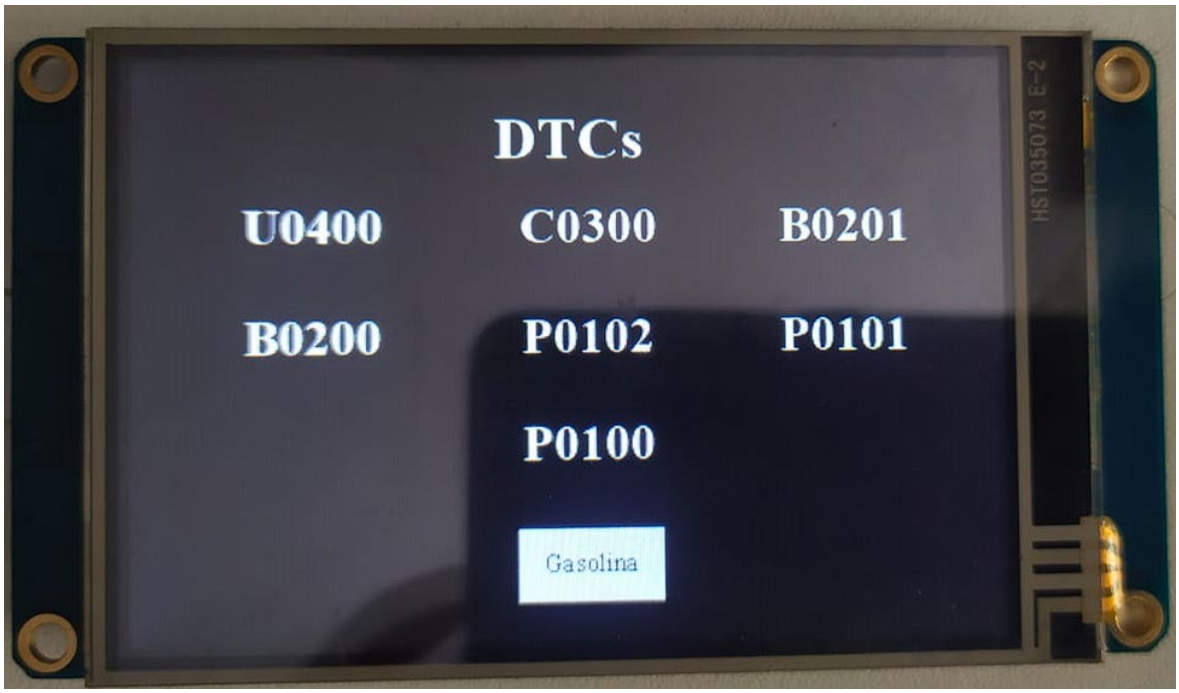
Anexo 5. Elementos que conforman el circuito del proyecto.



Anexo 6. Interfaz modo gasolina.

Anexo 7. Interfaz modo diésel.

Anexo 8. Interfaz modo DTC.



Anexo 9. Código de Arduino

```
#include <SoftwareSerial.h>
#include "ELMduino.h"

SoftwareSerial mySerial(10, 11); // RX, TX
SoftwareSerial myNextion(4, 5); // RX, TX

#define ELM_PORT mySerial


ELM327 myELM327;


/*
  rpm=revolutions per minutes
  ect=engine coolant temperature
  iat=intake Air Temperature;
  aat=Ambient air temperature
  frp=Fuel Rail Pressure;
  el=engine Load;
  kph= kilometers per hour
  fl=fuel Level;
*/
char* testrpm;
```

```
char charA[3];
char charB[3];



float rpm=0,el=0,kph=0,fl=0,ect=0,maf=0,app=0,o2v=0; // GASOLINA
float bat=0,maap=0,iat=0,aat=0,egrp=0,egre=0,baro=0;// DIESEL

float frp=0;

float DTC=0;


void setup()
{
  Serial.begin(38400);
  ELM_PORT.begin(38400);
  myNextion.begin(9600);

  delay(1000);

  if
(!myELM327.begin(ELM_PORT,false,6000))//(ELM_PORT,'1')//(ELM_PORT,true,2000)
  {
    Serial.println("Sin conexion");
    while (1);
  }

  Serial.println("\n\n\n\n");
  Serial.println("Connected to ELM327");
  Serial.println("\n\n\n\n");
  delay(500);

}

void loop()
{
  /*==================================GASOLINA============================
==================*/
  // RPM: servicio 1 y PID 12
  if(myELM327.queryPID(1,12)){

    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;
```

```
    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
    charB[2]=0;


    int A=(strtoul(charA,NULL,16));
    int B=(strtoul(charB,NULL,16));

    rpm=(64*A+0.25*B);
    //Serial.print("RPM: ");
    //Serial.print(rpm);


    myNextion.print("n0.val=");
    myNextion.print(round(rpm));
    myNextion.write(0xff);
    myNextion.write(0xff);
    myNextion.write(0xff);

  }else{
    Serial.println(myELM327.status);
  }
}

// Engine Load: servicio 1 PID 4
if(myELM327.queryPID(1,4)){

  if(myELM327.status==ELM_SUCCESS){
    testrpm=myELM327.payload;
    charA[0]=myELM327.payload[4];
    charA[1]=myELM327.payload[5];
    charA[2]=0;

    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
    charB[2]=0;


    int A=(strtoul(charA,NULL,16));
    int B=(strtoul(charB,NULL,16));

    el=(A/2.55);
    //Serial.print("   EL: ");
    //Serial.print(el);
```

```
    myNextion.print("n1.val=");
    myNextion.print(round(el));
    myNextion.write(0xff);
    myNextion.write(0xff);
    myNextion.write(0xff);

  }else{
    Serial.println(myELM327.status);
  }
}

// Vehicle Speed: servicio 1 PID 13
if(myELM327.queryPID(1,13)){

  if(myELM327.status==ELM_SUCCESS){
    testrpm=myELM327.payload;
    charA[0]=myELM327.payload[4];
    charA[1]=myELM327.payload[5];
    charA[2]=0;

    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
    charB[2]=0;


    int A=(strtoul(charA,NULL,16));
    int B=(strtoul(charB,NULL,16));

    kph=A;
    //Serial.print("   KpH: ");
    //Serial.print(kph);

    myNextion.print("n2.val=");
    myNextion.print(round(kph));
    myNextion.write(0xff);
    myNextion.write(0xff);
    myNextion.write(0xff);

  }else{
    Serial.println(myELM327.status);
  }
}



// fuel level: servicio 1 PID 47
if(myELM327.queryPID(1,47)){
```

```
  if(myELM327.status==ELM_SUCCESS){
    testrpm=myELM327.payload;
    charA[0]=myELM327.payload[4];
    charA[1]=myELM327.payload[5];
