UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo
a la obtención del título de
Ingeniero Electrónico

PROYECTO TÉCNICO:

RECOLECCIÓN, ANÁLISIS Y ENVÍO DE DATOS DE LA UNIDAD DE CONTROL DE MOTOR (ECU) A BORDO DE UN VEHÍCULO PARA MONITOREO A TRAVÉS DE UNA APLICACIÓN MÓVIL

AUTORES:

JONNATHAN IVÁN AGUILAR GUERRERO

JEFFERSON STEEVEN REYES ZAMBRANO

TUTOR:

ING. EDGAR EFRAÍN OCHOA FIGUEROA, MGT.

CUENCA - ECUADOR

2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Jonnathan Iván Aguilar Guerrero con documento de identificación N° 0105462246 y Jefferson Steeven Reyes Zambrano con documento de identificación N° 0706441821, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación: **"RECOLECCIÓN, ANÁLISIS Y ENVÍO DE DATOS DE LA UNIDAD DE CONTROL DE MOTOR (ECU) A BORDO DE UN VEHÍCULO PARA MONITOREO A TRAVÉS DE UNA APLICACIÓN MÓVIL"**, mismo que ha sido desarrollado para optar por el título de: *Ingeniero Electrónico,* en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, noviembre de 2021.

Jonnathan Iván Aguilar Guerrero
C.I. 0105462246

Jefferson Steeven Reyes Zambrano
C.I. 0706441821

CERTIFICACIÓN

Yo, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"RECOLECCIÓN, ANÁLISIS Y ENVÍO DE DATOS DE LA UNIDAD DE CONTROL DE MOTOR (ECU) A BORDO DE UN VEHÍCULO PARA MONITOREO A TRAVÉS DE UNA APLICACIÓN MÓVIL"**, realizado por Jonnathan Iván Aguilar Guerrero y Jefferson Steeven Reyes Zambrano, obteniendo el *Proyecto Técnico*, que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

Cuenca, noviembre de 2021.

EDGAR EFRAIN OCHOA FIGUEROA

Firmado digitalmente por EDGAR EFRAIN OCHOA FIGUEROA
Fecha: 2021.05.10 19:20:27 -05'00'

Ing. Edgar Ochoa Figueroa, MgT.

C.I. 0102134574

DECLARATORIA DE RESPONSABILIDAD

Nosotros, Jonnathan Iván Aguilar Guerrero con documento de identificación N° 0105462246 y Jefferson Steeven Reyes Zambrano con documento de identificación N° 0706441821, autores del trabajo de titulación: **"RECOLECCIÓN, ANÁLISIS Y ENVÍO DE DATOS DE LA UNIDAD DE CONTROL DE MOTOR (ECU) A BORDO DE UN VEHÍCULO PARA MONITOREO A TRAVÉS DE UNA APLICACIÓN MÓVIL",** certificamos que el total contenido del *Proyecto Técnico,* es de nuestra exclusiva responsabilidad y autoría

Cuenca, noviembre de 2021.

Jonnathan Iván Aguilar Guerrero
C.I. 0105462246

Jefferson Steeven Reyes Zambrano
C.I. 0706441821

AGRADECIMIENTOS

A mi padre, Ivan, por ser ejemplo de superación, porque en todo momento estuvo para apoyarme y brindarme fortaleza para alcanzar mis metas anheladas. Porque nunca tuvo duda alguna de mis capacidades y también nunca permitirme dudar acerca de él.

A mi madre, Margarita, mi compañera, mi confidente, que, con su dedicación, aliento y amor incondicional, me ha acompañado a lo largo de mi etapa de superarme como persona, como profesional y lo más importante como ser humano.

A mis docentes por todos los conocimientos entregados a lo largo de mi carrera, por su paciencia, por su entrega en la ardua labor de la enseñanza. Al Ing. Edgar Ochoa por su tutela y consejos.

A mis compañeros, amigos que fueron parte de este largo camino de aprendizaje, a todos muchas gracias.

Jonnathan Ivan Aguilar Guerrero

Primeramente le doy gracias a Dios por haberme permitido culminar un logro mas que son resultados de tu ayuda, por tener a mi familia y disfrutarla.

A mi apreciado padre, un ejemplo a seguir, a pesar de la distancia siempre me supo ayudar en todo momento y darme sus consejos. Por haberme dado mucha sabiduría durante estos 25 años de vida y por no hacer que me rinda.

A mi madre querida, mi vida entera, la que siempre ha estado para todo en buenos y malos momentos, con su dedicacion, esfuerzo y con mucho cariño estuvo acompañandome durante toda esta etapa de mi vida para convertirme en un gran profesional y forjarme como la persona que soy.

Agradezco a mis hermanos Edison, Jinsonp y Bryan por su apoyo, amor y consejos. A pesar de todas las dificultades que se han presentado me han dado su mano y haberme ayudado a salir adelante.

A todo el personal docente que me ayudaron con su vocación a enriquecerme de conocimiento. Al Ing. Edgar Ochoa por darme la oportunidad de recurrir a su conocimiento y capacidad como profesional. Al Ing. Fernando Guerrero por haberme tenido paciencia para guiarme en los ultimos semestres de la carrera.

Mi agradecimiento tambien va para el Instituo de Fomento al Talento Humano, que a través de su beca Eloy Alfaro me brindaron su apoyo económico para culminar mis estudios.

A todos los compañeros de clase, ya que con su compañerismo y amistad me ayudaron a que siga adelante.

A todo el departamento de Bienestar Estudiantil de la UPS, que gracias a su arduo labor en actividades como proyectos, programas, políticas y normativas específicas que favorecen el desarrollo y bienestar de la comunidad estudiantil.

Finalmente, agradezco a Christopher compañero desde la infancia, por haberme aguantado y apoyado en todo este largo camino hasta convertirme en profesional. Y a mis amigos Kevin, Belen, David y Angélica que se han sido parte de este proceso, a todos gracias totales.

Jefferson Steeven Reyes Zambrano

DEDICATORIAS

Dedico este trabajo a mi papá y a mi mamá, a mis hermanos, abuelos, tíos, primos, a mi familia en general que siempre fueron fuente de apoyo incondicional hacia mi persona y nunca perdieron la fe en que lograría mis objetivos.

A mi abuelo Jesús que siempre preguntaba cómo iban mis estudios, mi dedicatoria hacia él en el cielo.

Jonnathan Ivan Aguilar Guerrero

Dedico esta tesis a mis padres Jorge Gonzalo Reyes León e Irma Angelita Zambrano Torres que fueron el pilar fundamental para convertirme en la persona que soy actualmente y siempre me dieron su apoyo incondicional.

A mis hermanos y familia en general que siempre estuvieron brindándome su apoyo durante todo el proceso de mi carrera Universitaria.

Jefferson Steeven Reyes Zambrano

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El uso de automóviles en las empresas se ha intensificado en la actualidad, lo que ha generado la necesidad de tener un control constante de los mismos. En este contexto, el presente trabajo propone el diseño de un sistema para monitoreo remoto de un vehículo. El sistema tiene como objetivo medir la velocidad, distancia y consumo de combustible del vehículo para el seguimiento y el análisis del automóvil. El proceso a seguir consiste en obtener los datos de la Unidad de Control de Motor (ECU, por sus siglas en inglés Engine Control Unit) y enviarlos a una base de datos en la nube. Para ello se trabajó con el puerto OBD2, que está presente en todos los vehículos por normativa.

Para acceder a los datos del vehículo se utilizó el escáner llamado ELM327, el cual realiza la lectura de los datos presentes en la ECU y los envía mediante Bluetooth a un dispositivo móvil que realiza la función de procesar, visualizar y enviar los datos mediante red móvil. El proyecto cuenta con una aplicación para dispositivos móviles basado en el sistema operativo Android, el cual se encarga de tomar los datos provenientes del escáner, procesarlos, visualizarlos en el teléfono móvil y enviar a una base de datos en la nube utilizando la red de datos del teléfono. La base de datos está realizada en Cloud Firestores que es un servicio de Firebase, este servicio permite tener un esquema más sistemático, logrando acceder remotamente para la consulta de datos. La aplicación también implementa un sistema de posicionamiento global para determinar la ubicación del vehículo utilizando el GPS interno del teléfono móvil.

Al final, se realizan pruebas para verificar la funcionalidad del proyecto. Los resultados demuestran que el sistema diseñado es capaz de leer los distintos parámetros y puede procesar, transmitir y mostrar las lecturas.

INTRODUCCIÓN

Con el avance de la tecnología se han desarrollado avances significativos en los dispositivos electrónicos y el sistema automotriz no es la excepción. Con el pasar de los años, los automóviles han implementado sistemas electrónicos que mejoran el rendimiento, la seguridad y comodidad de los mismos. Por otra parte, los motores de los vehículos modernos cuentan con un sistema llamado Unidad de Control de Motor (ECU, por sus siglas en inglés Engine Control Unit), el cual equivale al cerebro del automóvil que controla diversos parámetros optimizando el funcionamiento dependiendo de las circunstancias. La ECU optimiza el rendimiento del motor basándose en los datos obtenidos de varios sensores en el auto; por ejemplo, los niveles de oxígeno en el aire afectarán la tasa de combustión del combustible (más oxígeno significa una tasa de quemado más rápida). A través de los datos obtenido del sensor de oxígeno, la ECU también puede ajustar el volumen de combustible entregado al motor.

Para acceder a la información que está presente en la ECU se utiliza el sistema "Diagnóstico a bordo" (OBD, por sus siglas en inglés On Board Diagnostic). Específicamente se utiliza el sistema OBD II, el cual es un estándar desarrollado en los Estados Unidos de América (EE.UU.) en 1996, por la Sociedad de Ingenieros Automotrices (SAE) [1]. Esta especificación fue definida para todos los vehículos fabricados para permitir la regulación de las emisiones de los vehículos. El desarrollo del OBD II también dió como resultado el desarrollo de herramientas de escaneo OBD II, conocidas como lectores OBD II. Estos lectores, pueden interactuar con cualquier vehículo a través de un puerto de 16 pines. Una herramienta de escaneo típicamente solicita información al ECU enviando un mensaje que contiene un código hexadecimal asociado a un parámetro específico [2].

Los datos OBD se pueden leer a través del puerto OBD y hay varios adaptadores disponibles comercialmente para leer los datos del puerto OBD II, una opción

accesible y de bajo costo es el escáner ELM327, con éste escáner los datos son leídos desde el puerto OBD II y se transmiten a través de Bluetooth o WIFI durante el emparejamiento. Estos datos se pueden visualizar en un teléfono o una computadora.

En el presente proyecto se realiza un prototipo de prueba tomando como base el estándar OBD II y un scanner que realiza la lectura y visualiza en tiempo real los datos que envía la ECU del automóvil a través de conexión Bluetooth, para luego guardarlos en una base de datos, utilizando la red de telefonía móvil.

ANTECEDENTES DEL PROBLEMA DE ESTUDIO

Dentro de la industria automotriz han surgido nuevas tecnologías obteniendo buenos resultados mejorando la calidad y seguridad del vehículo. Con este adelanto los usuarios, han hecho que el vehículo se convierta en un complemento principal para la colectividad actual [3].

Desde el año 1996, los vehículos integran un mecanismo de diagnosis conocido como OBD II, el cual se encarga de detectar y guardar errores encontrados en la ECU que tiene el vehículo y facilitar la lectura de los datos mediante la interface llamada Diagnostic Link Cnnector (DLC) [3]. El objetivo principal de este sistema era el monitoreo de los sensores del automóvil, sin embargo, en la actualidad este sistema verifica que el desempeño y productividad del vehículo sean óptimos [4].

Haciendo uso del sistema OBD II se puede conocer lo que está sucediendo con el motor del vehículo a través de la lectura de sus valores, mismos que viajan hacia el cerebro del vehículo aportando de esta manera un seguimiento del motor y demás periféricos [1]. Con la introducción del sistema OBD II, el nicho automotriz ha diseñado variedad de escáneres portables y costosos permitiendo diagnosticar la funcionalidad del vehículo. Para sustituir estos sistemas de alto costo nace como una alternativa la utilización de escáneres OBD basados en el integrado ELM327. Además de las funciones de detección de fallas y lectura del estado del motor, este sistema en conjunto con otros dispositivos, puede ser utilizado para el monitoreo remoto de un vehículo.

Actualmente, la Agencia de Regulación y Control de las Telecomunicaciones (ARCOTEL), cuenta con un agente externo que le brinda el monitoreo de geolocalización, el cual no permite desarrollar una correcta supervisión y control completo de los automóviles que posee la empresa. ARCOTEL busca contar con sus programas de monitoreo propios sin necesidad de contratar servicios externos. Para evitar incurrir en gastos que pueden ser innecesarios, la tendencia está en desarrollar las herramientas en software libre. En este sentido, el contar con un software propio de la empresa con todo el código fuente abierto, les permite a los técnicos que forman parte de ARCOTEL enriquecer este recurso con nuevas prestaciones, cosa que no pueden hacer con un servicio contratado.

JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES)

Las empresas que tienen vehículos para brindar sus servicios, necesitan conocer la ubicación, el trayecto recorrido por día, estado del vehículo; es decir, necesitan tener un control vehicular que sea remoto para acceder desde su empresa, desde una oficina, o en cualquier lugar a través de un Smartphone con conexión a internet. Para ello recurren a contratar empresas que se dedican a brindar ese servicio, las cuales no brindan un servicio completo de monitoreo [4].

Una de estas empresas es ARCOTEL, que tiene un sistema de geolocalización contratado a una empresa externa. Este servicio sólo permite saber la ubicación del vehículo sin tener información sobre el estado del automotor. Por tal motivo, se pretende hacer un análisis de los datos obtenidos del sistema OBD II del vehículo y diseñar una aplicación móvil donde se pueda tener un mejor control y monitoreo completo del vehículo, es decir, se tendrá acceso a más datos de interés de la empresa sin tener que contratar servicios a parte. Los datos necesarios para ARCOTEL son: localización del vehículo en tiempo real, control del consumo de gasolina, y velocidad del vehículo. Además, una vez obtenidos los datos necesarios en la aplicación móvil se transmite y visualizará estos datos en un servidor web que estará al servicio de la empresa. Este control estricto le ayuda a la empresa a evitar que los vehículos sean usados con fines no laborales e incluso puedan incurrir en faltas legales que pueden acarrear problemas mayores.

De acuerdo con esto, el sistema embebido que se propone, está diseñado para funcionar en varias marcas de vehículos y se desarrolla en software libre, brindando la posibilidad de escalabilidad en el sistema. En esta primera etapa se desarrolla el proyecto de manera funcional, pero compone principalmente la base para futuros desarrollos que puedan favorecer a la empresa beneficiaria acorde a las necesidades que puedan presentarse en el futuro.

OBJETIVOS

OBJETIVO GENERAL.

- Desarrollar una aplicación móvil para la recolección, análisis y envío de datos de la Unidad de Control de Motor (ECU) a bordo de un vehículo para monitoreo remoto

OBJETIVOS ESPECÍFICO.

- Desarrollar el estado del arte de las principales características del sistema OBD II y el ELM327.
- Obtener y analizar los datos del sistema OBD II para el monitoreo de vehículos mediante el escáner de diagnóstico ELM327.
- Diseñar y desarrollar una aplicación móvil para el sistema operativo Android que permita visualizar y enviar los datos obtenidos de la ECU mediante el sistema OBD II.

CAPÍTULO 1:

1 FUNDAMENTACIÓN TEÓRICA

1.1 SISTEMA OBD.

La Sociedad de Ingenieros Automotrices (SAE, por sus siglas en inglés Society of Automotive Enginneers) desarrolló la primera etapa del estándar OBD a finales de los 80. On-Board Diagnostics (OBD) es un sistema computacional creado con el fin principalmente de controlar las emisiones al realizar un seguimiento del desempeño de los componentes principales como el motor, el convertidor catalítico, el filtro de partículas, el sensor de oxígeno, control de emisiones, el sistema de combustible y control de gases de escape. Un sistema OBD básicamente consta en una ECU, que toma las señales de entrada de varios sensores controlando los actuadores para tener el rendimiento que se desea [5][6].

1.1.1 ESTRUCTURA DEL OBD

La estructura del OBD en base a la aplicación y requisitos tiene sus variaciones, por lo general los factores que rigen la estructura del OBD son [5]:

a) Tipo de combustible (como diésel/gasolina)

b) Tamaño del vehículo o del motor

c) Región geográfica

d) Aplicación del motor (como en la marina, en la carretera o fuera de ella)

e) Modelo o año

f) Fase o programa OBD

g) Capacidad de diagnóstico

1.1.2 FUNCIÓN DEL OBD.

La función del OBD consiste en los siguientes pasos [5]:

1.Vigilancia continua

2. Indicación de fallos

3. Almacenamiento de fallos

4. Tomar las medidas adecuadas

1.2 OBD-I.

OBD-I se considera la primera versión del sistema OBD integrado implementado en 1991. El enfoque principal es monitorear solo los componentes responsables de controlar las emisiones producidas por el vehículo. Sin embargo, carecen de coherencia en la comunicación y la interfaz, lo que conduce a diferentes interpretaciones de diferentes marcas de automóviles [6]. El primer estándar OBD no era perfecto, la conexión entre el escáner y la ECU no fue estandarizado por lo que la herramienta de lectura no funcionaba en todos los vehículos; ya que los códigos de error no eran los mismos entre las diferentes marcas de fabricantes. Para resolver estos problemas, en 1996 se desarrolló un nuevo estándar llamado OBD II, el cual es más utilizado hoy en día [7].

1.3 OBD-II.

OBD-II (On-Board Diagnostics Second Generation) es la segunda y última versión del sistema OBD. Es una nueva reforma de la Agencia de Proteccion Ambiental (EPA, por sus siglas en inglés Environmental Protection Agency) para automóviles y camiones ligeros en 1996. Proporciona un conector estándar de 16 pines en todo el mundo. Su objetivo principal es detectar fallas químicas, mecánicas y eléctricas en el vehículo y reducir automáticamente los indicadores de emisiones. Permite la autocorrección de cualquier anomalía en la mezcla de combustible o el encendido del vehículo sin importar el tipo de automóvil [8].

1.3.1 PROTOCOLOS DE COMUNICACIÓN OBD-II.

Por el motivo de que las diferentes marcas automovilísticas poseen una única ECU, dentro de OBD II hay cinco protocolos que pueden ser usados para comunicarse con la ECU, cada uno de estos protocolos posee diferente velocidad de comunicación y nivel de voltaje [11]. Los protocolos son descritos a continuación en la Tabla 1.1.

PROTOCOLO	COMUNICACIÓN	VOLTAJE	VELOCIDAD TRANSMISIÓN	FABRICANTE
SAE J1850 VPN	Modulación de ancho de pulso variable (VPN)	2.2 V - 0L 8 V - 1L	10.4 - 41.6 Kbaud/s	General Motors
SAE J1850 PWM	Modulación de ancho de pulso (PWM)	0 - 5 V	41.6 Kbaud/s	Ford, Lincoln y Mercury
ISO 9141-2	Usa comunicación estándar RS-232	0 - 12 V	10.4 Kbaud/s	Fabricantes europeos, asiáticos, Chrysler, Jeep y Dodge
ISO 14230 KWP	Comunicación estándar RS-232	0 - 12 V	1.2 - 10.4 Kbaud/s	Fabricantes europeos y asiáticos
ISO 15765 CAN	Red de Área del controlador	2.5 - 5 V (CANH) 2.5 - 0 v (CANL)	250 - 500 Kbps	Compañía Bosch

Tabla 1.1. Características de cada protocolo.
Fuente: Obtenido en [11]

1.3.2 MENSAJE EN OBD-II.

Este mensaje contiene una cierta cantidad de bytes en los que se encuentran el encabezado de identifica y los bytes de datos. Hay dos estructuras de mensaje [4] [12].

1.3.2.1 Estructura de mensaje de protocolos SAE J1850 e ISO.

Ambos protocolos, tanto como SAE J1850 e ISO se basan en una misma estructura, estos protocolos se conforman por tres bytes de encabezado al iniciar la trama, un límite de siete bytes de información y dos bytes para control de errores al finalizar la trama, organizados de la siguiente manera. [4] [12].

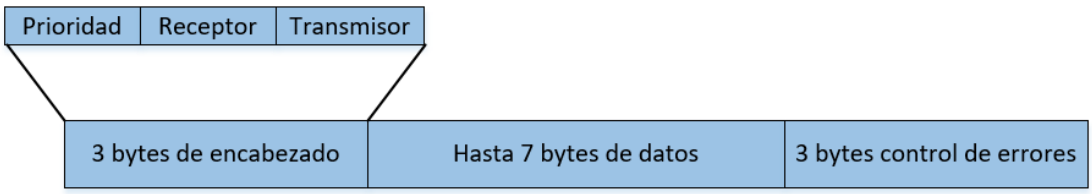


Figura 1.1 Mensaje protocolos SAE J1850 e ISO
Fuente: Obtenido en [12]

Cada uno de los parámetros que se muestran en la figura 1.1 tiene una función principal que se detalla a continuación:

- Prioridad. Contiene la relevancia del mensaje.
- Receptor. Contiene la dirección hacia dónde va el mensaje.
- Transmisor. Dirección del origen.
- Datos. Contenido del mensaje, datos o solicitudes.
- Control de errores. Controla que no haya errores introducidos durante la transmisión del mensaje y si lo detecta realiza una petición de retransmitir el mensaje [12].

1.3.2.2 Estructura de mensajes de protocolo CAN.

Contiene una estructura de mensaje similar al anterior, con la única diferencia de la forma en la cual se compone el encabezado. En el protocolo CAN la cabecera posee bits de identificación y dependiendo de la norma CAN, la cantidad de bits de identificación puede ser de once o veinte y nueve. En la figura 1.2 se observa cómo está distribuida la estructura del mensaje en el protocolo CAN [4] [12].

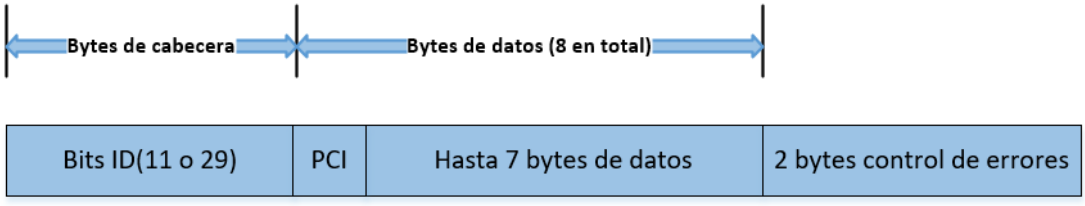


Figura 1.2. Mensaje protocolo CAN.
Fuente: Obtenido en [12].

La función de cada bloque de bytes es:

- Bits de identificación. Contiene información del tipo de datos, la prioridad, receptor y transmisor.
- PCI. Contiene el tamaño de bytes que tiene la trama. Usualmente 7 bytes.