    charA[2]=0;

    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
    charB[2]=0;


    int A=(strtoul(charA,NULL,16));
    int B=(strtoul(charB,NULL,16));

    fl=A/2.55;
    //Serial.print("   FL: ");
    //Serial.print(fl);

    myNextion.print("n3.val=");
    myNextion.print(round(fl));
    myNextion.write(0xff);
    myNextion.write(0xff);
    myNextion.write(0xff);

  }else{
    Serial.println(myELM327.status);
  }
}


// Engine Coolant Temperature: servicio 1 PID 5
if(myELM327.queryPID(1,5)){

  if(myELM327.status==ELM_SUCCESS){
    testrpm=myELM327.payload;
    charA[0]=myELM327.payload[4];
    charA[1]=myELM327.payload[5];
    charA[2]=0;

    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
    charB[2]=0;


    int A=(strtoul(charA,NULL,16));
    int B=(strtoul(charB,NULL,16));
```

```
    ect=A-40;
    //Serial.print("   ECT: ");
    //Serial.print(ect);

    myNextion.print("n4.val=");
    myNextion.print(round(ect));
    myNextion.write(0xff);
    myNextion.write(0xff);
    myNextion.write(0xff);

  }else{
    Serial.println(myELM327.status);
  }
}



// maf: servicio 1 PID 16
if(myELM327.queryPID(1,16)){

  if(myELM327.status==ELM_SUCCESS){
    testrpm=myELM327.payload;
    charA[0]=myELM327.payload[4];
    charA[1]=myELM327.payload[5];
    charA[2]=0;

    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
    charB[2]=0;


    int A=(strtoul(charA,NULL,16));
    int B=(strtoul(charB,NULL,16));

    maf=2.56*A+0.01*B;
    //Serial.print("   MAF: ");
    //Serial.print(maf);

    myNextion.print("n5.val=");
    myNextion.print(round(maf));
    myNextion.write(0xff);
    myNextion.write(0xff);
    myNextion.write(0xff);

  }else{
    Serial.println(myELM327.status);
```

```
  }
}



// Accelerator Pedal Position D: servicio 1 PID 73
if(myELM327.queryPID(1,73)){
  if(myELM327.status==ELM_SUCCESS){
    testrpm=myELM327.payload;
    charA[0]=myELM327.payload[4];
    charA[1]=myELM327.payload[5];
    charA[2]=0;

    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
    charB[2]=0;


    int A=(strtoul(charA,NULL,16));
    int B=(strtoul(charB,NULL,16));

    app=A/2.55;
    //Serial.print("   APP: ");
    //Serial.print(app);

    myNextion.print("n6.val=");
    myNextion.print(round(app));
    myNextion.write(0xff);
    myNextion.write(0xff);
    myNextion.write(0xff);

  }else{
    Serial.println(myELM327.status);
  }
}


// O2V: servicio 1 PID 20
if(myELM327.queryPID(1,20)){
  if(myELM327.status==ELM_SUCCESS){
    testrpm=myELM327.payload;
    charA[0]=myELM327.payload[4];
    charA[1]=myELM327.payload[5];
    charA[2]=0;

    charB[0]=myELM327.payload[6];
    charB[1]=myELM327.payload[7];
```

```
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      o2v=(A/1.28-100);
      //Serial.println("   O2V: ");
      //Serial.print(o2v);
      //Serial.println("");

      myNextion.print("n7.val=");
      myNextion.print(round(o2v));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }

/*=========================================DIESEL=======================
==============================*/

//  control module voltaje BAT: servicio 1 PID 66
  if(myELM327.queryPID(1,66)){
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      bat=0.256*A+0.001*B;
      Serial.print("BAT: ");
      Serial.print(bat);

      myNextion.print("n8.val=");