- Datos y control de errores. Realizan la misma función del formato de mensaje de los protocolos ISO y SAE J1850 [12].

1.3.3 MODOS DE MEDICIÓN.

OBD-II mide nueve modos diferentes, según el modo escogido, ciertos tipos de información se pueden recuperar a través de la unidad de control del vehículo. Cada modo hace uso de un parámetro de identificación conocido como PID, el cual hace la solicitud de información del automóvil y fue creado específicamente para comunicarse con el escáner del automóvil. Los nueve modos son descritos a continuación [8] [11].

- Modo 1. Datos actualizados

Accede a los datos de los sensores en tiempo real como, por ejemplo, temperatura, voltajes, RPM del motor, entre otros [8].

- Modo 2. Datos guardados

El modo 2 accede a los datos guardados en la memoria de la computadora del vehículo que se relaciona con alguna avería del vehículo [8].

- Modo 3. Códigos de fallas

La información se extrae de los Códigos de error de datos (DTC, por sus siglas en inglés Data Trouble Codes) en la ECU, estos datos se originan en la lectura de un fallo por el motivo de la lectura del valor de un sensor que está fuera de rango predeterminado [8].

- Modo 4. Eliminar códigos de falla y datos almacenados

Envía un comando a la ECU para borrar todos los DTCs y apagar la lámpara indicadora de mal funcionamiento (MIL) si está encendido [8].

- Modo 5. Resultados de la prueba de monitorización del sensor de oxígeno
- Modo 6. Resultados de prueba de otros sensores

En este modo están guardados los datos de pruebas realizados al sistema de monitoreo no continuo.

- Modo 7. Códigos de falla pendientes

Se accede a los DTC que no han sido arreglados [8].

- Modo 8. Control del sistema de a bordo

En este modo se realiza pruebas en los actuadores que se active o desactive.

- Modo 9. Información del vehículo

Accede al número de identificación del vehículo (VIN, por sus siglas en inglés Vehicle Identificacion Number), el cual es un único valor de identificación del vehículo en todo el mundo [8].

1.3.4 COMPONENTES DE OBD II.

El sistema OBD monitoriza continuamente el sistema de control de emisiones de un vehículo para su correcto funcionamiento. Hay dos tipos de supervisión, la supervisión de componentes y la supervisión del sistema. En la detección de fallos, la lámpara MIL que está montada en el tablero se ilumina para indicar el fallo. El almacenamiento de fallas se hace guardando la información sobre la falla en forma de Código de Diagnóstico de Problemas (DTC) estructurado en la memoria de la ECU [7].

Para evaluar esta información útil, los técnicos necesitan un dispositivo llamado herramienta de escaneo. La herramienta de escaneo se comunica de manera efectiva con el sistema OBD del vehículo y recibe información sobre el funcionamiento del sistema.

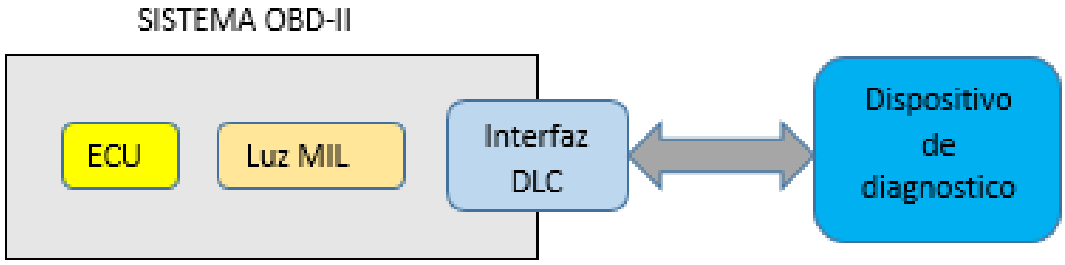


Figura 1.3. Componentes OBD-II.
Fuente: Autores.

1.3.4.1 ECU.

La ECU (Unidad de Control de Motor) del motor es el cerebro del motor del automóvil, consiste en un grupo de dispositivos electrónicos colocados en una placa de circuito impreso, el dispositivo electrónico está ubicado en una cubierta de aluminio, y está equipado con un disipador de calor para disipar el calor generado, como se muestra en la Figura 1. Varias señales de control llegan a la unidad de control

electrónico. Se evalúa la entrada del sensor que indica el funcionamiento del motor y se calcula el tiempo de activación del elemento actuador a partir de la señal de salida [15].

1.3.4.2 Luz MIL.

La luz MIL es un componente del sistema OBD-II y se activa cuando el automóvil tiene un mal funcionamiento; su propósito es avisar al conductor que debe ser revisado [4]. La luz MIL tiene 2 tipos de alarmas; la primera cuando la luz indicadora está siempre activada, indicando que el automóvil debe ser inspeccionado de inmediato, y la otra es cuando la luz indicadora parpadea, lo que significa que el automóvil debe ser inspeccionado en el término corto [3].

1.3.4.3 Interfaz DLC.

OBD-II posee un puerto de diagnóstico o DLC (conector de enlace de diagnóstico) que tiene forma de trapezoide y sirve como una interfaz de acceso y para recuperar la información de la ECU a los dispositivos de diagnóstico. Los enchufes o conectores tipo OBD-II tienen 16 pines, pero hay que recalcar que no se utilizan todos. De hecho, estrictamente hablando, solo uno o dos de los dieciséis pines se utilizan para transmitir datos. En la mayoría de casos los pines 4 y 16 son utilizados para energizar o alimentar al escáner; excluyendo los pines número 4, 5 y 16, se considera al resto de pines para diagnóstico [8] [9].

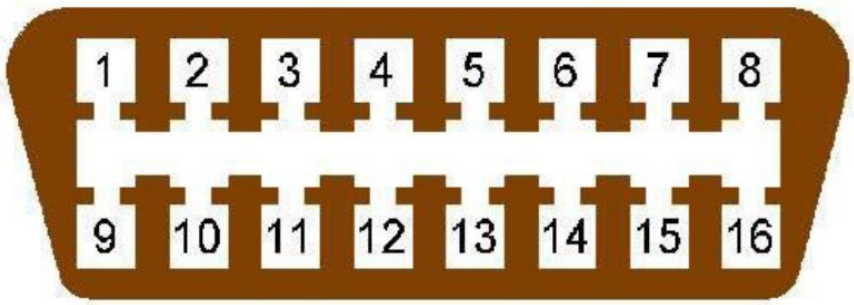


Figura 1.4. Pines OBD-II.
Fuente: Obtenido en [9].

La tabla 1 muestra el número de pin y a su vez las funciones que realiza cada uno de los pines.

N° Pin	FUNCION
1	Sin uso
2	J1850 Bus positivo y PWM
3	Sin uso
4	Tierra del vehículo
5	Tierra (señal)
6	CAN Alto
7	ISO 9141-2 Línea K
8	Sin uso u opcional
9	Sin uso u opcional
10	J1850 Bus negativo
11	Sin uso u opcional
12	Sin uso u opcional
13	Sin uso o tierra de la señal
14	Bus de datos CAN bajo
15	ISO 9141-2 Línea L
16	Batería-positivo

Tabla 1.2. Funciones de pines del conector DLC.
Fuente: Obtenido en [11].

Cabe señalar que un tipo de vehículo sólo implementa un protocolo. La diferencia más significativa de los cinco protocolos anteriores es la colocación de pines para la comunicación. La figura 1.5 muestra que pines del conector utiliza cada protocolo para la comunicación.

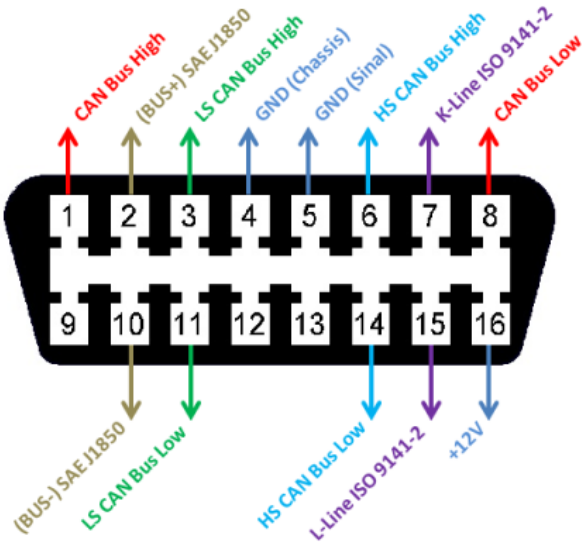


Figura 1.5. Pines usados por cada protocolo.
Fuente: Obtenido en [10].

Ubicación del puerto DLC en el automóvil.

El conector en ocasiones está cubierto o tapado por protector negro. Las letras OBD están impresas en este protector, en otros casos puede que no se muestre, pero la forma del protector nos dice que es el conector [16].

A continuación, en la imagen se muestra de forma rápida las ubicaciones del conector DLC.

Figura 1.6. Ubicaciones del conector DLC.
Fuente: Obtenido en [16].

- Reposapiés del conductor parte superior o incluso parte superior derecha y también a lado izquierdo.
- Debajo de la columna de dirección.
- En varios modelos en la parte izquierda del volante muy similar a una guantera pequeña.
- Quitando la protección del volante, se puede confundir con el protector donde se pone la llave.
- Reposapiés del copiloto.
- Puede ubicarse debajo de los asientos (piloto o copiloto), se puede encontrar por una tapa con las siglas impresas OBD.
- En la parte central, delante del freno de mano [16].

Las ubicaciones del conector pueden ser diferentes esto depende del fabricante del vehículo y de manera muy general el puerto DLC está ubicado en los reposapiés del conductor, debajo del volante o incluso en el asiento del copiloto [16].

1.3.4.4 Dispositivo de diagnóstico.

a Escáner de diagnóstico profesional.

Estos dispositivos tienen un precio elevado que ronda entre $1500 a $2000, por lo que son usados por personas capacitadas del servicio automotriz, son equipos para diagnosis calificado y algunos específicamente diseñados para cada fabricante [11].

Figura 1.7. Escáner profesional.

Fuente: Obtenido en [11].

b Escáner con conexión USB.

En la actualidad se han desarrollado algunos conectores que logran obtener los datos de la ECU y pueden ser transferidos a la computadora mediante el puerto serial o USB para luego analizarlos, este conector necesita de un software especial. Una cosa a tomar en cuenta, su precio no es elevado, el precio oscila entre los 29,95 dólares y los 99,95 dólares [11] [13].

Figura 1.8. Escáner con puerto USB.
Fuente: Obtenido en [13].

c Dispositivos de acceso WiFi o Bluetooth.

En el presente se puede observar la variedad de lectores de códigos OBD II que se conectan de manera inalámbrica vía WiFi o Bluetooth. Debido a su bajo costo estos dispositivos son muy comerciales y existen aplicaciones que pueden ser instaladas en un Smartphone o Tablet [11].

1.4 LECTOR DE CÓDIGOS ELM327.

Actualmente, según normativa, el 100% de los vehículos ofrecen realizar una diagnosis mediante escáneres a un bajo costo, ya sea un escáner de coche o un prototipo de diagnóstico de vehículo basado en código abierto. La información transmitida se sustenta en distintos protocolos, algo a mencionar es que, si se establece una comunicación directa entre el vehículo y el ordenador o con cualquier dispositivo inteligente, estos no nos permitirán acceder a la información [4].

El lector de fallas ELM327 es un microcontrolador fabricado por ELM Electronics que permite obtener la información de la ECU y luego enviar a un elemento de recepción mediante comunicación serial, bluetooth, USB o WI-FI [4].

Está diseñado para ser el intérprete en la comunicación Bluetooth entre el OBD-II y un Smartphone Android que viene a ser el receptor de los datos. Logra reconocer de manera automática los nueve protocolos del sistema OBD. Soporta comunicaciones de alta velocidad y se puede adaptar completamente a los requisitos exigidos por el programador [4].

1.4.1 CARACTERÍSTICAS PRINCIPALES DEL ELM327.

- Opera como un escáner automotriz, se necesita de un dispositivo de recepción para la decodificación de los datos de la ECU.
- Interfaz de comunicación serial, Bluetooth, WiFi
- Se configura en su totalidad con comandos AT
- Soporta nueve modos de medición.
- Está compuesto por un circuito integrado ELM327 del fabricante ELM Electronics, en el anexo se adjunta el datasheet [4].

Figura 1.9. ELM327.
Fuente: Obtenido en [4].

CAPÍTULO 2:

2 MARCO METODOLÓGICO

La solución para alcanzar los objetivos propuestos se basa en dos subsistemas, el uno de adquisición y análisis de datos; y el otro comprende la visualización y envío de datos para almacenamiento en la nube mediante una aplicación móvil. El subsistema de adquisición permite conectarse a la computadora del automóvil, leer los parámetros del mismo para posteriormente ser procesados como se observa en la figura 2.1.

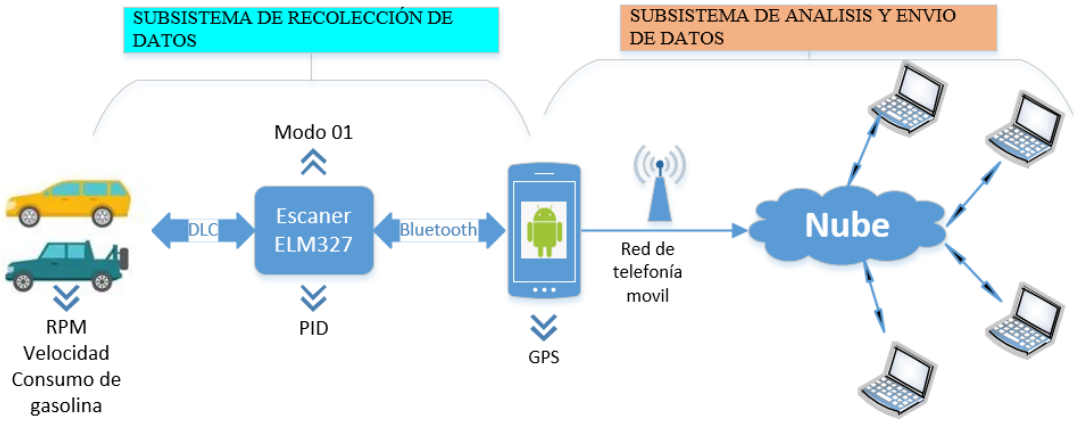


Figura 2.1. Topología del sistema.
Fuente: Autores.

El sistema completo comprende los siguientes requerimientos, teniendo en base las exigencias y las cualidades de los vehículos. Se tiene el procedimiento:

- Puede ser utilizado en la mayoría de marcas de automóviles que posean el puerto DLC y soporte diferentes protocolos del sistema OBD II.
- Funcionará con la mayoría de los comandos OBD o también llamados PIDs del modo 1, lo que significa que se obtendrán los parámetros en tiempo real, los PIDs involucrados son los que nos dan un resultado sobre: RPM, velocidad del vehículo, flujo de masa de aire, consumo de gasolina.

- Utilizará la red de telefonía móvil para la transferencia de los datos hacia la base de datos.
- Aprovechando de una de las funcionalidades del Smartphone, hará uso del GPS para obtener las coordenadas de posición y calcular la distancia recorrida.
- Guardará los datos en la nube.

Con lo dicho anteriormente, este capítulo detalla la metodología que se realiza para acceder a la computadora del vehículo y adquirir los datos o parámetros del automóvil, el sistema de adquisición constará de un escáner de diagnóstico inalámbrico para la conexión entre el vehículo y el Smartphone, y también permitirá el acceso y obtención de datos.

2.1 COMUNICACIÓN CON EL AUTOMÓVIL.

Para realizar la comunicación con el vehículo se utiliza uno de los dispositivos de escaneo que funciona de forma inalámbrica, en este caso trabajamos con el dispositivo ELM327 con conexión Bluetooth debido a la facilidad y comodidad del usuario sin la necesidad de cables que molesten al conductor. El integrado ELM327 nos ayudará como sistema para la conexión, acceso y obtención de los datos de la ECU del automóvil, para ser enviados a través de bluetooth a un dispositivo móvil [17].

Figura 2.2. Dispositivo utilizado en el proyecto.
Fuente: Obtenido en [17].

El dispositivo se puede configurar mediante comandos AT y para acceder a los datos del vehículo se realiza mediante el envío de comandos OBD, existen diversos comandos que se utilizan para la configuración y lectura de los parámetros, a continuación, se detalla los comandos más importantes y también el procedimiento a seguir para hacer uso de los comandos [17].

2.1.1 COMANDOS AT.

Muchos parámetros en ELM327 se pueden ajustar para modificar su respuesta, aunque se haya establecido la conexión con el vehículo, no es necesario modificar estos parámetros. [4] Los comandos AT usualmente empieza con las letras “A” o “T”. el formato de mensaje de un comando AT es el siguiente:

AT	COMANDO	[DATOS ESPECIFICOS DEL COMANDO]

Figura 2.3 Formato de mensaje de comando AT.
Fuente: Obtenido en [4].

En la tabla 2.1 que se muestra a continuación se detalla algunos de los códigos AT más importantes.

AT <CR>	Repite el comando que se envió al último.
AT Z	Resetear.
AT E0/ AT E1	Modo eco activado, desactivado.
AT RV	Leer voltaje.
AT DP	Devuelve el tipo de protocolo.
AT DPH	Descripción de protocolo utilizado.
AT SP h	Grabar el protocolo h y usarlo
AT SP Ah	Buscar automáticamente el protocolo basado en el número h
AT TP Ah	Busca comunicarse mediante protocolo h
AT D	Devuelve las configuraciones a defecto
AT BRD hh	Divide la velocidad de baudios
AT BRT hh	Fijar el tiempo de espera de la tasa de baudios

Tabla 2.1. Principales comandos AT.
Fuente: Obtenido en [4].

2.1.2 COMANDOS OBD.

Cuando los comandos que son enviados al escáner empiezan diferente de la letra “A” ó “T”, estos toman el nombre de comandos OBD. Los comandos del OBD son enviados al vehículo integrado en un paquete de datos, cada paquete es comprobado para asegurar que se envíen solo dígitos hexadecimales válidos. La mayoría de los estándares requieren que se incluyan 3 bytes de encabezamiento y un byte de control de errores para cada comando OBD [4]. Los valores por defecto de los bytes agregados suelen ser compatibles de la mayoría de las peticiones [14] [4].

La longitud de los bytes se verá limitado por el ELM327 esto es debido a las normas establecidas normalmente suelen ser 7 bytes o también pueden estar formados por 14 caracteres hexadecimales. Cuando el número de bytes que se intentó enviar excede el número de bytes especificado en la norma, se producirá un error y el comando enviado se ignorará, la respuesta es desconocida y se mostrará un signo de interrogación. El formato de un mensaje de comando OBD se muestra a continuación [14].

ID DE SERVICIO DE OBD-II	[DATOS ESPECIFICOS DEL SERVICIO]

Figura 2.4 Formato de mensaje de comando OBD.
Fuente: Obtenido en [14].

Los hexadecimales comúnmente se utilizan para el intercambio y comunicación con el ELM327 debido a que es el formato de datos más usado por las normas de OBD. La mayoría de las solicitudes hacen uso de la simbología hexadecimal al momento de presentar los resultados, siendo esta la estructura más utilizada. No se puede evitar tratar con los dígitos hexadecimales, eventualmente todos los usuarios necesitan procesar los datos de alguna forma dependiendo del parámetro deseado [14].

Número hexadecimal	Número decimal
0	0
1	1
2	2
3	3
4	4

5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Tabla 2.2. Tabla de conversión hexadecimal a decimal.
Fuente: Obtenido en [12].

Suponga que se envía un comando, como ejemplo A6 (166 en decimal). Estos tres caracteres se enviarán a ELM327 a través del socket Bluetooth [18]. ELM327 almacenará los caracteres cuando los reciba, y cuando reciba el tercer carácter, comenzará a evaluar los otros dos caracteres, verá que son válidos todos los números hexadecimales y los transformará a un byte. Luego, se agregarán el byte de encabezado y el byte de suma de verificación, por lo general, se envía un total de cinco bytes al vehículo [14] [4].

Una vez enviado el comando, el ELM327 espera respuestas y busca una que esté con destino hacia él. Cuando se tiene una coincidencia en un mensaje, los bytes receptados se envían al cliente a través del Bluetooth, mientras que los mensajes recibidos sin una dirección coincidente se ignoran. ELM327 continuará esperando los mensajes que se le envíen hasta que no se encuentren mensajes dentro del tiempo establecido de espera. Tras continuar recibiendo información, el escáner sigue reiniciando este temporizador y buscando más contenido. Además, el circuito integrado constantemente tendrá una respuesta para cada solicitud, inclusive si dice "NO DATA" (es decir que no se encontró ninguna información o se encontraron algunos mensajes, pero no cumplieron con las condiciones de recepción) [4] [14].

2.2 ACCESO A LOS DATOS DEL VEHÍCULO A TRAVÉS DEL LECTOR ELM327

La normativa exige ante una petición OBD enviada a un automóvil debe adherirse a una estructura prestablecida. El byte inicial que se envía, llamado “modo”, nos dice que información es solicitada, en cambio el siguiente byte, pueden ser tres o incluso más, es la información relevante. Los otros bytes que están a continuación al

byte “modo” son conocidos como "bytes de identificación de parámetros" o número PID. Los vehículos no están obligados a soportar los diferentes modos, y, además, no todos los PID están en la obligación a ser soportados dentro de los modos, en el caso de los automóviles que implementaron por primera vez el sistema OBD II únicamente soportaban una cantidad pequeña de estos. En los modos, el PID 00 solo es usado para indicar cuales son los PID soportados en ese modo. El modo 01 funciona en todas las marcas de automóviles [18].

Para acceder a este modo primeramente se debe asegurar que el escáner está conectado correctamente al automóvil. El ELM327 no responderá si la llave del automóvil no se encuentra en la posición de encendido, pero no arrancar el motor. Si no se consigue que el ELM327 sea reconocido, se debe reiniciarlo enviando el siguiente comando [9] [12]:

“AT Z”

Las luces leds empezaran a titilar y debería responder con ELM327 y la versión, seguido de un carácter de aviso. A continuación, se podrá establecer el protocolo para comunicarse con el vehículo, es más fácil seleccionar simplemente el protocolo “0”: “AT SP 0” [12][18]

Una vez preparado al ELM327 para comunicarse con un vehículo. Se envía el comando modo 01 PID 00 [9][12]:

“01 00”

El ELM327 debería decir que está “BUSCANDO…”, es decir que está en el proceso de búsqueda de un protocolo para entablar una comunicación y se imprimirá lo siguiente o algo similar [9][12]:

41 00 BE 1F B8 10

Ahora vamos a explicar que significa cada byte de la respuesta anterior, comenzando por el “41” el cual significa una respuesta del modo 01 (01 + 41 = 41), en cambio el segundo número “00” es el PID requerido. En el caso que sea el modo 02, la respuesta será “42”, un modo 03 se tendrá “43”, y así sucesivamente. Los otros bytes que restan “BE, 1F, B8 y 10” son datos requeridos [12].

Un ejemplo es la temperatura actual del motor. Para solicitar es sencillo se debe enviar el PID 05 del modo 01, de la siguiente forma [12]:

“01 05”

Una respuesta sería:

“41 05 7B”

Como se indicó anteriormente el “41 05” nos quiere decir que se respondió a una solicitud en el modo 01 con el PID 05, el 7B corresponde a la información enviada desde la ECU que es presentado en hexadecimal, al convertir a decimal el resultado obtenido es 123 [9][12]. Este valor es la temperatura expresada en grados centígrados con desplazamiento del cero. Para obtener la temperatura actual, la fórmula nos indica que el valor recibido se le restará 40 y el resultado es de 83°C [9][12].

Finalmente, como último ejemplo para hacer una solicitud de las revoluciones del motor. Dentro del modo 01 con el PID 0C, siendo así [12]:

“01 0C”

Se tendría una respuesta como:

“41 0C 1A F8”

Se tiene como resultado “1A F8” representado en hexadecimal por lo cual debe ser transformado a decimal para interpretar el dato obtenido. Convirtiéndolo, dando como resultado 6908, este valor puede ser un valor elevado del rpm [12].

El valor obtenido del rpm es elevado debido a que son enviados en aumento de un cuarto de las revoluciones. Al dividir 6908 por 4 se obtiene el valor real de las revoluciones por minuto siendo 1727 rpm [12].

En los ejemplos anteriores se realizaba la petición de los parámetros del automóvil directamente sin conocer que protocolo era compatible con el automóvil. Debido a que el escáner hace todo el trabajo del formato y la traducción de los datos, aunque el usuario busque configuraciones especiales, sin embargo, el escáner maneja nueve protocolos y basta que el vehículo utilice alguno de estos [9].

Hay veces en la que no se obtiene una respuesta de una línea como se mostró en los ejemplos anteriores, en varios casos se recibe respuestas de varias líneas de información separadas. El escáner posee un tiempo por defecto de espera para comprobar si hay más líneas de información, cuando se conoce el número de respuestas que van a ser enviadas. Se puede aumentar el tiempo en el cual se recupera la información, diciéndole al escáner el número de líneas de información que se espera receptar, una vez configurado eso el escáner sabe cuándo terminar ignorando el tiempo por defecto esperando datos que no llega [9]. Simplemente hay que añadir un solo dígito hexadecimal proporcionando el máximo número de respuestas a obtener, y el ELM327 se encargará de hacer el proceso. Como ejemplo, al saber que se espera una sola respuesta al solicitar la temperatura, se envía [9][12]:

"01 05 1"

Después de leer la solicitud enviada por el dispositivo móvil, el escáner responderá de inmediato luego de recibir una línea de respuesta logrando ahorrar tiempo, ignorando el tiempo impuesto por defecto de 200 ms, si no se tiene una respuesta el escáner continua ajustando el tiempo de espera después de realizar la petición [9][12].

En el caso de unos protocolos, como ejemplo el protocolo "J1850 PWM" es necesario un acuse de recibo generado por el escáner para cada conjunto de información que se envía. Cuando la cantidad de respuestas proporcionada es muy pequeña, el escáner responderá de manera temprana, y debido a esto generar un colapso en el canal por lo tanto la computadora del vehículo intenta reenviar varias veces mensajes desconocidos. Por ende, se necesita saber el número de respuestas que se esperan para hacer uso de esta característica [9][12].

2.2.1 SELECCIÓN DE PROTOCOLOS.

El ELM327 soporta diferentes protocolos como se mostró en la tabla 2.3. Al momento de utilizarlo, como se mencionó anteriormente es posible que nunca se tenga que elegir el protocolo, ya que la configuración de fábrica hace que se realice una búsqueda automática, sin embargo, puede que se desee especificar un protocolo a utilizar [9][12].

Protocolo	Descripción
0	Automático
1	SAE J1850 PWM (41.6kbaud)
2	SAE J1850 VPW (10.4 kbaud)
3	ISO 9241-2 (5 baud init)
4	ISO 14230-4 KWP (5 baud init)
5	ISO 14230-4 KWP (fast init)
6	ISO 15765-4 CAB (11-bit ID, 500 kbaud)
7	ISO 15765-4 CAB (29-bit ID, 500 kbaud)
8	ISO 15765-4 CAB (11-bit ID, 250 kbaud)
9	ISO 15765-4 CAB (29-bit ID, 25 kbaud)
A	SAE J1939 CAN (29-bit ID, 2580 kbaud)
B	User1 CAN (11-bit ID, 125 kbaud)
C	User2 CAN (11-bit ID, 50 kbaud)

Figura 2.4. Protocolos usados por el ELM327.
Fuente: Obtenido en [12].

Cuando se conoce que protocolo posee el automóvil, por ejemplo, si un vehículo utiliza ISO 9141-2, se ordena al ELM327 hacer uso exclusivo de ese protocolo. Para hacer esto, simplemente se determina el número de protocolo que se muestra en la tabla, y luego usar el comando AT "Set Protocol" de la siguiente manera [12] [19]:

- AT SP 3

  OK

Una vez configurado el protocolo, cada vez que se encienda se regirá al protcolo 3 o el que se configure. Se puede verificar esto pidiendo al ELM327 que describa el protocolo [12][19]:

- AT DP

  SAE J1850 VPW

Anteriormente el ELM327 fue configurado para usar un protocolo en específico, pero si se conecta en otro vehículo el cual usa el protocolo ISO 14230-4 KWP. Cuando se conecte en el otro vehículo se tiene la opción de realizar un cambio para que dispositivo realice una búsqueda de manera automática del protocolo. Para configurarlo se inserta una "A" antecediendo al número del protocolo, como se tiene a continuación [12][19]:

- AT SP A2

  OK

- AT DP

  AUTO, SAE J1850 VPW

El ELM327 intentará usar el protocolo 2, si la conexión con este, entonces se iniciará una búsqueda automática para intentar conectarse con otro protocolo [12][19].

Los comandos "AT Set Protocol" escriben sobre la memoria EEPROM del escáner. Realizar la escritura toma una gran cantidad de tiempo y repercute en la configuración de la petición que continua y puede que no sea apropiada cuando el protocolo que se selecciona no es el indicado para el automóvil en uso. Se puede realizar un test de protocolo antes de realizar la escritura utilizando el comando de prueba [12][19].

El protocolo de prueba es muy similar al comando de "Selección de Protocolo". Se utiliza igual que el comando "AT SP", solo diferenciándose que se escribe en la memoria luego de realizar la conexión con el protocolo correcto y también que este activada la función. En base el ejemplo mostrado, lo que se envía es [12][19]:

- AT TP A2

  OK

Es muy complejo intentar usar un protocolo para conectarse a la ECU del vehículo. En estos casos, la mejor opción es que el ELM327 decida cual usar. Esto se

hace de manera sencilla basta con decirle que use el protocolo número 0 (con los comandos “SP” o “TP”) [12][19].

Si necesita que el escáner realice una búsqueda de protocolos de manera automática, se hace una petición con el siguiente comando [12][19]:

- AT SP 0

  OK

El ELM327 buscará de forma automática uno que responda. Se mostrará un mensaje como el siguiente "BUSCANDO...", seguido de una respuesta, después se pregunta al ELM327 cual protocolo dio respuesta solicitando con el comando “AT DP” [12][19].

Los escáneres anteriores, es decir, las primeras versiones, realizaban una búsqueda ordenada recomendada por SAE, eso ya no se tiene en los últimos escáneres que se han desarrollado, en cambio realizan la búsqueda en función de las solicitudes activas que se presentan [12][19].

En el sistema OBD II se puede realizar una búsqueda de manera automática, y si desea experimentar, no es necesario implementarlo. Durante el proceso de búsqueda, ELM327 ignorará cualquier encabezado que haya definido previamente utilizará el encabezamiento OBD predeterminado. Durante la búsqueda, se utilizará el indicador estándar (es decir, 01 00). Si esto no es lo que desea, los resultados pueden ser frustrantes [12][19].

Cuando intente conectarse a una ECU utilizando valores propios de encabezado, no ordenar al ELM327 que use protocolo 0, en cambio ordenar que use el protocolo deseado, es decir, AT SP n o bien se puede usar la búsqueda automática en caso de suceder un fallo al utilizar el protocolo que se le ordenó con el comando AT SP An. [19] Luego, enviar la solicitud con los encabezados y datos que se necesiten, en este proceso el ELM327 procederá a conectarse usando los encabezados y datos que requiera el usuario, y solo si falla el intento de conectarse y se ha conectado utilizando el comando “AT SP An” el escáner utilizará los datos de fábrica del OBD [19].

2.2.2 CÓDIGOS DE FALLA.

Uno de los usos que se le da al ELM327 es obtener los códigos de falla o “DTC”. Para acceder a estos códigos se realiza la petición en el modo 3, teniendo en cuenta que es necesario conocer el número de códigos de fallas que se han guardado en la memoria. Para solicitar el número de códigos se envía una petición con el PID 01 en el modo [9][12]:

- 01 01

  41 01 81 07 65 04

El 41 01 no se toma en cuenta debido a que es una respuesta común como se ha visto anteriormente, y el 81 que corresponde a los datos, es la cantidad de fallas actuales [9]. Pero es ilógico ese número de códigos de falla 81 en hexadecimal o 129 en decimal existentes cuando el automóvil funciona. En otras palabras, el de datos hace tiene dos funciones, y el bit más significativo informa que se activó la luz MIL por efecto de una de las fallas leídas, en cambio, los otros siete bits restantes indican la cantidad de errores guardados [9][12]. Si se quiere obtener la cantidad real de fallas cuando la luz indicadora de fallo está encendida, simplemente se tiene que restar 128 u 80 en hexadecimal [9].

La respuesta anterior nos quiere decir entonces que hay un solo código de fallo existente, y que fue el motivo por el cual se encendió la luz indicadora del motor de control o MIL. Los otros bytes que se recibieron nos indican que pruebas soporta esta función [9] [12].

Se observa que en el ejemplo se obtiene una respuesta con una línea, pero cuando haya códigos almacenados de otras direcciones, se mostrara una respuesta con una línea para cada módulo. Si se desea conocer que modulo realizo el envió del código de error, se activa los encabezados “AT H1” y observar el byte número tres del encabezado para conocer que modulo realizo el envió de esos datos [9][12].

Si ya se conoce cuantos códigos están guardados, a continuación, se realiza la petición dentro del modo 3 sin la necesidad de un PID [9][12].

- 03

  43 01 33 00 00 00 00

El “43” hace referencia a que se está trabajando en el modo 3, los otros bytes restantes se leen de par en par y así obtener los códigos de error. Se observa que se a rellenado con ceros a la respuesta como recomienda la norma SAE, los ceros que se han agregado no se toman en cuenta es decir no son errores [9][12].

De la misma forma al solicitar la cantidad de errores guardados, la respuesta contiene los errores como también información adicional que son los bits más significativos. Se recomienda utilizar la siguiente figura en la decodificación de los bits adicionales en los primeros dígitos, como se muestra [18]:

If the first hex digit received is this,	Replace it with these two characters	
0	P0	Powertrain Codes - SAE defined
1	P1	“ “ - manufacturer defined
2	P2	“ “ - SAE defined
3	P3	“ “ - jointly defined
4	C0	Chassis Codes - SAE defined
5	C1	“ “ - manufacturer defined
6	C2	“ “ - manufacturer defined
7	C3	“ “ - reserved for future
8	B0	Body Codes - SAE defined
9	B1	“ “ - manufacturer defined
A	B2	“ “ - manufacturer defined
B	B3	“ “ - reserved for future
C	U0	Network Codes - SAE defined
D	U1	“ “ - manufacturer defined
E	U2	“ “ - manufacturer defined
F	U3	“ “ - reserved for future

Figura 2.5 Código de bits adicionales.
Fuente: Obtenido en [18].

2.2.3 RESTABLECIMIENTO DE LOS CÓDIGOS DE FALLA.

Otra de las funciones que trae el ELM327 es de reiniciar los códigos de falla, con tan solo enviar una petición en el modo 4. Sin embargo, antes de enviarlo hay que tomar en cuenta los efectos que causan, ya que realizara un reinicio completo y no solo de la luz indicadora del motor. De hecho, enviar una petición en este modo provocara que [19]:

- Restablece la cantidad de DTCs.
- Borra cualquier DTC de problemas de diagnóstico
- Borra cualquier información almacenada
- Borra el DTC que inició el marco de congelación
- Borra la información de los modos 6, 7

- Mantiene los códigos de falla que están en estado activo los cuales son reseteados solamente por la computadora del vehiculo [19].

El borrado de todos los códigos de falla no es exclusivo del escaner ELM327, sino que se realiza cuando se utiliza un escaner para restablecer los errores. La pérdida de estos datos tiene como consecuencia el mal funcionamiento del vehículo durante un periodo de tiempo, mientras realiza una re calibración [19].

CAPÍTULO 3:

3 IMPLEMENTACIÓN

A lo largo de este capítulo se detalla y explica el diseño e implementación para la recolección, análisis y envío de datos de la ECU para monitoreo a través de una aplicación móvil, tomando en cuenta los dispositivos y programas que se utilizaron.

3.1 RECOLECCIÓN, ANÁLISIS Y ENVÍO DE DATOS.

El sistema completo está compuesto de dos subsistemas. El subsistema de recolección de datos y el subsistema de análisis y envío de datos.

3.1.1 SUBSISTEMA DE RECOLECCIÓN DE DATOS

El desarrollo del proyecto empieza con la obtención de los datos de la ECU mediante el escáner de diagnóstico ELM327, el cual ya ha sido escogido para el desarrollo del proyecto. Las funciones y características ya fueron abarcadas en el capítulo 2.

Como otra parte del hardware se hace uso de un dispositivo móvil con sistema operativo Android para la visualización de los datos de la ECU.

3.1.2 SUBSISTEMA DE ANÁLISIS Y ENVÍO DE DATOS.

Este subsistema este compuesto por todo el software utilizado para diseñar la aplicación que comprende, analizar los datos y realizar el envío de los mismos.

3.1.2.1 Entorno de programación.

El entorno de programación seleccionado es Android Studio, se sustenta en IntelliJ IDEA. Cuenta un editor de códigos robusto, Android Studio además tiene más funciones para mejorar la productividad al desarrollar aplicaciones para sistemas operativos Android [20]:

- Cuenta con una complicación sustentado en gradle
- Tiene un emulador con bastantes funciones
- El entorno es unificado para dispositivos con sistema operativo Android
- Gran variedad de herramientas
- Cuenta con la integración de GitHub que ayudan a las funciones de las aplicaciones a compilarse y facilitan la importación de código abierto.
- Compatibilidad con las herramientas que ofrece Firebase. [20].

a Estructura de un proyecto

Generalmente la estructura de un proyecto en Android Studio, está formada por varios módulos que contienen archivos con código fuente y otros recursos. [21]:

- Paquete de modulo app.
- Paquete de modulo para biblioteca
- Paquete de modulo para Google App [21].

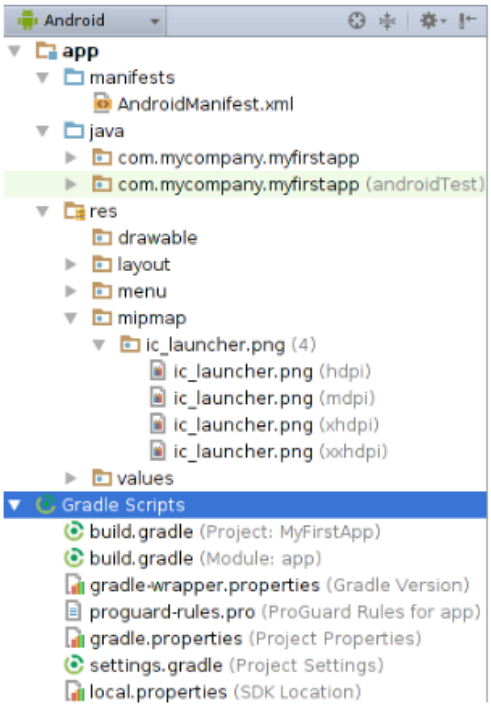


Figura 3.1 Vista de los archivos de un proyecto de Android.
Fuente: Obtenido en [21].

Por defecto, Android Studio estructura los archivos de los proyectos como se observa en la ilustración 3.1. Esta interfaz está estructurada por diferentes módulos que dan acceso de los archivos fuente [21].

Los módulos del proyecto son:

- manifests: Tiene el archivo AndroidManifest.xml, mismo que muestra las características como también los componentes de la aplicación.
- java: Está formado por todos los archivos del código fuente.
- res: Está formado por recursos, diseños XML, strings e imágenes.
- Buildgradle módulo: Contiene configuraciones del módulo.
- Buildgradle proyecto: Contiene las configuraciones globales para los módulos [21].

b Interfaz del entorno

La interfaz de Android Studio consta de áreas lógicas como se presenta a continuación.

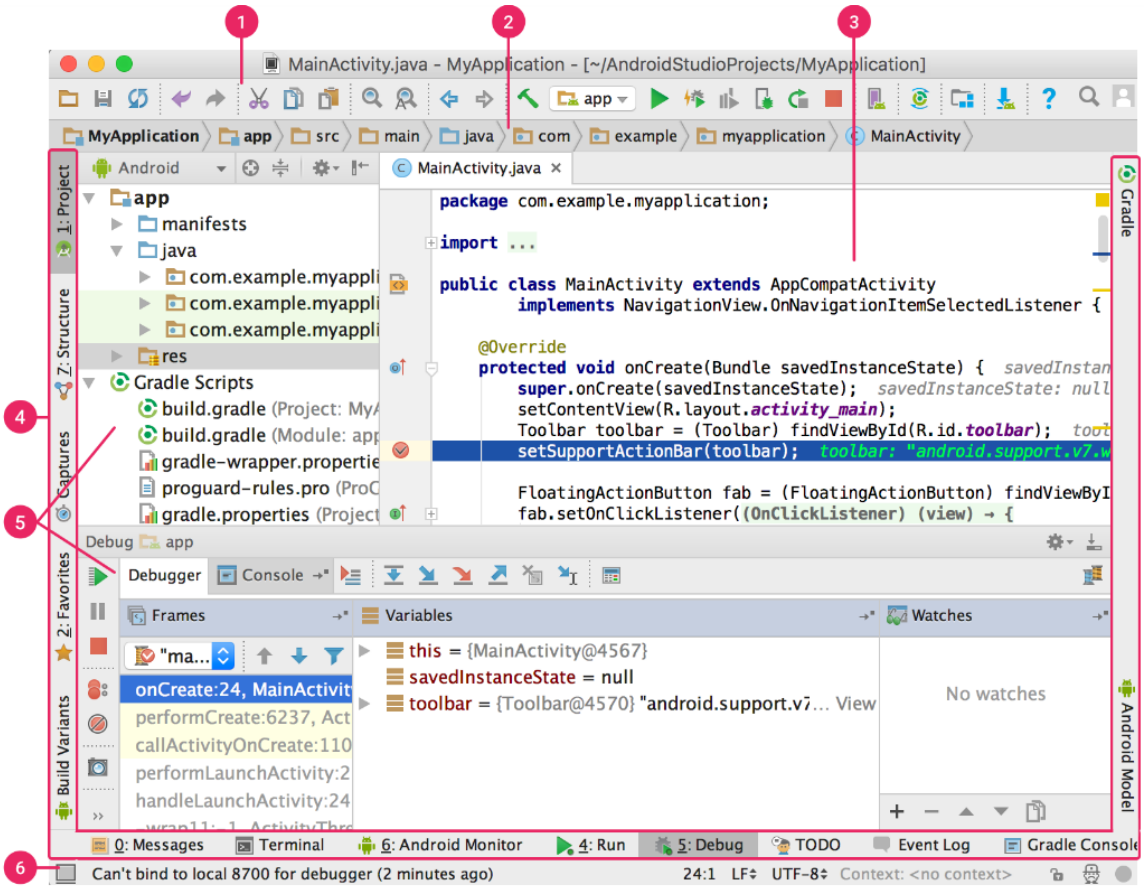


Figura 3.2. Interfaz principal de Android Studio.
Fuente: Obtenido en [20].

1. Tiene varias funciones, como ejecutar la aplicación y también iniciar las herramientas del entorno de desarrollo.
2. Permite navegar y abrir archivos del proyecto. Presenta una estructura compacta**.**
3. Esta zona sirve para editar el código fuente. Dependiendo la actividad actual, el editor puede cambiar.
4. En esta área se encuentran los botones que permiten navegar por las ventanas de las herramientas individuales.
5. Permiten explorar ciertas actividades en específico, la administración, búsqueda, control de versiones, etc. Se puede maximizarlas y minimizarlas.
6. Básicamente esta barra nos indica el estado de nuestro proyecto y algunas notificaciones [20].

3.1.2.2 Creación del proyecto

Una vez visto la introducción al entorno de desarrollo Android Studio como siguiente paso será crear el proyecto, para esto se debe hacer clic en la pestaña File > New > New Project y aparecerá una ventana como se muestra a continuación.

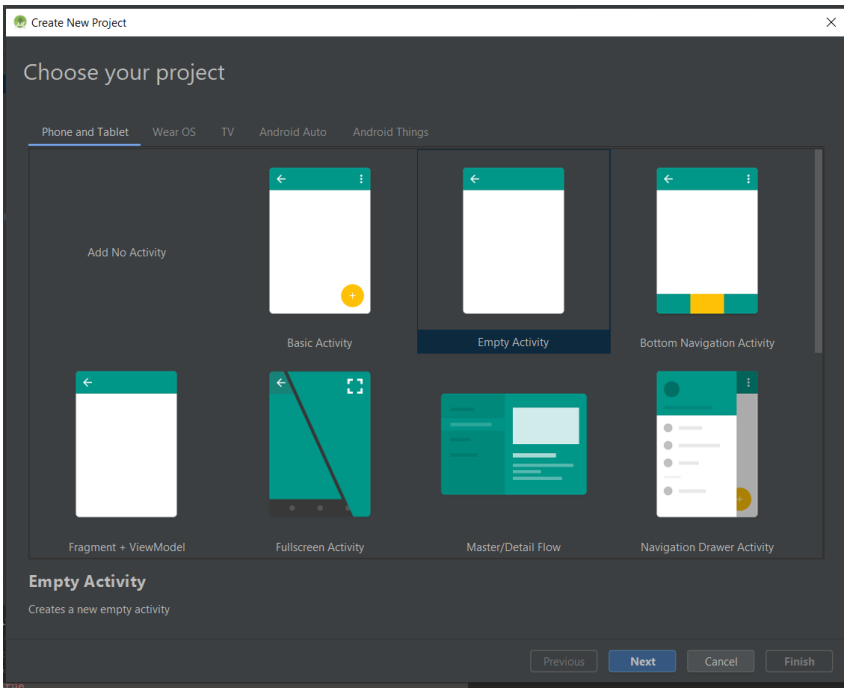


Figura 3.3. Venta para elegir un proyecto.
Fuente: Autores.