      myNextion.print(round(bat));
```

```
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }



//  Intake manifold absolute pressure (MAP): servicio 1 PID 11
  if(myELM327.queryPID(1,11)){
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      maap=A;
      Serial.print("   MAP: ");
      Serial.print(maap);

      myNextion.print("n9.val=");
      myNextion.print(round(maap));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }

//  Intake Air temperature: servicio 1 PID 15
  if(myELM327.queryPID(1,15)){
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
```

```
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      iat=A-40;
      Serial.print("   IAT: ");
      Serial.print(iat);

      myNextion.print("n10.val=");
      myNextion.print(round(iat));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }


//  Ambient Air temperature: servicio 1 PID 70
  if(myELM327.queryPID(1,70)){
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      aat=A-40;
      Serial.print("   AAT: ");
      Serial.print(aat);
```

```
      myNextion.print("n11.val=");
      myNextion.print(round(aat));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }


//  Fuel Rail Gauge Presure: servicio 1 PID 35
  if(myELM327.queryPID(1,35)){
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      frp=20.145*A+0.079*B;
      //frp=2^16-1;
      Serial.print("   FRP: ");
      Serial.print(frp);

      myNextion.print("n12.val=");
      myNextion.print(round(frp));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }


//  EGRp: servicio 1 PID 44
  if(myELM327.queryPID(1,44)){
```

```
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      egrp=A/2.55;
      Serial.print("   EGRp: ");
      Serial.print(egrp);

      myNextion.print("n13.val=");
      myNextion.print(round(egrp));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }


//  EGRe: servicio 1 PID 45
  if(myELM327.queryPID(1,44)){
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      egre=A/1.28-100;
```

```
      Serial.print("   EGRe: ");
      Serial.print(egre);

      myNextion.print("n14.val=");
      myNextion.print(round(egre));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }


//  BARO: servicio 1 PID 51
  if(myELM327.queryPID(1,44)){
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
      charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      baro=A;
      Serial.print("   BARO: ");
      Serial.println(baro);

      myNextion.print("n15.val=");
      myNextion.print(round(baro));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }
```

```
  /*==========================DTC=======================================
============*/
//  servicio 3 PID ##
  if(myELM327.queryPID(3,1)){
    if(myELM327.status==ELM_SUCCESS){
      testrpm=myELM327.payload;
      charA[0]=myELM327.payload[4];
      charA[1]=myELM327.payload[5];
      charA[2]=0;

      charB[0]=myELM327.payload[6];
      charB[1]=myELM327.payload[7];
    charB[2]=0;


      int A=(strtoul(charA,NULL,16));
      int B=(strtoul(charB,NULL,16));

      DTC=A+B;
      Serial.print("   DTC: ");
      Serial.println(DTC);


      myNextion.print("va0.val=");
      myNextion.print(round(DTC));
      myNextion.write(0xff);
      myNextion.write(0xff);
      myNextion.write(0xff);

    }else{
      Serial.println(myELM327.status);
    }
  }

}
```