En la figura 3.3 se muestra la ventana para escoger el tipo de Actividad que tendrá la aplicación y seleccionaremos Empty Activity, este diseño es para celular y Tablet y finalmente seleccionamos Next.

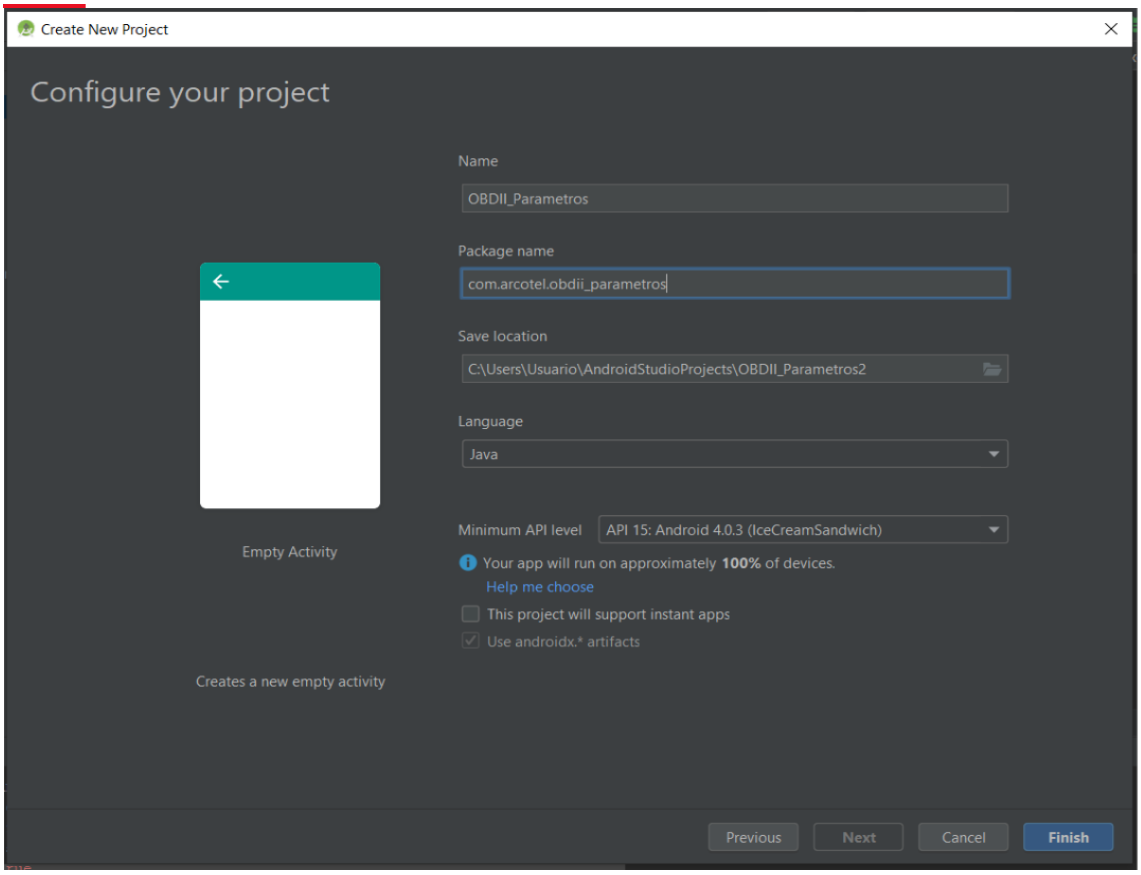


Figura 3.4. Configuración del proyecto.
Fuente: Autores.

La figura 3.4 muestra la configuración del proyecto se muestra el nombre del proyecto como OBII_Parametros y en el nombre del paquete se encuentra como com.arcotel.obdii_parametros y también consta de la ubicación del proyecto.

Android Studio crea por defecto una Clase principal con su respectivo archivo de diseño xml.

Finalmente se muestra como está estructurado el proyecto.

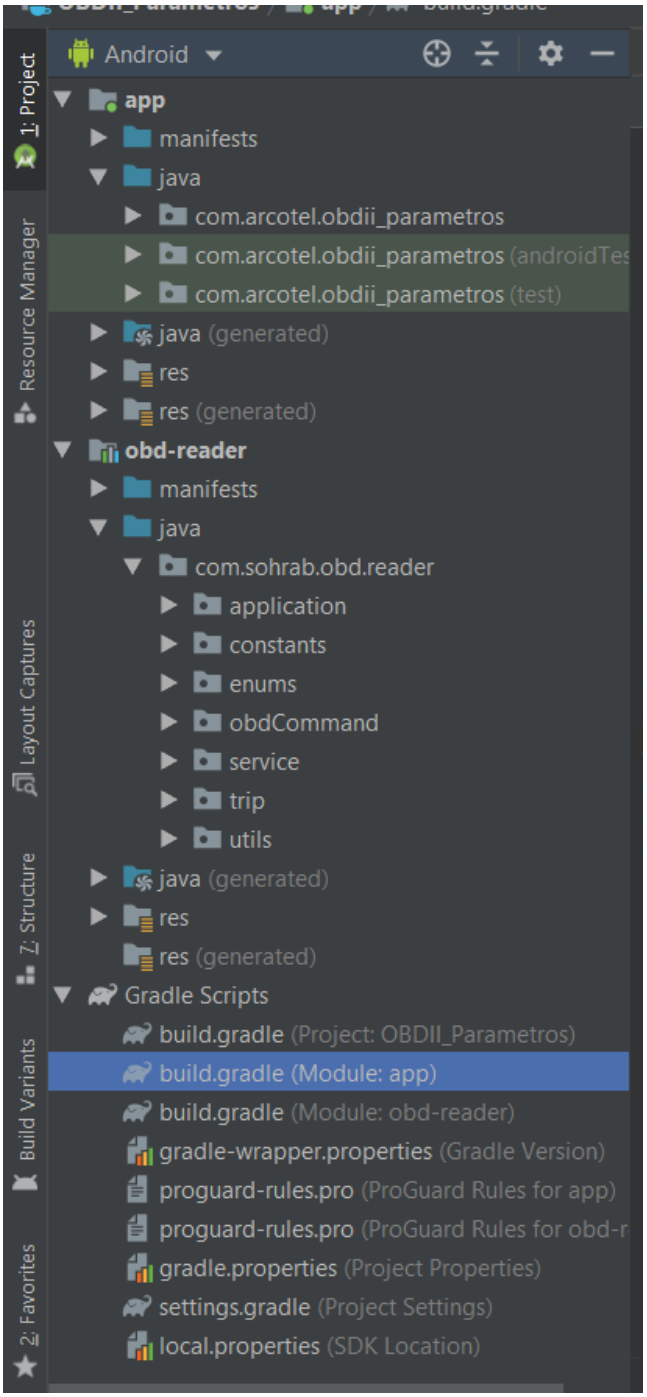


Figura 3.5. Estructura del proyecto.
Fuente: Autores.

En el módulo app se muestra el paquete com.arcotel.obdii_parametros se encuentran todos los archivos condigo fuente Java utilizados en este proyecto.

En el módulo de biblioteca llamado obd-reader consta con todos los archivos de código fuente Java para la recolección de la ECU, misma que se detallará a continuación.

a Biblioteca obd-reader.

El entorno programación escogido permite el uso de módulos de biblioteca, las cuales sirven de gran ayuda para la recolección de los datos de la ECU, la biblioteca a usar es obd-reader.

La biblioteca obd-reader es de código abierto, creado por Sohrab Alam, que es una mejora y corrección de errores de la biblioteca original obd-java-api, tiene como funciones principales la lectura de los datos en tiempo real por parte de los sensores del motor del automóvil provenientes de la ECU. Esta biblioteca tiene comunicación directa con el escáner de diagnóstico ELM327, por tal motivo es ideal y nos permite el diseño de la aplicación para el dispositivo Android.

Características principales de la biblioteca obd-reader son las siguientes:

- Permite establecer la conexión con el ELM327.
- Permite la lectura de los datos en tiempo real.
- Incluye una clase para configurar los PID del modo 01.
- Control sobre errores de lectura.

b Importación de la biblioteca obd-reader a Android Studio

Mencionadas las características principales que posee la biblioteca obd-reader. Para usarla hay que dirigirse a los repositorios de GitHub accediendo a la siguiente dirección https://github.com/ y haciendo uso del buscador se tiene lo siguiente:

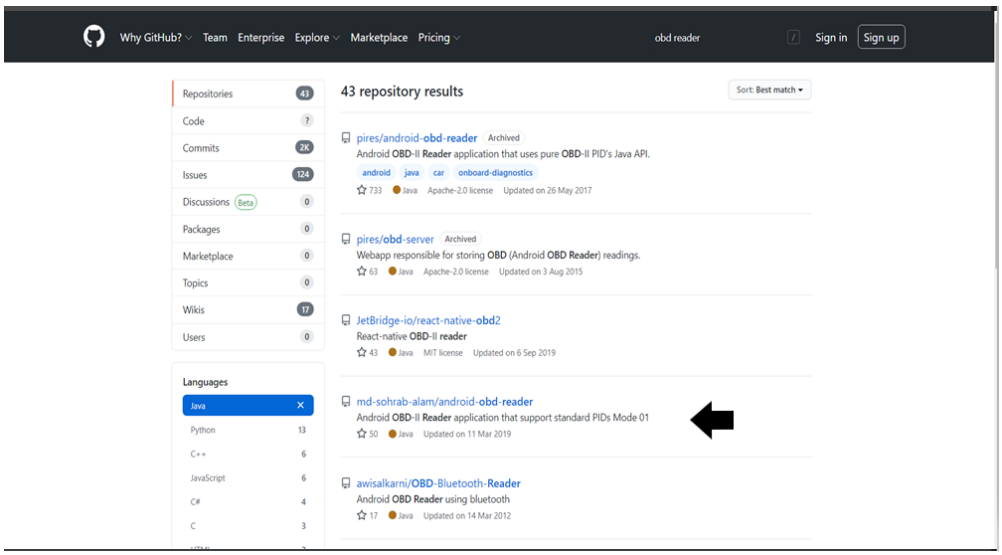


Figura 3.6. Resultados de la búsqueda en los repositorios de GitHub.
Fuente: Obtenido en [22].

Se ha aplicado un filtro para que la búsqueda sea más específica y muestre aquellos repositorios que estén desarrollados en lenguaje Java, ya que la aplicación se desarrolló con el mismo lenguaje. Una vez mostrado los resultados se ha seleccionado el repositorio md-sohrab-alam/Android-obd-reader.

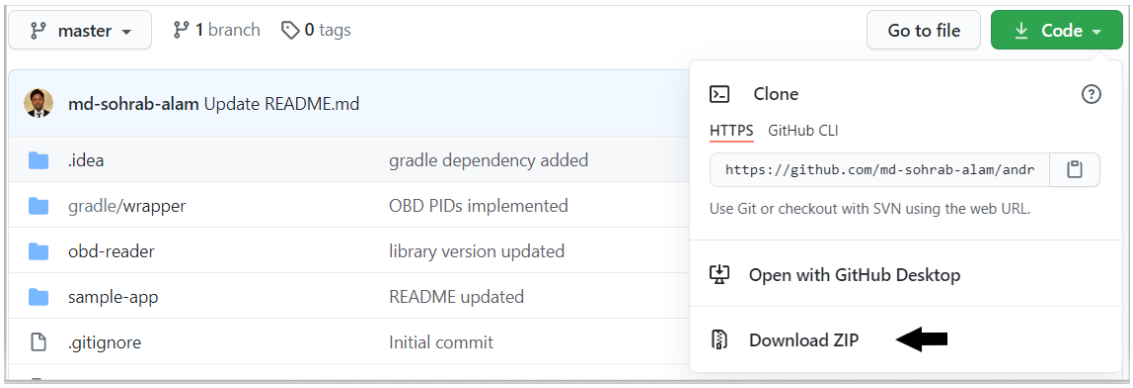


Figura 3.7. Repositorio biblioteca obd-reader.
Fuente: Obtenido en [23].

Como muestra en la figura 3.7, se tiene que descargar la biblioteca como archivo ZIP.

Una vez de descomprimir el archivo ZIP, se debe ingresar a la carpeta android-obd-reader-master y ubicar la carpeta obd-reader en donde se encuentra todos los archivos de la biblioteca, como se muestra en la figura 3.8.

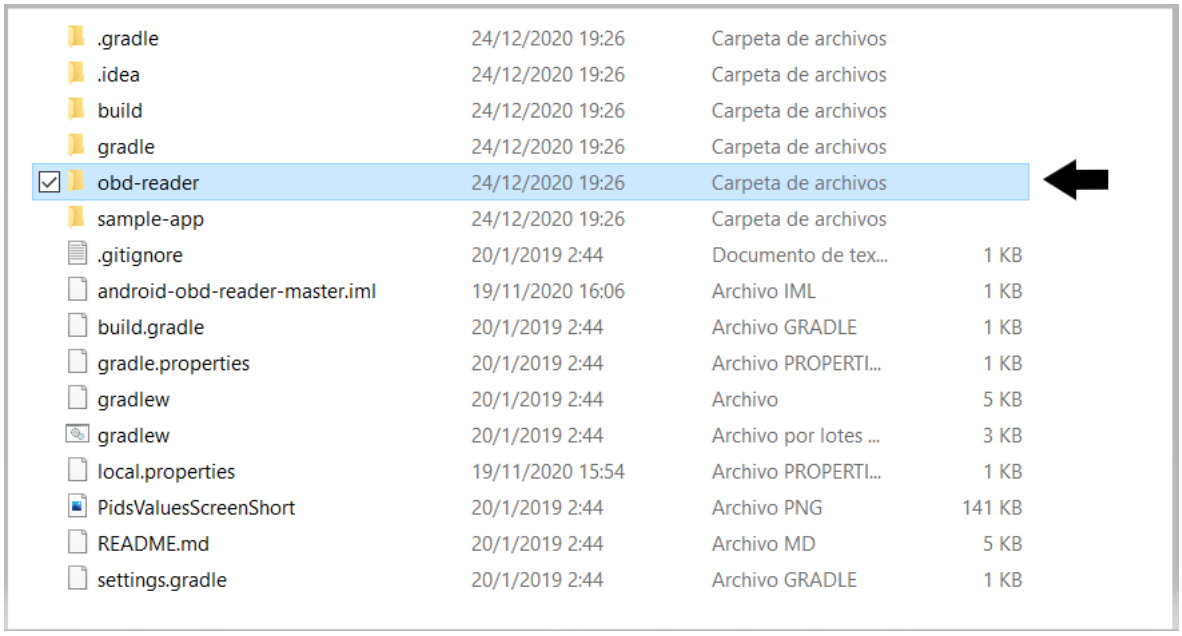


Figura 3.8. Carpeta obd-reader.
Fuente: Autores.

A continuación, se dirige al software Android Studio para importar la biblioteca como módulo, por lo tanto, se debe hacer clic en la pestaña File > New > Import Module, como lo siguiente:

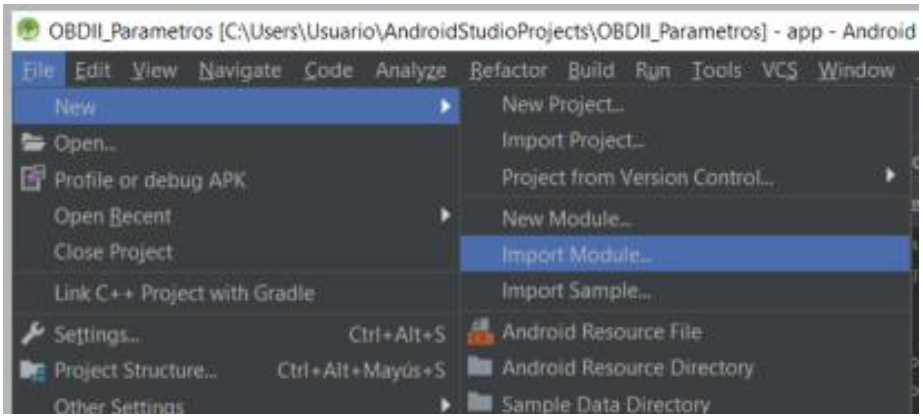


Figura 3.9. Importar biblioteca como modulo.
Fuente: Autores.

Como siguiente paso, se tiene que seleccionar la ubicación de la carpeta de los archivos de la biblioteca, como se había mencionado en la figura 3.8.

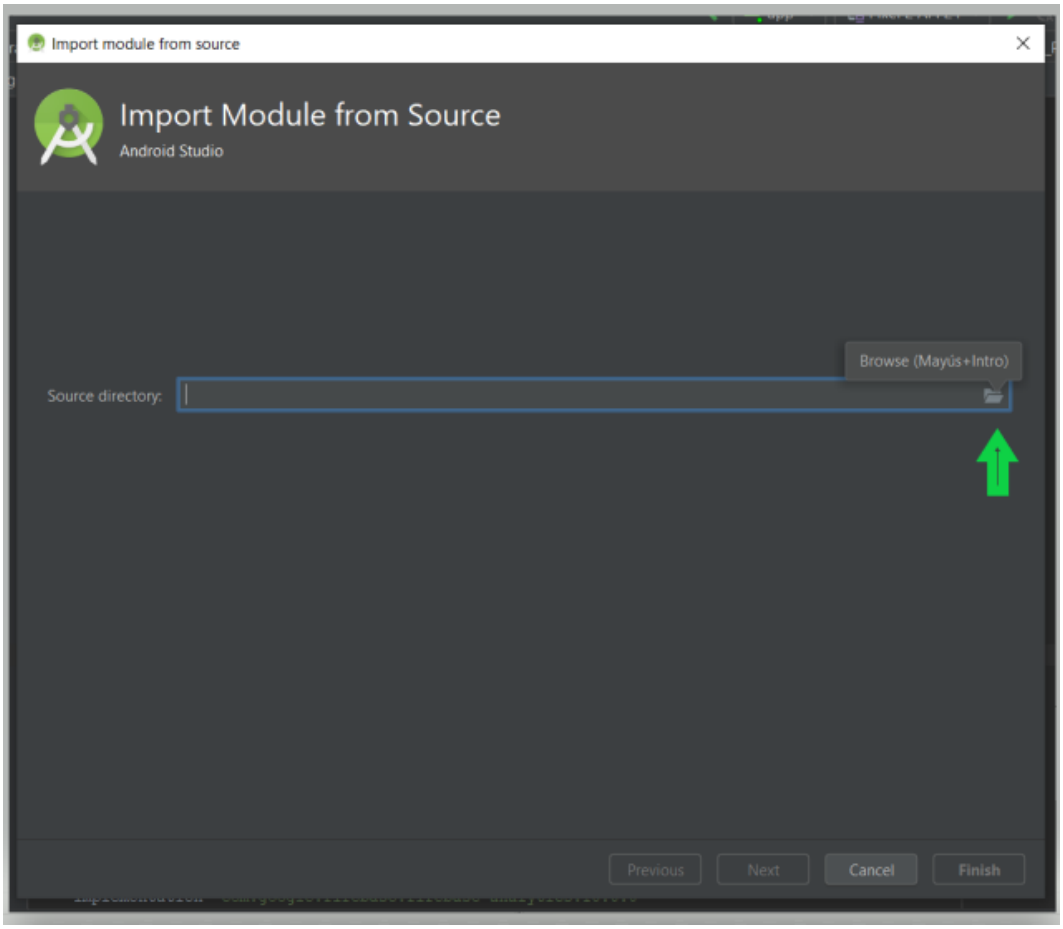


Figura 3.10. Selección de biblioteca obd-reader.
Fuente: Autores.

Una vez de haber importado la biblioteca como módulo, se tiene que dirigirse al módulo de Gradle Scripts y seleccionar el archivo de build.gradle a nivel de proyecto, finalmente se agrega las siguientes dependencias.

```
dependencies {
    classpath 'com.android.tools.build:gradle:3.5.3'
    classpath "com.jfrog.bintray.gradle:gradle-bintray-plugin:1.8.0"
    classpath 'com.github.dcendents:android-maven-gradle-plugin:2.0'
```

Figura 3.11. buil.gradle nivel de proyecto.
Fuente: Autores.

Finalmente, en el mismo módulo de Gradle Scripts se tiene que dirigir al archivo build.gradle a nivel de app y agregar la siguiente dependencia.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation project(':obd-reader')
```

Figura 3.12. build.gradle nivel de app.
Fuente: Autores.

Ya que se agregaron todas las dependencias, la biblioteca esta lista para usarse y se tendría estructurado el proyecto como en la figura 3.5 mostrada.

c Diseño y visualización de los datos

Para empezar a obtener y visualizar los datos de la ECU, se creó una clase llamada “ParametrosOBD” con su respectivo archivo xml para su diseño, como se muestra:

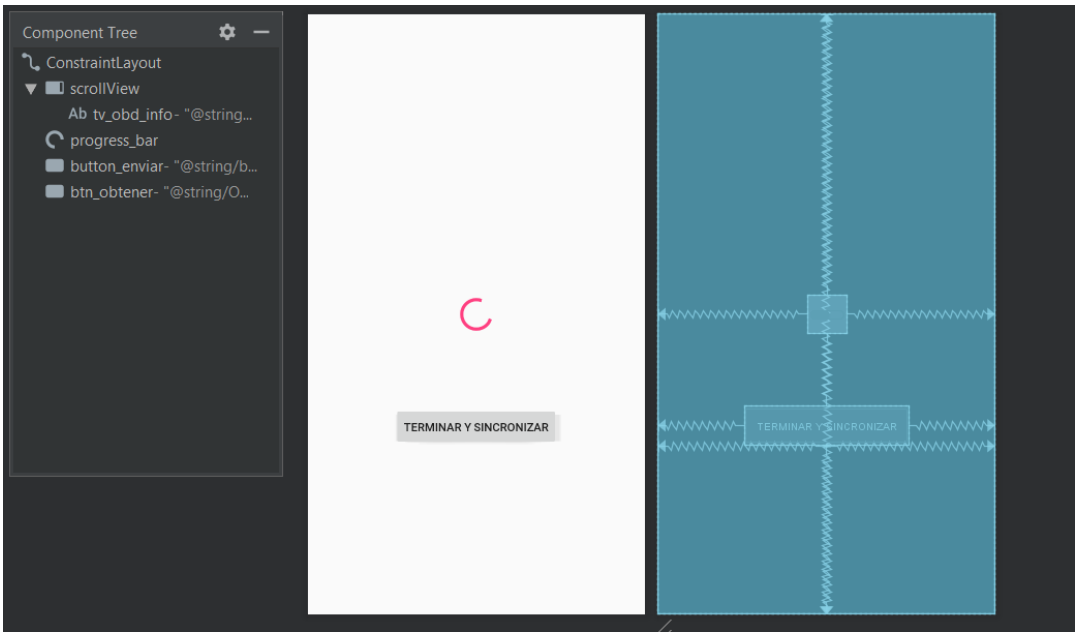


Figura 3.13. Archivo activity_main.xml.
Fuente: Autores.

En la figura 3.13 se muestra el diseño de la actividad donde se visualizarán los datos, cada componente tiene su respectivo id para ser instanciados, se incluyen los siguientes componentes:

- Progress Bar, este componente simula un progreso, el cual pasa a un modo invisible cuando inicie la conectividad con el ELM327.
- Text View, este componente tiene como función mostrar el estado que se generen durante el transcurso de la conectividad con el dispositivo de diagnóstico, además de visualizar los datos del vehículo y finalmente mostrará el estado cuando se termine la obtención de los mismos.
- Button, la aplicación cuenta con dos botones funcionales el botón de "Terminar y sincronizar" enviará y sincronizará, si es necesario, los datos del vehículo. Finalmente, también se añadió un botón de "Obtener parámetros", que permanecerá oculto al usuario y solamente se mostrará en caso de que no se logre una conectividad con el ELM327 o también si el vehículo se apaga perdiendo la conectividad con el dispositivo de diagnóstico, así evitando el reinicio de la aplicación móvil.

```
public class ParametrosOBD extends AppCompatActivity {

    private TextView mObdInfoTextView;
    Button btnenviar,btnobtener;
```

Figura 3.14. Clase ParametrosOBD.
Fuente: Autores.

La figura 3.14 muestra parte de la estructura principal de la clase "ParametrosOBD", donde se declaran las variables de los componentes antes mencionados.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mObdInfoTextView = findViewById(R.id.tv_obd_info);
        btnenviar = findViewById(R.id.button_enviar);
        btnobtener = findViewById(R.id.btn_obtener);
        btnenviar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finializar();
            }
        });
```

Figura 3.15. Método onCreate.
Fuente: Autores.

El método onCreate es el primero en ejecutarse cuando se llama a esta actividad, por lo tanto, aquí se instancian las variables creadas con su respectivo id de los componentes que se mostraron en el diseño para la visualización de los datos.

La biblioteca obd-reader consta de un servicio que se ejecuta en segundo plano, el cual tiene como función principal de establecer la conectividad con el ELM327 y la obtención en tiempo real de los datos de la ECU.

Para recibir los datos se tiene que instanciar un objeto de tipo IntentFilter, para receptar los intents en un BroadcastReceiver que son emitidos por dicho servicio.

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(ACTION_READ_OBD_REAL_TIME_DATA);
intentFilter.addAction(ACTION_OBD_CONNECTION_STATUS);
intentFilter.addAction(ACTION_UBICACION_TIEMPO_REAL);
registerReceiver(LectorOBDReceiver, intentFilter);
String id_vehicle = getIntent().getStringExtra( name: "vehicle_id");
startService(new Intent( packageContext: this, ObdReaderService.class));
startService(new Intent( packageContext: this, UbicacionService.class).putExtra( name: "id_vehicle",id_vehicle));
```

Figura 3.16. IntentFilter
Fuente: Autores.

Dentro del método onCreate, como se muestra en la figura 3.16, se instancia el objeto de tipo IntentFilter agregando los intents o actions que son emitidos por el servicio, consta dos intents que son:

- ACTION_OBD_CONNECTION_STATUS, este intent nos indica receptar el estado de conectividad con el ELM327.
- ACTION_READ_OBD_REAL_TIME_DATA, este intent permite la recepción de los datos de la ECU en tiempo real.

ACTION_UBICACION_TIEMPO_REAL, este intent permite obtener la ubicación a través del GPS del dispositivo móvil en tiempo real, este intent es emitido por otro servicio llamado “UbicacionService”.

Finalmente se llama al receptor “LectorOBDReceiver” y se ejecuta el servicio “ObdReaderService”.

```
private final BroadcastReceiver LectorOBDReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {

        findViewById(R.id.progress_bar).setVisibility(View.GONE);
        mObdInfoTextView.setVisibility(View.VISIBLE);
        String action = intent.getAction();
```

Figura 3.17. BroadcastReceiver LectorOBDReceiver.
Fuente: Autores.

En la figura 3.17 se muestra el BroadcastReceiver, el cual permitirá recibir mediante intents el estado de conexión con el ELM327 y la obtención de los datos de la ECU en tiempo real.

```
if (action.equals(ACTION_OBD_CONNECTION_STATUS)) {

    String connectionStatusMsg = intent.getStringExtra(ObdReaderService.INTENT_OBD_EXTRA_DATA);
    mObdInfoTextView.setText(connectionStatusMsg);
    Toast.makeText( context: ParametrosOBD.this, connectionStatusMsg, Toast.LENGTH_SHORT).show();

    if (connectionStatusMsg.equals(getString(R.string.obd_connected))) {

    } else if (connectionStatusMsg.equals(getString(R.string.connect_lost)) || connectionStatusMsg.equals(getString(R.string.obd2_adapter_not_responding)) ) {
        desconectar();
    } else {
```

Figura 3.18. ACTION_OBD_CONNECTION_STATUS.
Fuente: Autores.

El intent ACTION_OBD_CONNECTION_STATUS permite controlar la conectividad con el ELM327 que va desde la búsqueda de dispositivos bluetooth, emparejamiento y establecer la conexión con el dispositivo de diagnóstico, como se muestra en la figura 3.18. Además, se implementó un método llamado desconectar que es llamado cuando se pierda la conexión con el ELM327 ya sea por qué el dispositivo se des energizó debido a que apagaron el vehículo o en caso de que no se pudo establecer una conexión con el mismo.

Al diseño de la aplicación se agregó un servicio para obtener la ubicación en tiempo real, como se muestra en la siguiente figura:

Para que el servicio pueda obtener las coordenadas del dispositivo móvil en el archivo AndroidManifest.xml se tiene que agregar los siguientes permisos:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.arcotel.obdii_parametros">

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
```

Figura 3.19. AndroidManifest.xml permisos.
Fuente: Autores.

Adicionalmente, se debe agregar las siguientes dependencias en el archivo app/build.gradle a nivel de app.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation project(':obd-reader')
    implementation 'com.android.support.constraint:constraint-layout:2.0.1'
    implementation 'com.google.android.gms:play-services-location:17.1.0'
```

Figura 3.20. Dependencias para obtener la ubicación.
Fuente: Autores.

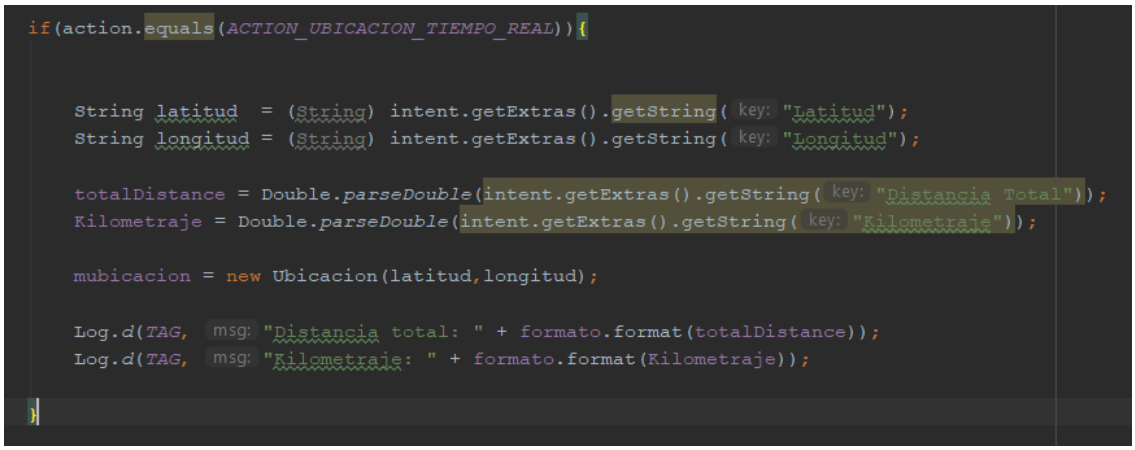


Figura 3.21. ACTION_UBICACION_TIEMPO_REAL.
Fuente: Autores.

El servicio denominado “UbicacionServicie”, que se ejecuta en segundo plano, obtiene actualizaciones de coordenadas en intervalos de 1 segundo y dependiendo del estado del vehículo, es decir, si se encuentra en movimiento se encuentra la distancia entre dos coordenadas. De tal manera que se pueda obtener la distancia recorrida y a su vez poder llevar un control del kilometraje del vehículo ya que a través del dispositivo de diagnóstico no se puede obtener este dato.

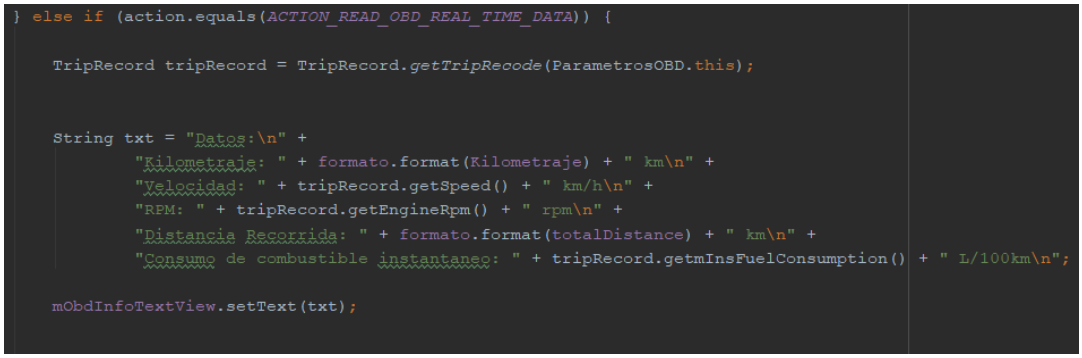


Figura 3.22. ACTION_OBD_REAL_TIME_DATA.
Fuente: Autores.

Luego se haberse establecido la conexión con el ELM327, el intent ACTION_OBD_REAL_TIME_DATA permite recibir los datos en tiempo real y ser visualizados en la aplicación, los datos a mostrar:

- Kilometraje.
- Velocidad
- RPM
- Distancia recorrida
- Consumo de combustible instantáneo

Para obtener los parámetros del vehículo como se mencionó en el capítulo 2 se debe enviar los comandos OBD también conocidos como PID (Parameter ID) para luego ser procesados, cada PID usado tiene su fórmula para su interpretación y se presenta a continuación:

PID (hex)	Descripción	Bytes de Respuesta	Fórmula
01 0D	Velocidad del vehículo	1	A
01 0C	Revoluciones del motor	2	(256A+B)/4
01 10	Velocidad del flujo del aire MAF	2	(256A+B)/100

Tabla 3.1. Fórmulas de los PID.
Fuente: Autores.

El PID 01 10 que corresponde a la velocidad del flujo del aire MAF ayudará a calcular o tener un aproximado consumo de combustible instantáneo del vehículo, aunque en la norma OBD II existe el PID 01 5E para la velocidad del combustible del motor medida en L/h, sin embargo, no se dispone de este en la mayoría de los vehículos. Debido a este problema se ha optado por usar el MAF, para solucionar este problema se usa la siguiente ecuación [24]:

$$Fuel_{Consumption} = \left(\frac{MAF}{AFR * D * V}\right) * 3600 \qquad (3.1)$$

Donde el MAF es la velocidad del flujo del aire, la relación de aire-combustible AFR, la densidad del combustible en gramos por decímetro cubico g/dm^3 (D), la velocidad del vehículo en Km/ h (V) y 3600 es un factor de conversión de segundos a horas. La relación aire/combustible AFR es igual a 14.7:1 y la densidad 680 kg/m^3 [24].

Si el tipo de combustible usado es diésel el AFR equivale a 14.5:1 y la densidad es de 850 kg/m^3 [24].

3.1.2.3 Envió de los datos

Una vez de haber obtenido los datos de la ECU como siguiente objetivo es el envío de los mismos, para esto se hizo uso de los servicios de Google en este caso de Firebase para el almacenamiento en la nube teniendo acceso remoto y monitorear los datos del vehículo.

a Firebase.

Es una de las funciones integradas a Android Studio para el diseño de aplicaciones móviles y que también sirve para aplicaciones web. Una plataforma creada en 2012 por James T. y Andrew L. y en 2014 pasó a ser dirigida por Google [25].

Su función principal era una plataforma de chat, allí James T. y Andrew L. detectaron que usaban ampliamente este chat para transmitir información de las aplicaciones, como guardar información de sus juegos. En ese momento, deliberaron dividir estas funciones, la plataforma de chat de la arquitectura en tiempo real, lo que llevó a la creación de Firebase en abril de 2012 [25].

Figura 3.23. Logo de Firebase.
Fuente: Obtenido en [26].

Firebase tiene herramientas que permiten intercambiar datos entre aplicaciones móviles o web, y también se puede utilizar para aplicaciones como IOS y Android. Una de sus funcionalidades es la implementación de una base de datos, que opera en tiempo real, teniendo como ventaja la sincronización instantánea de la información de las aplicaciones e inclusive permite usar métodos para la autenticación de usuarios y brindar una mayor seguridad. También posee servicios como hosting para compartir o publicar las aplicaciones [27].

b Funciones y características de Firebase

Google Analytics: Es una función que permite estimar las estadísticas del uso de las aplicaciones como también la interacción del usuario. Los informes de analytics brindan una comprensión clara del comportamiento de sus usuarios para que se pueda tomar decisiones informadas sobre el marketing de aplicaciones y la optimización del rendimiento [28].

Firebase Authentication: La mayoría de las aplicaciones tienen que identificar a los usuarios. Si se conoce las credenciales del usuario, la aplicación puede almacenar los datos de forma segura en la nube y funciona en todos los dispositivos que tenga el usuario [29].

Realtime Database: Los datos son alojados en la nube, se guardan en un formato json y se sincronizan con cada usuario conectado en tiempo real, y permanecen disponibles incluso cuando la aplicación no tiene acceso a la red [30].

Cloud Firestore: Esta base es flexible y escalable, puede ser programada tanto en servidores como en aplicaciones móviles. Al igual que la anterior función, mantiene sincronizados los datos de los usuarios mediante métodos de escucha en tiempo real. Además, proporciona el uso sin conexión para dispositivos móviles y la web para que se pueda crear aplicaciones receptivas que funcionen a pesar de la inexistencia de la red o esta sea lenta [31].

Almacenamiento (cloud storage): Este es una de las funcionalidades para el almacenamiento de objetos poderoso, simple y con un precio razonable. Sin importar del estado de la red, puede proporcionar carga y descarga de archivos seguras para las aplicaciones. Los desarrolladores pueden usarlo para guardar archivos multimedia u algún otro tipo de contenido [32].

Hosting: Está diseñado para desarrolladores web modernos. Gracias a la aparición de frameworks front-end de JavaScript, los sitios web y las aplicaciones son más poderosos que nunca [33].

Cloud Messaging (FCM): Es una opción para la mensajería de manera gratuita y muy segura. Características como, alertar al usuario mediante notificaciones que le ha llegado un correo nuevo o para la sincronización de los datos. [34].

c Integrar Firebase al proyecto de Android

Para integrar Firebase al proyecto de Android existen dos opciones para poder hacerlo:

- 1: A través de la configuración de la consola de Firebase.
- 2: Utilizar el asistente de Android Studio (es posible que requiera configuración adicional).

En este caso se utilizó la opción dos que consiste en usar el Firebase Assistant de Android Studio y agregar los archivos, los complementos y las dependencias que sean necesarias de Firebase en el proyecto de Android.

1. Dentro de del proyecto de Android Studio se tiene que acceder a Firebase Assistant:
   a. Dirigirse a **Tools > Firebase** para acceder al panel Asistente.

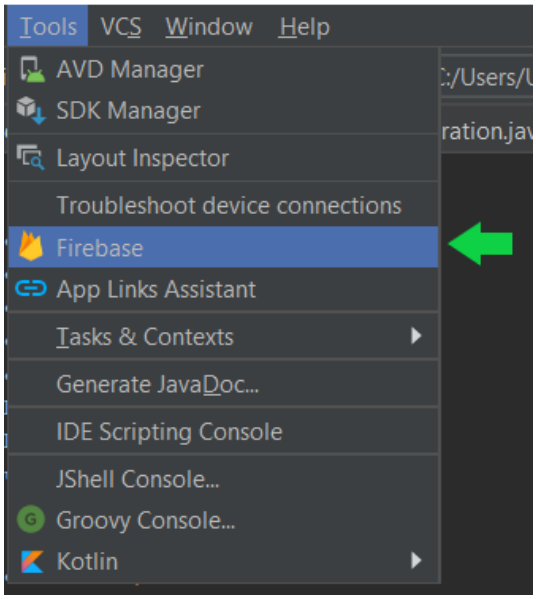


Figura 3.24. Pestaña de herramientas de Android Studio.
Fuente: Autores.

2. Elegir un producto de Firebase para agregar a la aplicación. Al expandir se muestra lo siguiente:

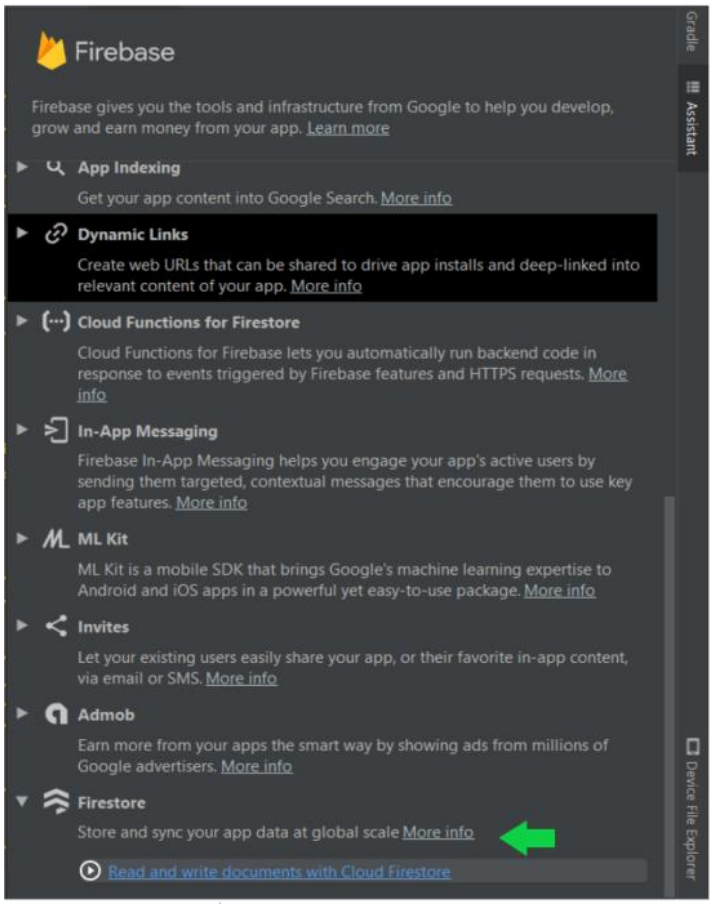


Figura 3.25. Firebase Assistant.
Fuente: Autores.

La figura 3.25 muestra todos los productos que ofrece Firebase para integrar a la aplicación, los cuales ya fueron mencionados anteriormente, para la aplicación se eligió el producto de Firestore el cual consiste el uso de Cloud Firestore.

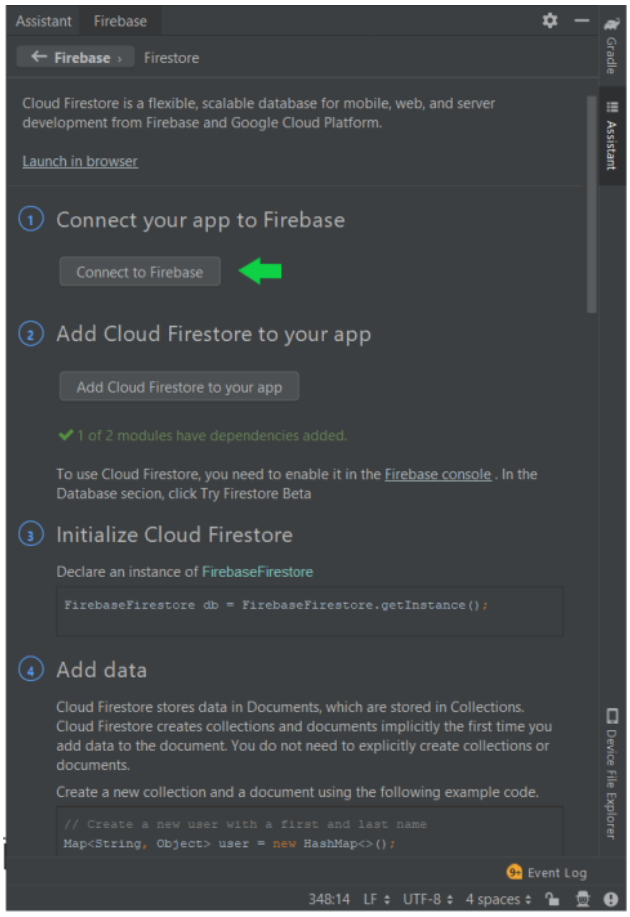


Figura 3.26. Pasos para implementar Firestore.
Fuente: Autores.

3. Ingresar o crear una cuenta Gmail para administrar el proyecto de Firebase.
4. Seleccionar o crear un nuevo proyecto para Firebase.

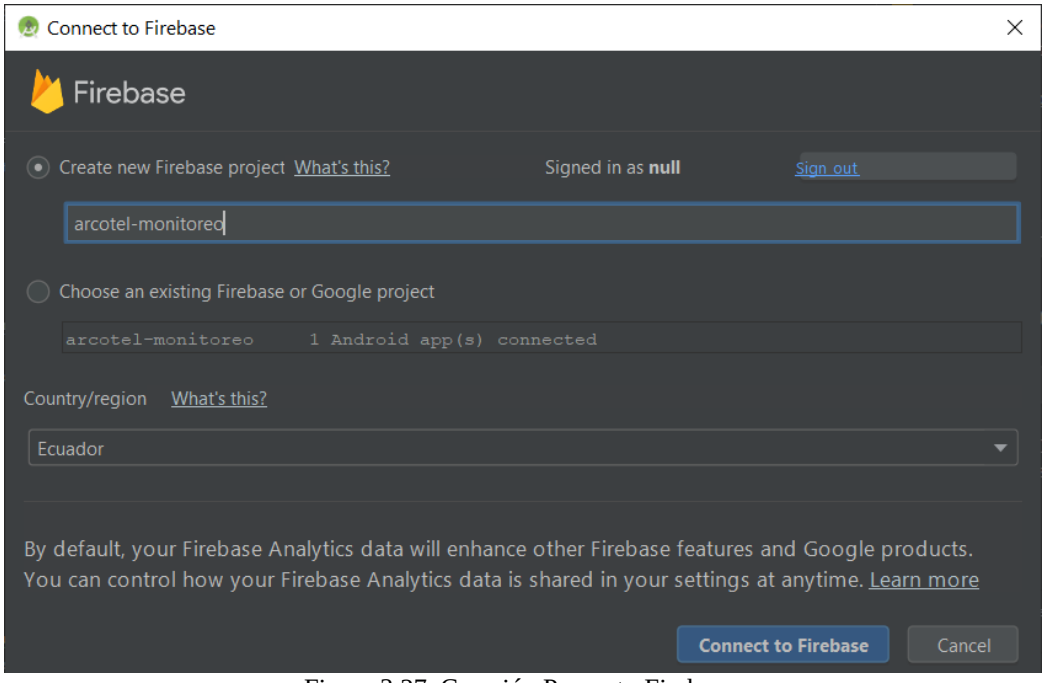


Figura 3.27. Creación Proyecto Firebase.
Fuente: Autores.

En la figura 3.27 se muestra cómo crear o seleccionar un proyecto de Firebase desde Android Studio, luego de haber creado el proyecto se tiene que verificar si hay un archivo llamado google-services.json como se muestra a continuación:

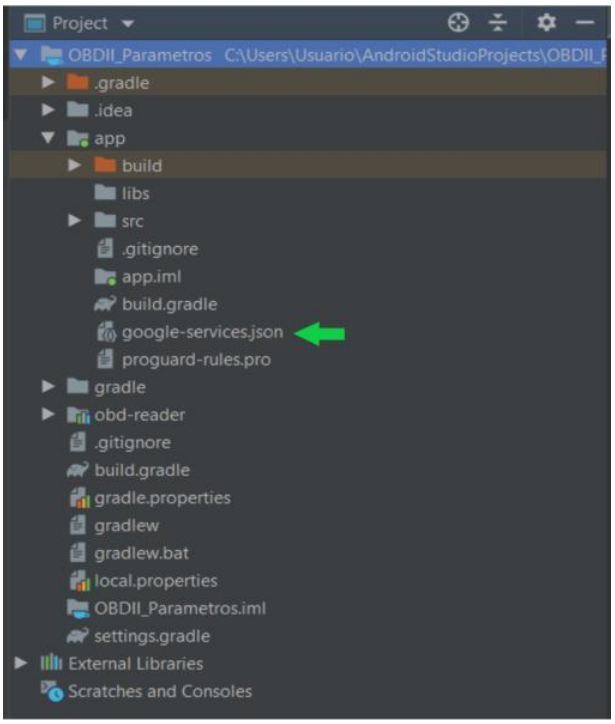


Figura 3.28. Archivo google-services.json.
Fuente: Autores.

En caso que no se haya agregado de forma automática este archivo, se tiene que dirigir al dashboard del proyecto de firebase y entrar a la configuración del mismo.

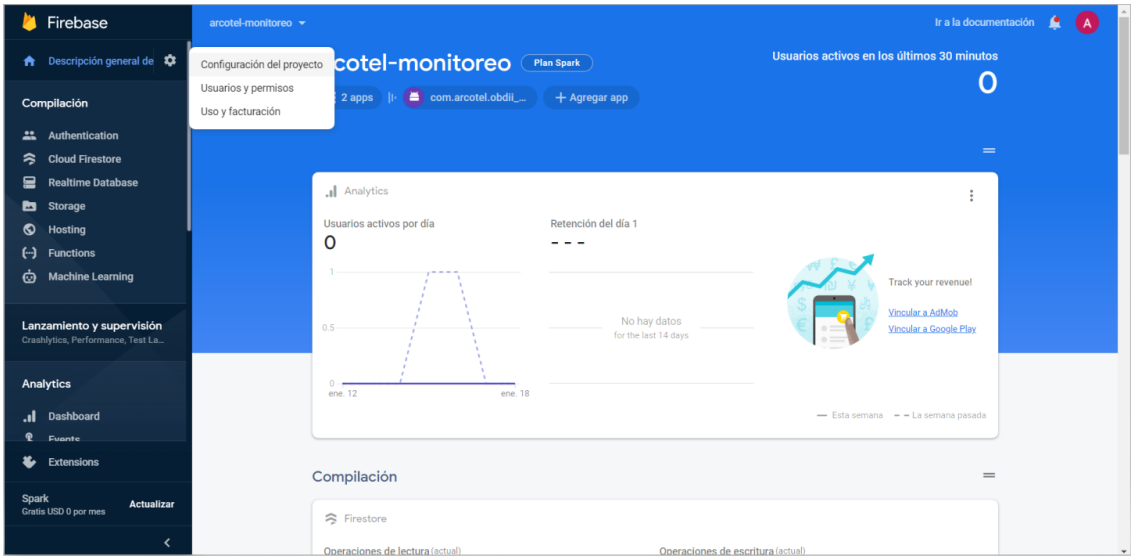


Figura 3.29. Dashboard del proyecto arcotel-monitoreo.
Fuente: Autores.

Luego de ingresar a la configuración del proyecto, en la pestaña general encontraremos el archivo que se tiene que descargar y agregar en los repositorios del proyecto de Android Studio, como se muestra:

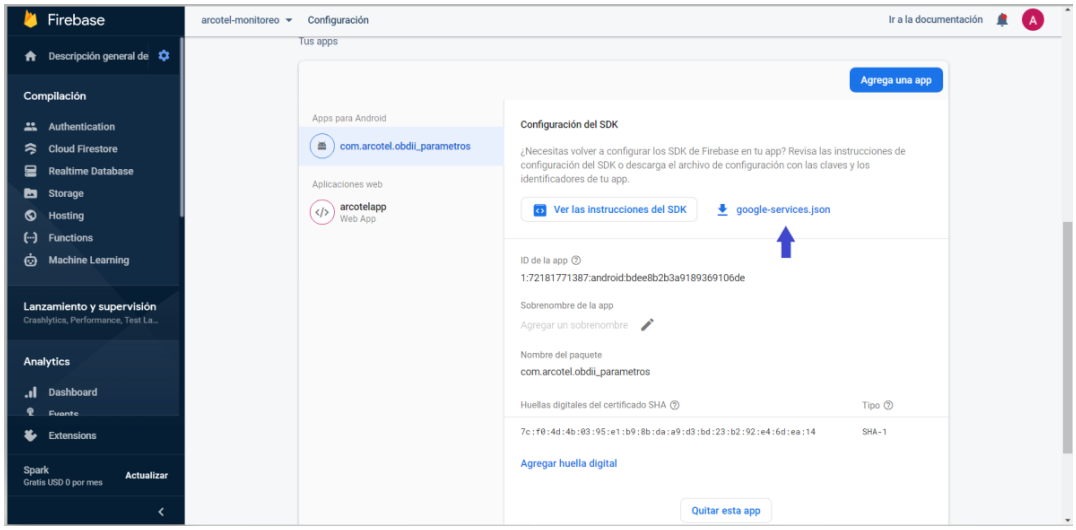


Figura 3.30. Configuraciones generales del proyecto de Firebase.
Fuente: Autores.

La figura 3.30 muestra las configuraciones generales, se puede observar que la aplicación de Android Studio está integrada con el proyecto de Firebase, donde encontraremos el nombre del paquete de la aplicación, el certificado SHA-1 e inclusive se encuentra el archivo google-services.json que permite establecer la conexión de la aplicación con el proyecto de Firebase llamado arcotel-monitoreo.

5. Agregar las dependencias necesarias

   a. Agregar las dependencias al archivo Gradle (build.gradle) a nivel de proyecto, como lo siguiente:

```
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.3'
        classpath "com.jfrog.bintray.gradle:gradle-bintray-plugin:1.8.0"
        classpath 'com.github.dcendents:android-maven-gradle-plugin:2.0'
        classpath 'com.google.gms:google-services:4.3.3'
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

Figura 3.31. Gradle a nivel de proyecto.
Fuente: Autores.

   b. Al archivo Gradle (app/build.gradle) a nivel de app, agregar lo siguiente:

```
1  apply plugin: 'com.android.application'
2  apply plugin: 'com.google.gms.google-services'
3
```

Figura 3.32. Gradle a nivel de app.
Fuente: Autores.

d Autenticación de usuarios

La aplicación como parte del diseño tendrá la autenticación de los usuarios y para que estos puedan acceder. La autenticación del usuario puede ser mediante correo electrónico y contraseña u optar por una llave de un proveedor de identidad federada (Google, Facebook, Twitter, GitHub).

Antes de utilizar el producto de autenticación de Firebase se tiene que implementar las siguientes dependencias dentro del app/build.gradle a nivel de app.

```
implementation platform('com.google.firebase:firebase-bom:25.12.0')
implementation 'com.google.firebase:firebase-auth:20.0.0'
implementation 'com.google.android.gms:play-services-auth:18.1.0'
```

Figura 3.33. Dependencias para la autenticación.
Fuente: Autores.

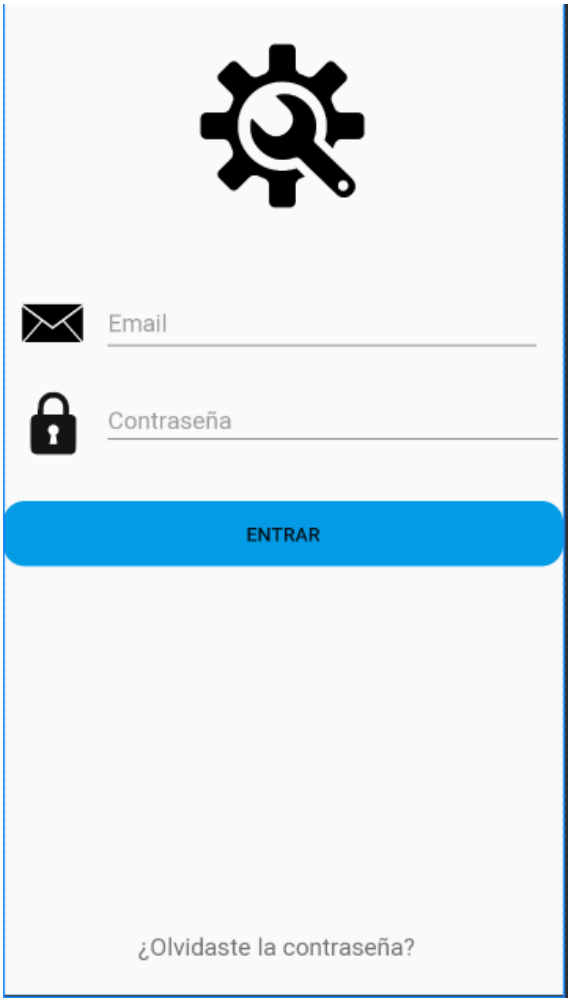


Figura 3.34.Diseño login de la aplicación.
Fuente: Autores.

La figura 3.34 muestra el diseño de inicio de sesión para el usuario. La aplicación tiene como grupo beneficiario a la empresa de ARCOTEL, entonces desde el dashboard del proyecto de Firebase se tienen algunos correos de los trabajadores de la empresa con una contraseña establecida por defecto, que en cualquier momento pueden reestablecerla desde la aplicación.

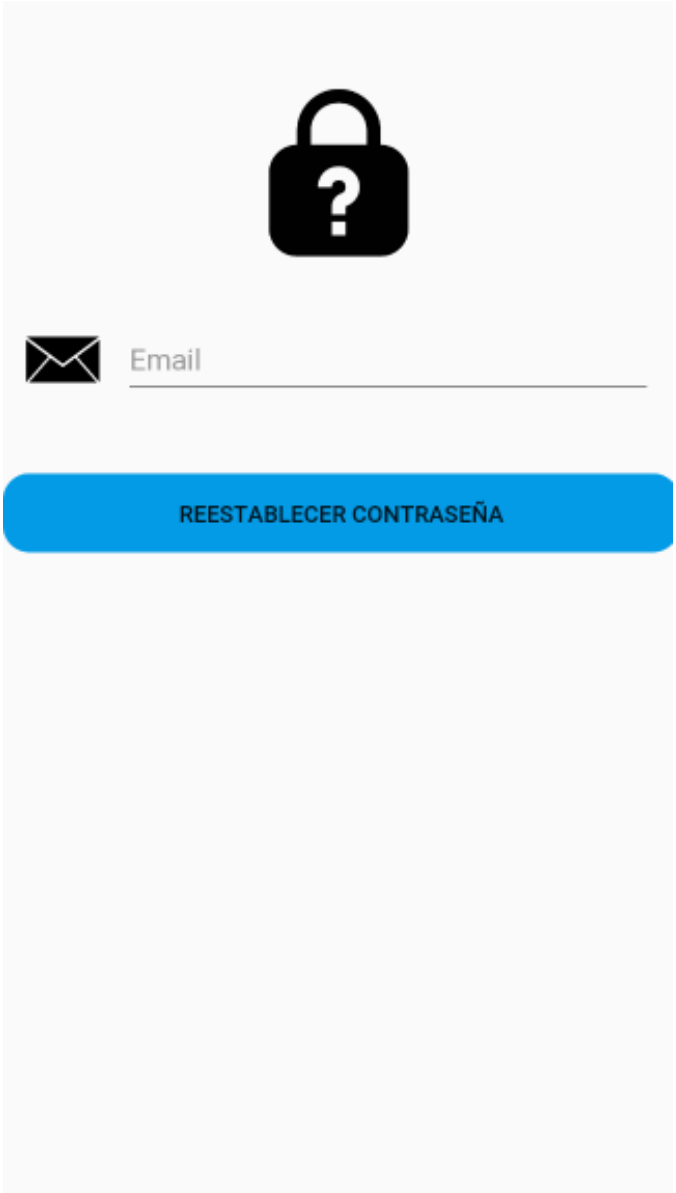


Figura 3.35. Diseño reestablecer contraseña.
Fuente: Autores.

La interfaz para reestablecer la contraseña se observa en la figura 3.35 y consiste en ingresar el correo del usuario, el cual debe estar ya registrado dentro del proyecto de Firebase e incluso debe ser un correo válido, para que se pueda enviar un mensaje de confirmación al correo del usuario y que contiene una dirección donde será redirigido a un sitio donde podrá ingresar la contraseña nueva.

3.1.2.4 Almacenamiento y estructura de los datos

Una vez visto una introducción a Firebase donde se enunciaban todos los productos que ofrece este servicio, y uno de estos productos es Cloud Firestore.

a Cloud Firestore

Se ha escogido Cloud Firestore ya que es una base de datos fácil de utilizar para los desarrolladores de aplicaciones móviles, aplicativos web y Google Cloud Platform; además permite una mejor estructuración de los datos. Sus datos se alojan en la nube. Se puede acceder a sus aplicaciones web, Android y iOS de manera directa desde el SDK local. Además de REST y API de RPC. [31].

Los datos se pueden almacenar en documentos que contienen información mediante campos asignados con tipos valores. Los documentos se organizan y guardan a través de colecciones, facilitando la complicación de consultas. Estos documentos permiten almacenar varios tipos de datos, como simples cadenas, números y hasta estructuras con anidaciones de objetos [35].

Para empezar a usar Cloud Firestore se deben agregar las siguientes dependencias (app/build.gradle) a nivel de app, como se muestra en la figura 3.36:

```
implementation 'com.google.firebase:firebase-firestore:22.0.0'
implementation platform('com.google.firebase:firebase-bom:25.12.0')
```

Figura 3.36. Dependencias para Cloud Firestore.
Fuente: Autores.

b Estructura de los datos

Cloud Firestore permite tener una buena estructuración de los datos a través de documentos, los cuales se almacenan en colecciones y realizar consultas.

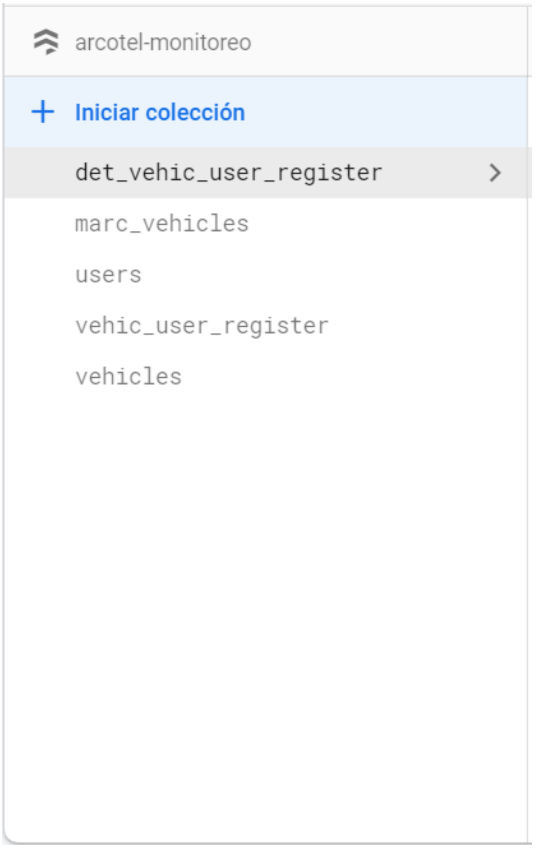


Figura 3.37. Colecciones.
Fuente: Autores.

La figura 3.37 muestra las colecciones que se han creado en el proyecto de Firebase para organizar los datos, cada una de estas colecciones cuenta con documentos, en los cuales se encuentra diferentes campos que corresponden a los diferentes datos del vehículo como también la información del usuario. A continuación, se describirán que contienen los documentos de esas colecciones.

b.1 Colección vehicles

Está colección contiene una lista de documentos donde cada uno de estos posee información de los diferentes vehículos de la empresa ARCOTEL, como se muestra:

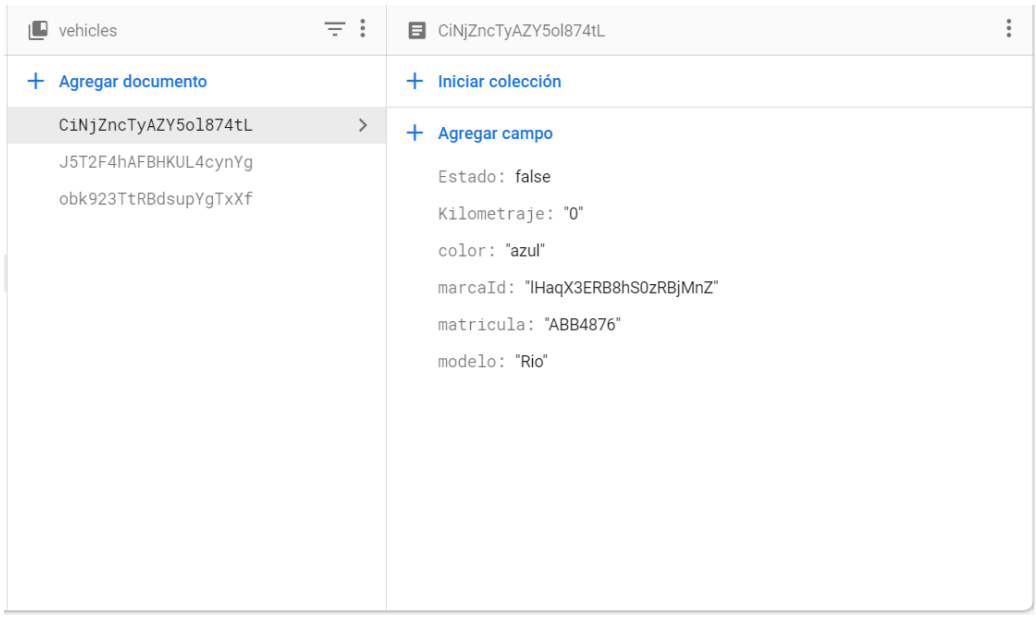


Figura 3.38. Colección vehicles.
Fuente: Autores.

Como se observa la figura 3.38 la colección “vehicles” consta de varios documentos que corresponden a información de los vehículos. Esta colección con sus documentos fue definida previamente, es decir, el usuario tendrá a disposición de escoger el vehículo con el cual va a realizar el recorrido.

Figura 3.39. Lista de vehículos.
Fuente: Autores.

Al usuario dentro de la aplicación se le mostrará la lista de los vehículos de la empresa con su respectiva información como se puede observar en la figura 3.39. Aquí aprovecharemos lo que nos ofrece Cloud Firestore, cualquier cambio que se haga en la base de datos se sincronizará inmediatamente con el usuario a través de la lista de los vehículos, esto incluye editar, eliminar o agregar.

En la figura 3.38 se tiene dos campos que no son mostrados al usuario, estos campos tienen otra función. Empezando con el campo “Estado” este es un dato de tipo booleano que en primera instancia se encuentra en un estado falso, cuando el usuario seleccione un vehículo este estado pasará de falso a un estado verdadero, esto ayuda a tener un control sobre que vehículos y cuales no están en un recorrido.

Por otro lado, se tiene el campo en la figura 3.38 el Kilometraje, este campo permitirá al usuario ingresar de forma manual y única vez el kilometraje del vehículo, como ilustración se tiene:

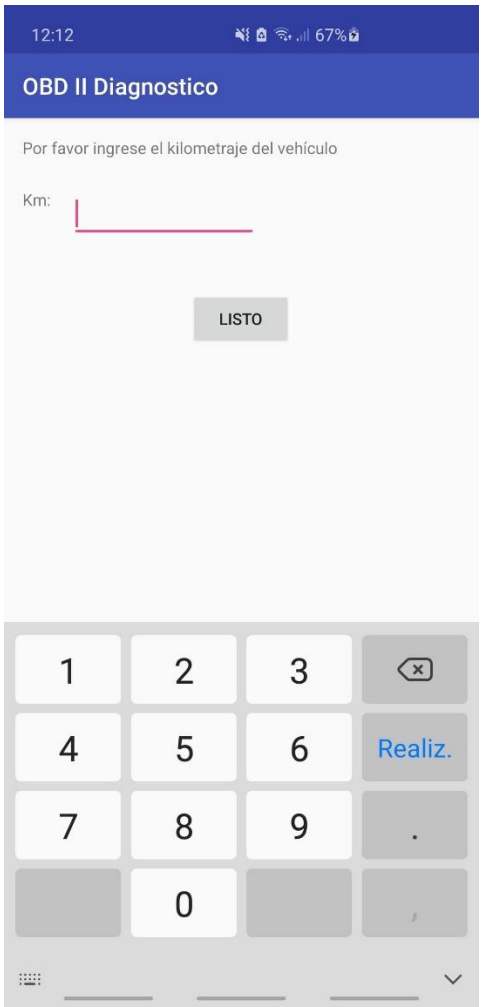


Figura 3.40. Ingreso del kilometraje.
Fuente: Autores.

El usuario deberá ingresar el kilometraje del vehículo siempre y cuando en la base de datos de la colección vehicles en el campo de “Kilometraje” esté un valor de 0, una vez que ingrese el valor del kilometraje este valor será actualizado instantáneamente en la base de datos para posteriormente tener un control del kilometraje del vehículo en los diferentes recorridos que se realicen.

b.2 Colección users

Está colección contiene documentos con información del usuario como es el correo y un id que nos servirá como un identificador, los se generarán cuando el usuario haya iniciado sesión en la aplicación.

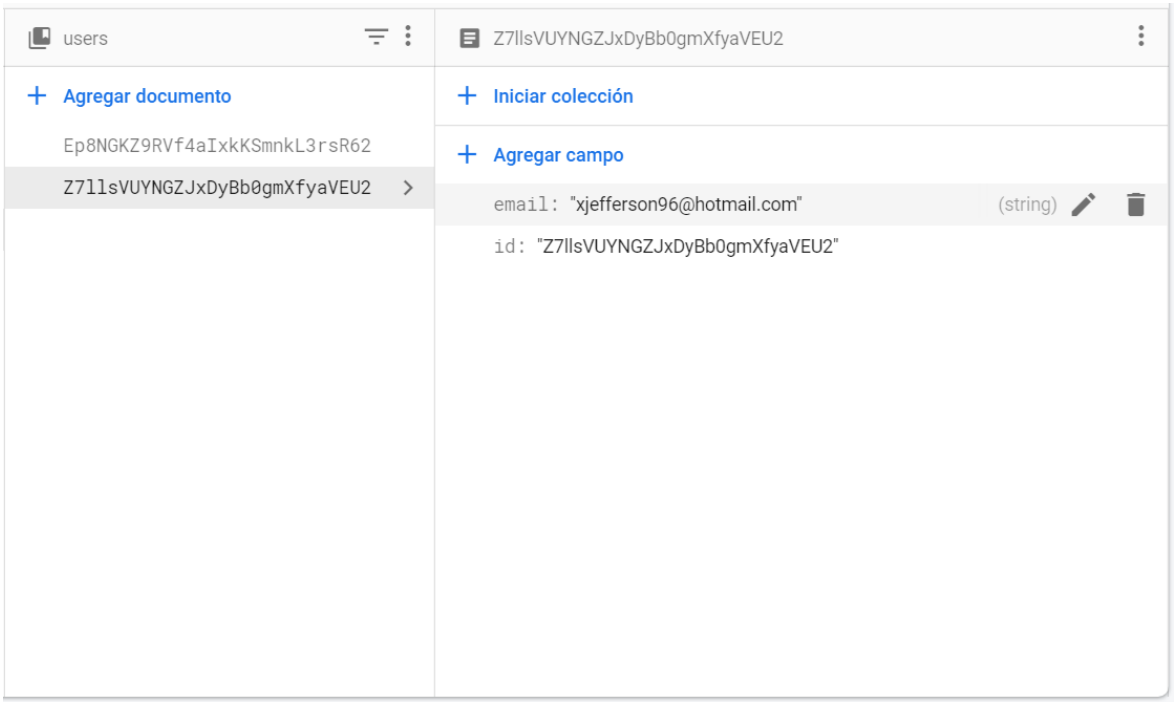


Figura 3.41. Colección users.
Fuente: Autores.

b.3 Colección marc_vehicles

En esta colección están alojados los documentos con las marcas de los vehículos, estos documentos tienen un id que está en el campo llamado como “marcaId” en los documentos de la colección vehicles. A continuación, se muestra la colección marc_vehicles:

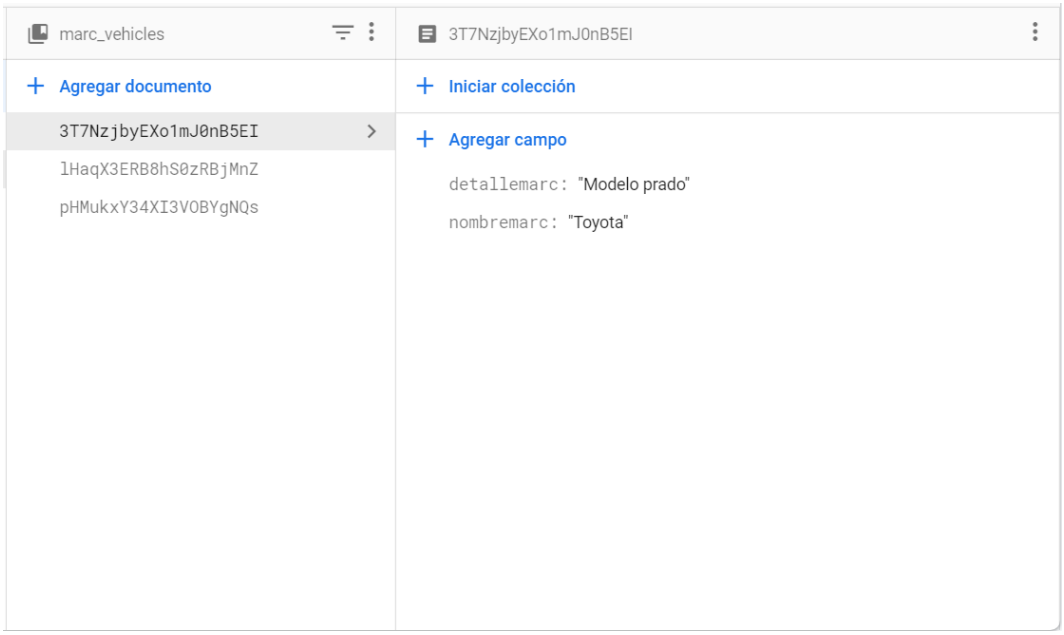


Figura 3.42. Colección marc_vehicles.
Fuente: Autores.

b.4 Colección vehic_user_register

En esta colección se guardarán los documentos creados después de que el usuario haya escogido el vehículo. Los datos que contendrán estos documentos son la fecha en la que se inicia el recorrido, id del vehículo, y también el id del usuario, en primera instancia los documentos se crearán con los siguientes campos:

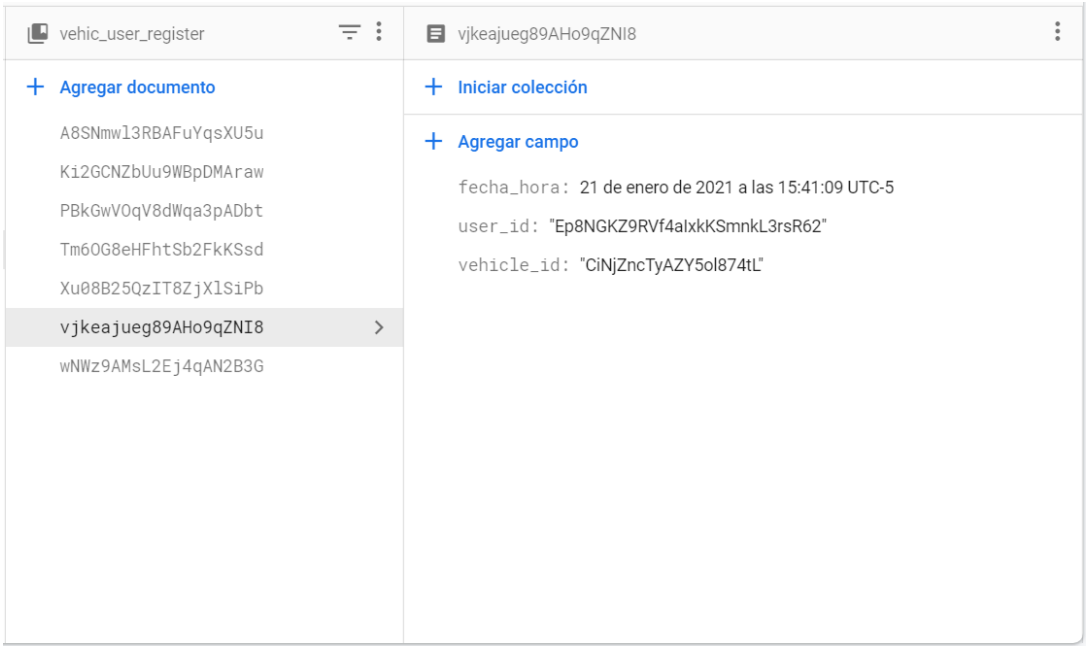


Figura 3.43. Colección vehic_user_register.
Fuente: Autores.

La colección vehic_user_register como se muestra en la figura 3.43 almacena los registros de cada recorrido y que contienen la fecha y la hora en que se inicia el

recorrido, también nos indica el id del usuario, el cual nos brinda la información del mismo, así como también el id del vehículo donde se almacena la información del vehículo que ha escogido determinado usuario.

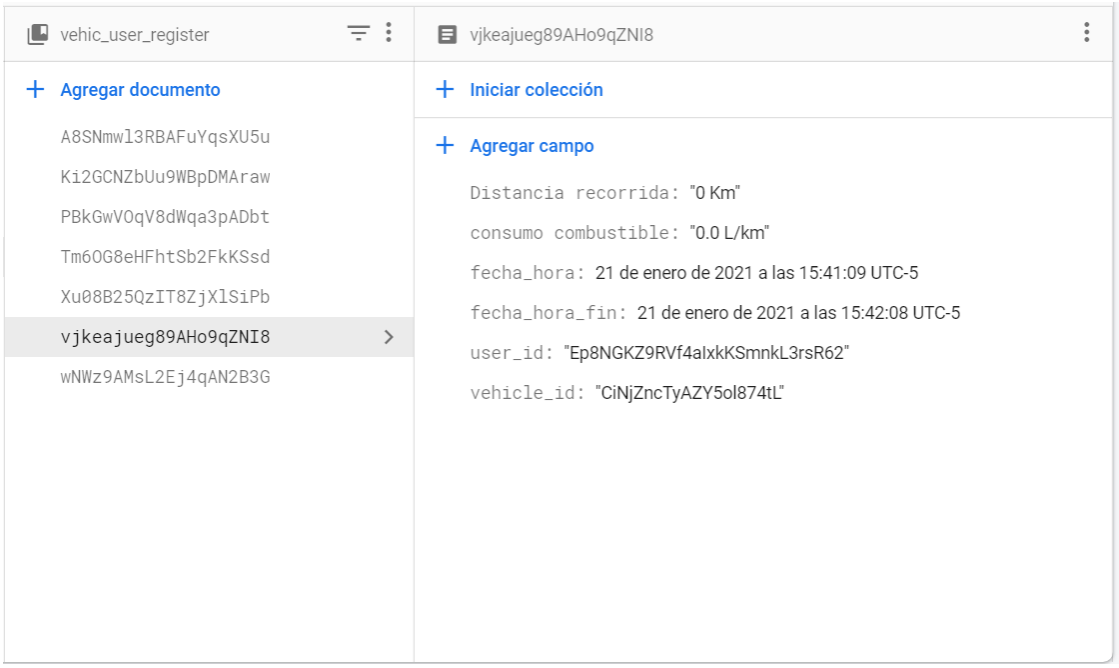


Figura 3.44. Colección vehic_user_register.
Fuente: Autores.

Cuando el usuario haya terminado el recorrido la colección vehic_user_register agregará unos campos los cuales son: fecha y hora en la que terminó el recorrido, la distancia que recorrió y también el consumo de gasolina en litros por kilómetro, como se observa en la figura 3.44

b.5 Colección det_vehic_user_register

Finalmente se tiene la colección det_vehic_user_register donde se guardan los documentos con los datos obtenidos del dispositivo de diagnóstico, la ubicación y la información de la colección vehic_user_register.

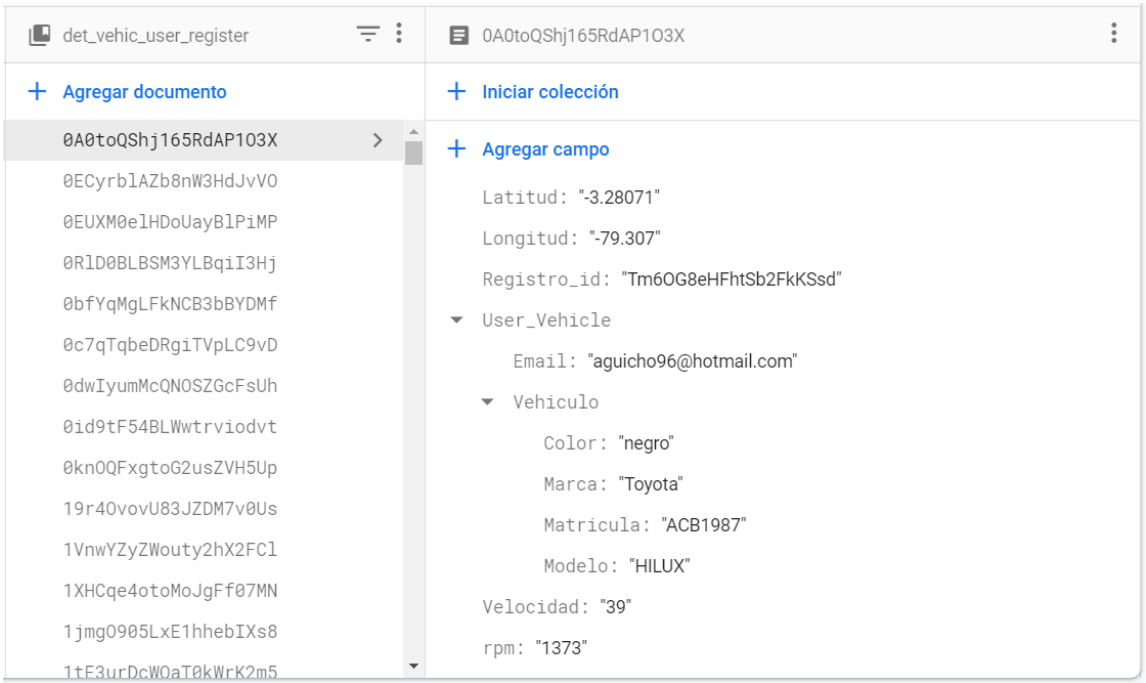


Figura 3.45. Colección det_vehic_user_register.
Fuente: Autores.

CAPÍTULO 4:

4 PRUEBAS Y RESULTADOS DEL FUNCIONAMIENTO DE LA APLICACIÓN MÓVIL

En este capítulo se presentan las pruebas del funcionamiento del proyecto recolección, análisis y envío de datos de la ECU a bordo de un vehículo para monitoreo a través de una aplicación móvil. Obtenidos y análisis de los resultados, podemos decir que se ha cumplido con los objetivos propuestos de este proyecto.

4.1 CARACTERÍSTICAS DEL VEHÍCULO UTILIZADO

La aplicación móvil desarrollada fue puesta a prueba en tres vehículos de diferentes fabricantes, en los cuales se pudo constatar que el dispositivo ELM327 no siempre obtiene las medidas requeridas de los sensores del motor. Debido al hecho de que el sistema OBD II no era estandarizado, ya que el fabricante usaba su estándar de transmisión de datos propio, por lo que algunos de los datos no son compatibles y no se pueden obtener los resultados de medición.

Por lo antes mencionado, el vehículo usado para las pruebas y obtener los resultados del funcionamiento fue una Toyota Hilux, cuyas características se muestran en la tabla 4.1.

Características	
Marca	Toyota Hilux
Año	2002
Color	Verde

Tabla 4.1. Características del vehículo.
Fuente: Autores.

4.2 PRUEBAS Y RESULTADOS DEL FUNCIONAMIENTO

4.2.1 INSTALACIÓN DEL ELM327

Para realizar las pruebas del funcionamiento se procedió a la instalación del dispositivo de diagnóstico ELM327 en el vehículo, para esto, se ubicó el conector

DLC, el cual se encontraba bajo el volante y cerca de los pedales del conductor como se observa en la figura 4.1 y como también se mencionó en el capítulo dos las posibles ubicaciones del conector DLC.

Figura 4.1. Ubicación conector DLC.
Fuente: Autores.

4.2.2 ADQUISICIÓN DE LOS DATOS CON LA APLICACIÓN MÓVIL

Las pruebas realizadas con la aplicación móvil tuvieron como objetivo principal comprobar su funcionamiento, para receptar la información a través de bluetooth, para luego procesar los datos que provienen de la ECU del vehículo y finalmente visualizar los datos obtenidos de la ECU en tiempo real.

La figura 4.2 ilustra que al iniciar la aplicación en el dispositivo Android en este caso un Samsung S10 plus, al usuario se pide los permisos necesarios para poder acceder a la ubicación del dispositivo.

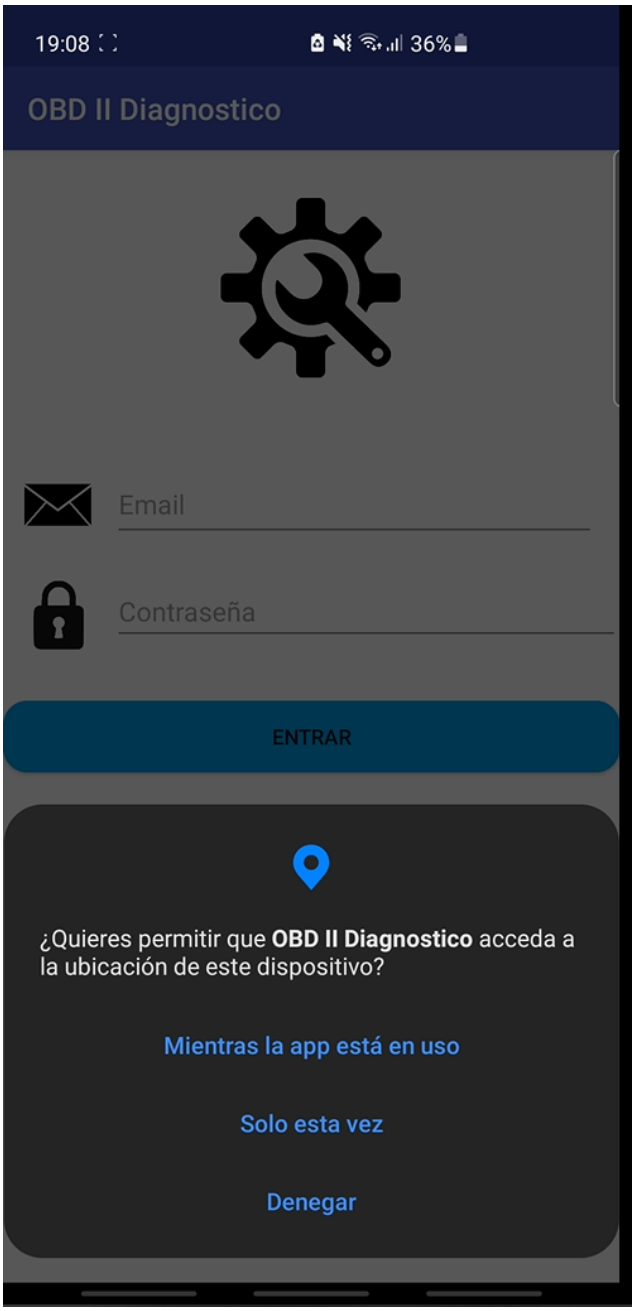


Figura 4.2. Solicitud de permisos.
Fuente: Autores.

Luego de dar los permisos a la aplicación, el usuario debe acceder con un usuario y una contraseña, estas credenciales ya fueron registradas en la base de datos de Firebase con una contraseña por defecto, la cual puede ser cambiada posteriormente si así lo desea el usuario, como se observa en la figura 4.3.

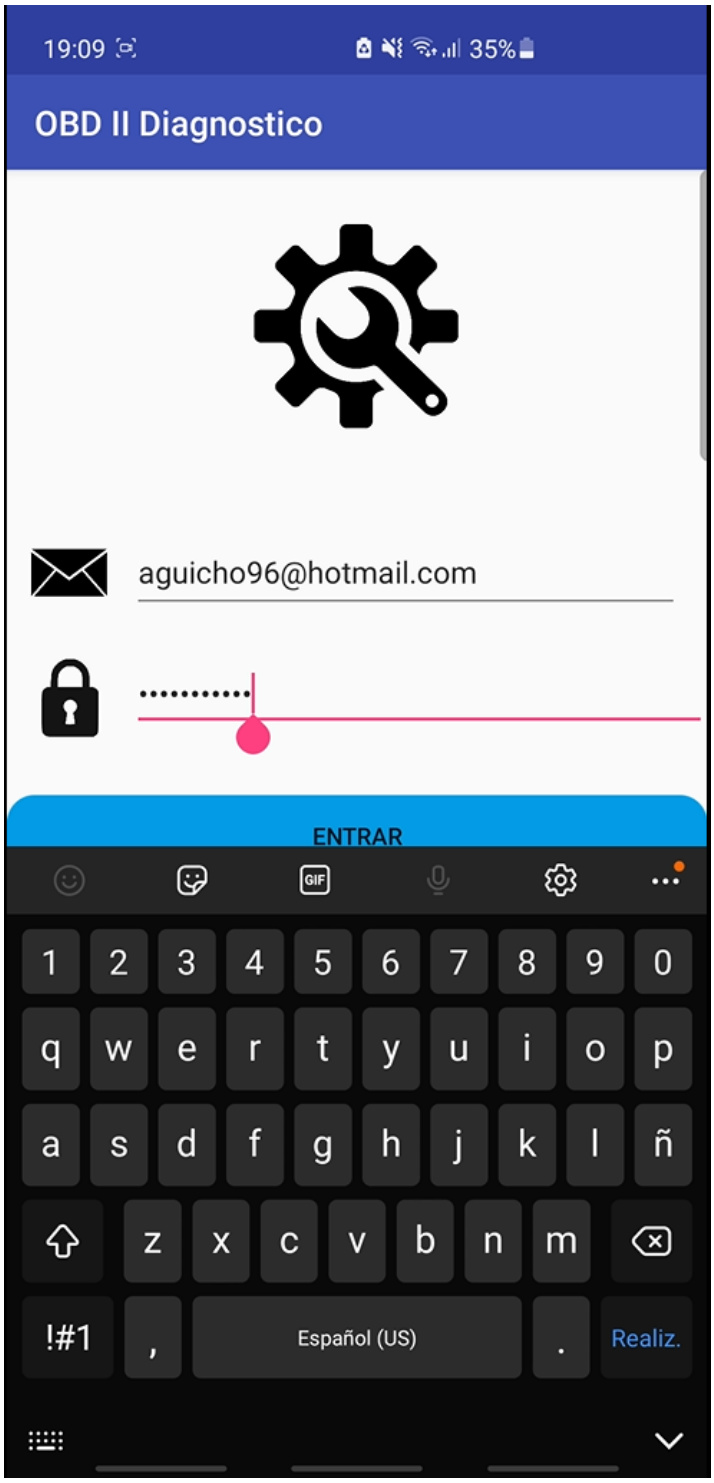


Figura 4.3. Inicio de sesión.
Fuente: Autores.

Una vez iniciado sesión la aplicación mostrará todos los autos que se hayan registrado o agregado a la base de Firebase, para este caso se ha agregado el vehículo que se ha utilizado para las pruebas del funcionamiento, como se observa en la figura 4.4.

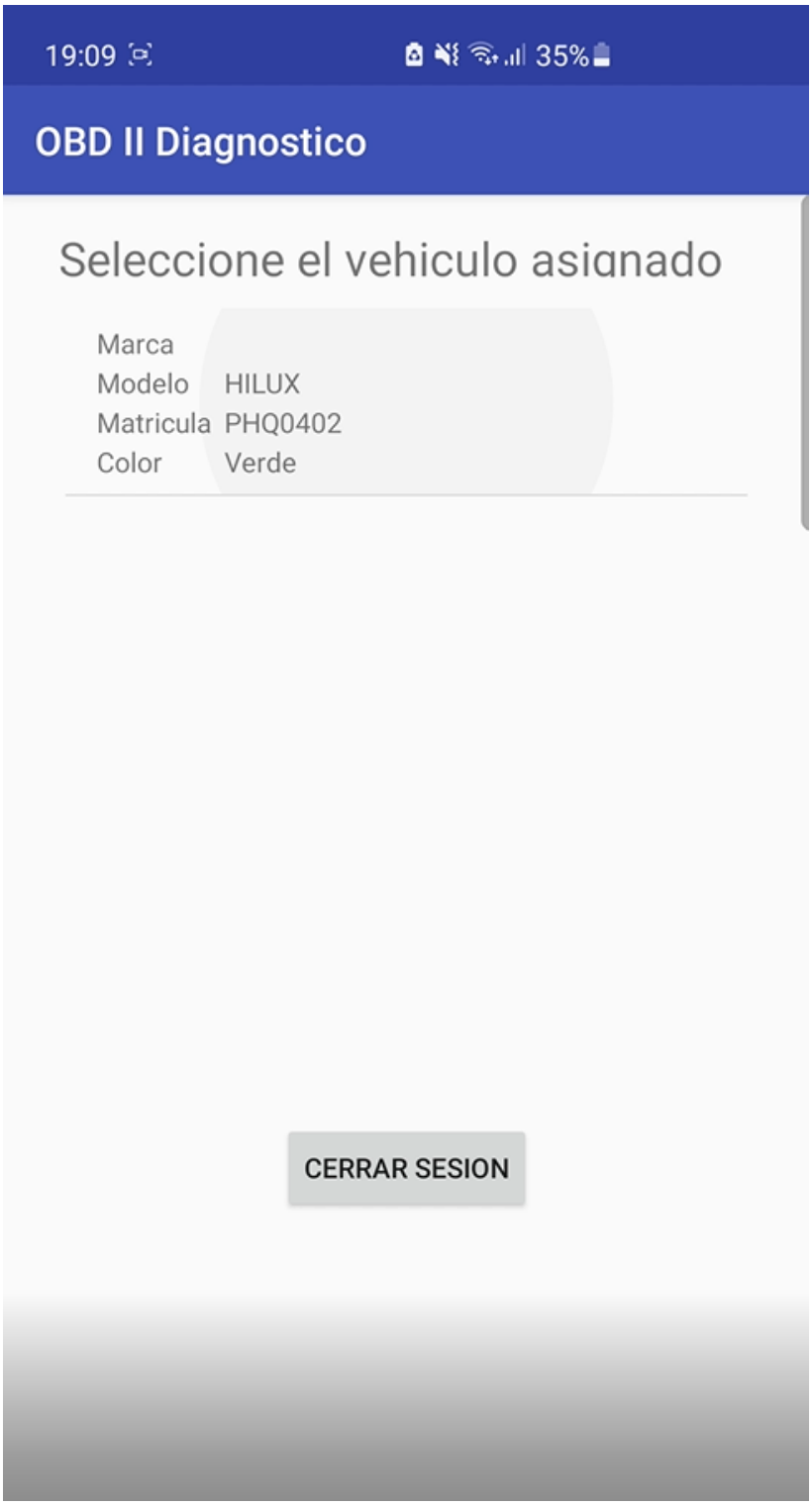


Figura 4.4. Listado de vehículos.
Fuente: Autores.

En el capítulo tres en el diseño de la aplicación móvil se mencionó que al usuario se le mostrará una pantalla en la cual debe ingresar por única vez el kilometraje del vehículo para poder llevar un control sobre el mismo y observa en la figura 4.5.

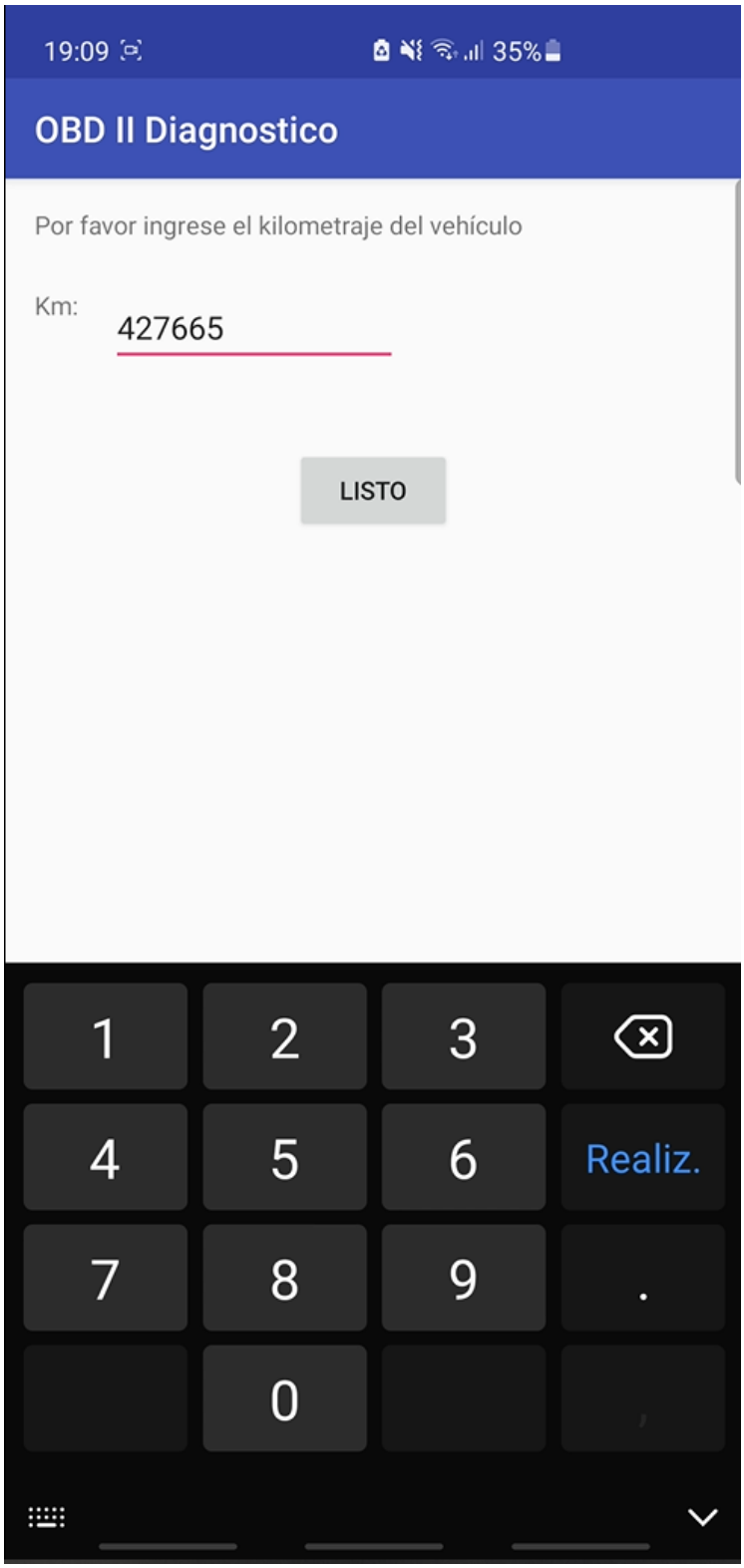


Figura 4.5. Ingreso del kilometraje.
Fuente: Autores.

Luego de ingresar el kilometraje la aplicación ejecutará en segundo plano el servicio para obtener los parámetros de la ECU en tiempo real, así como también se ejecutará el servicio de ubicación en tiempo real. Véase las figuras 4.12 y 4.13.

En la pantalla del usuario observará algo como en la figura 4.6, mientras en segundo se configura toda la conexión con el dispositivo de diagnóstico ELM327.

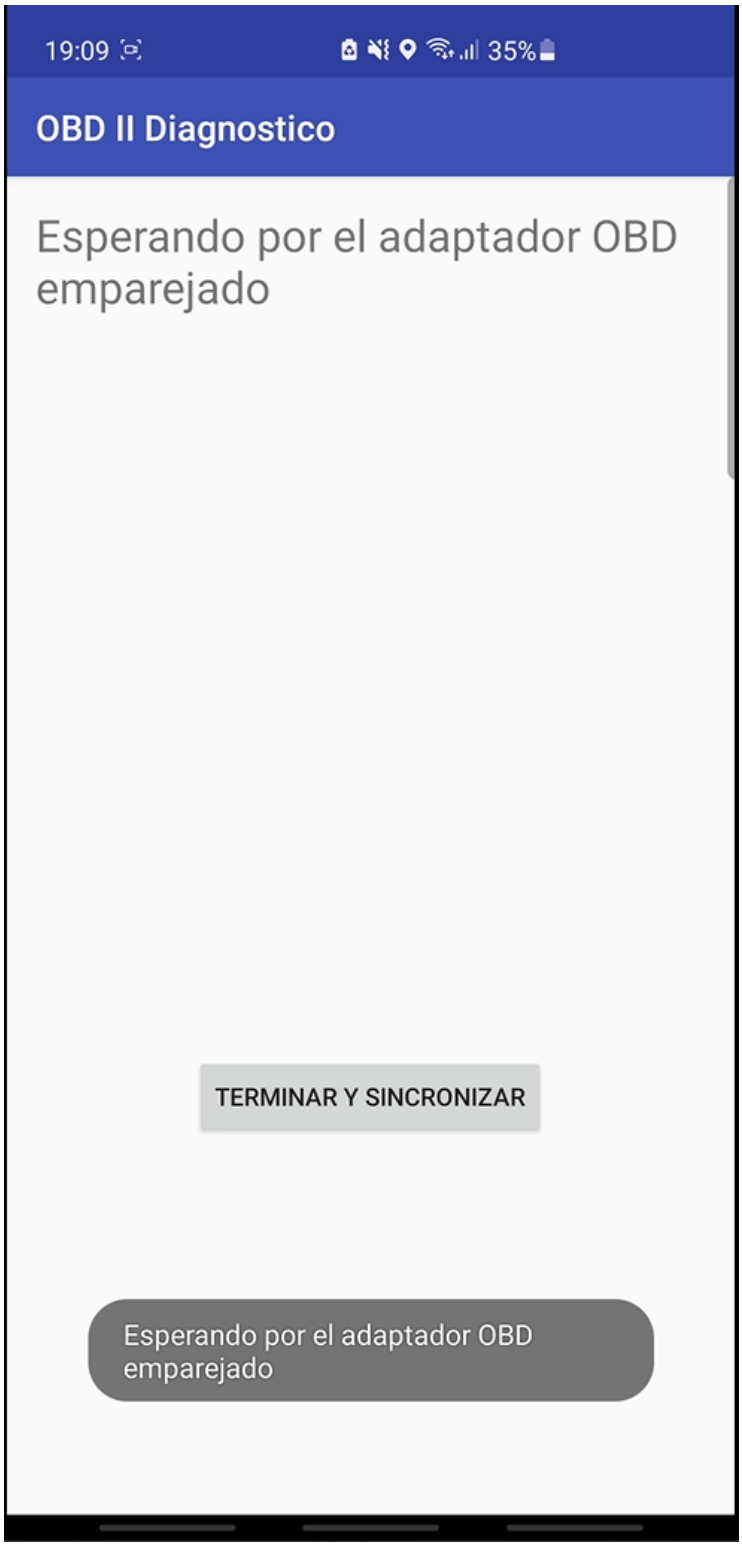


Figura 4.6. Pantalla de espera.
Fuente: Autores.

Ya que se haya terminado todo el proceso de configuración y establecer la conexión con el ELM327, el usuario verá lo siguiente:

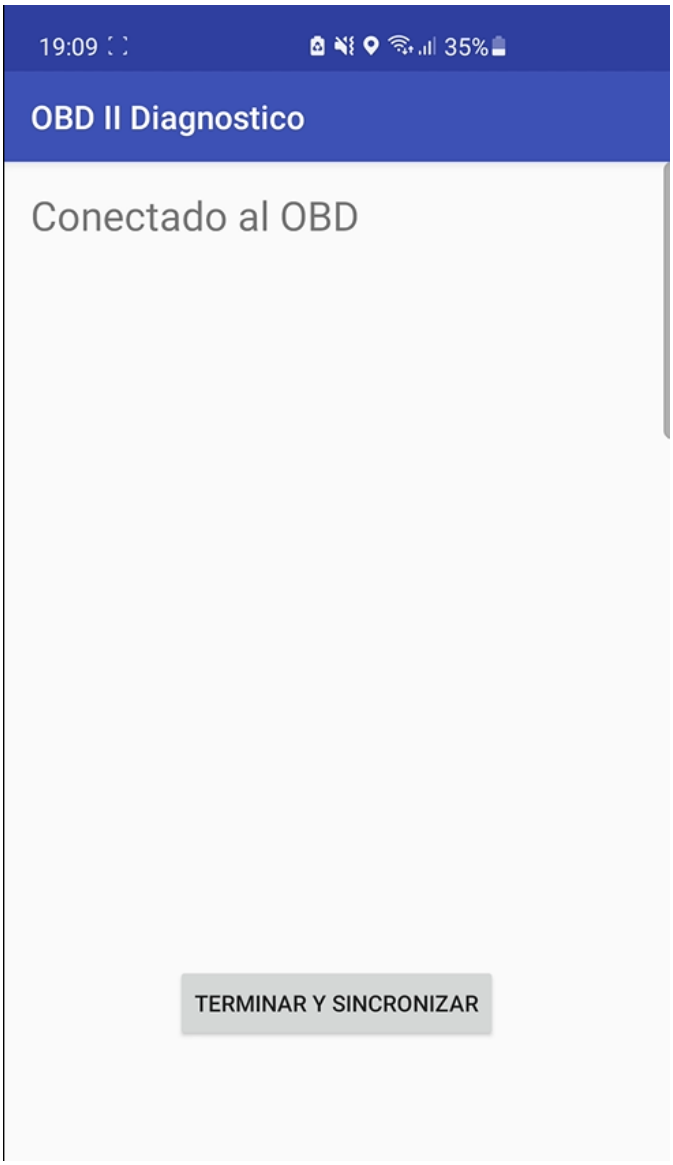


Figura 4.7. Pantalla de confirmación.
Fuente: Autores.

En la figura 4.7 se puede observar el mensaje de confirmación que se ha conectado con éxito el dispositivo ELM327.

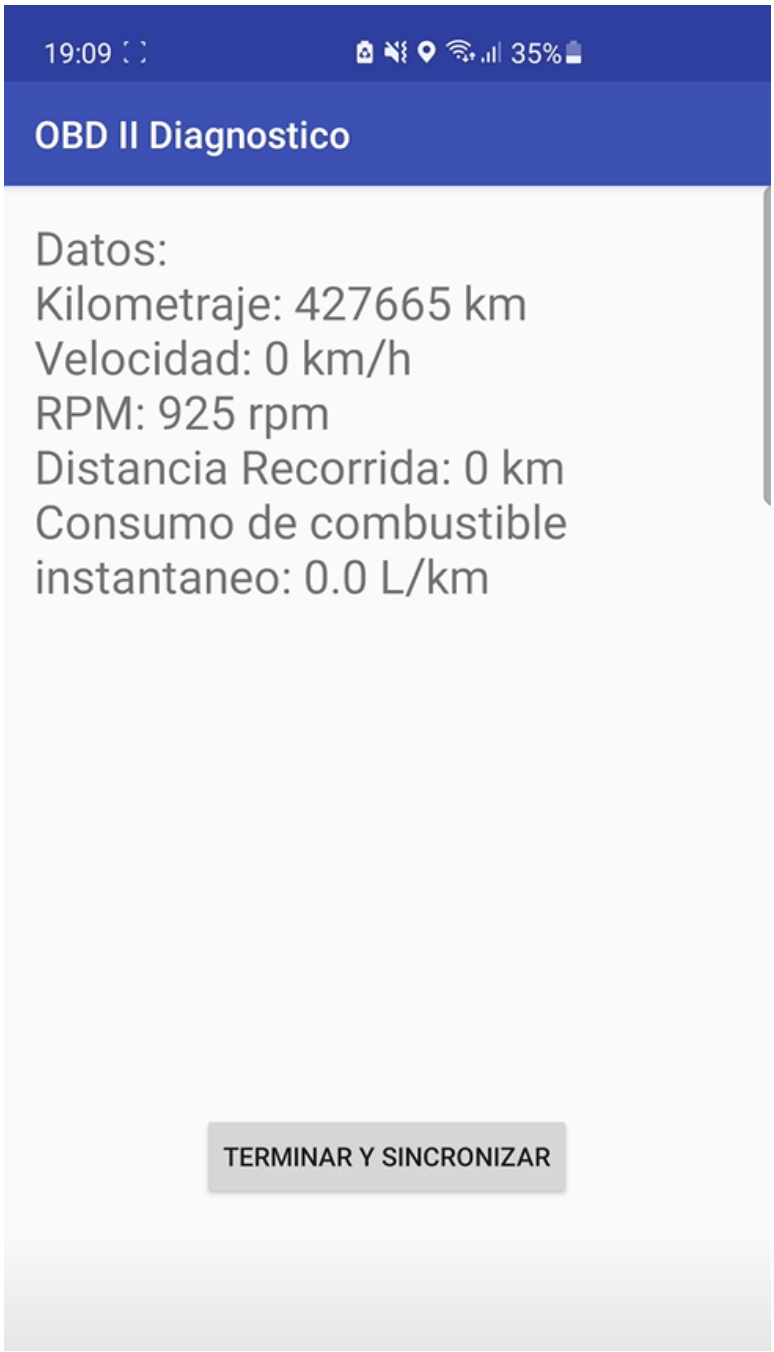


Figura 4.8. Visualización de los datos.
Fuente: Autores.

La figura 4.8 se puede observar como la aplicación móvil muestra los datos como son: el kilometraje antes ingresado de manera manual, la velocidad, las revoluciones por minuto, el consumo de combustible instantáneo del vehículo y la distancia recorrida, en este caso el vehículo se encontraba parado por lo cual en las siguientes figuras se mostrarán resultados cuando el vehículo se encontraba en movimiento.

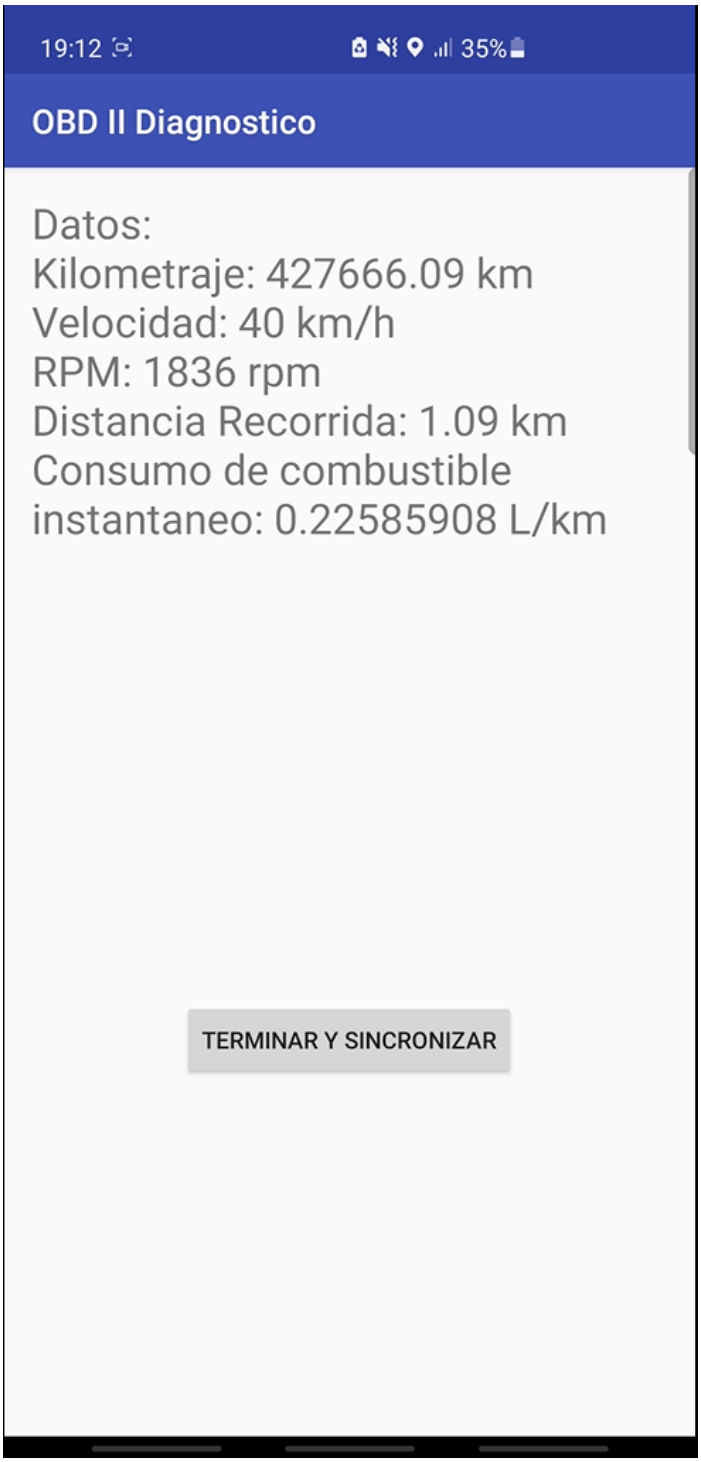


Figura 4.9. Visualización de datos en movimiento.
Fuente: Autores.

En la figura 4.9 se puede observar cómo los datos de la velocidad y el consumo de combustible se empezaron a visualizar ya que el vehículo se encontraba en movimiento y también se puede ver como el kilometraje se va sumando con la distancia que se ha recorrido.

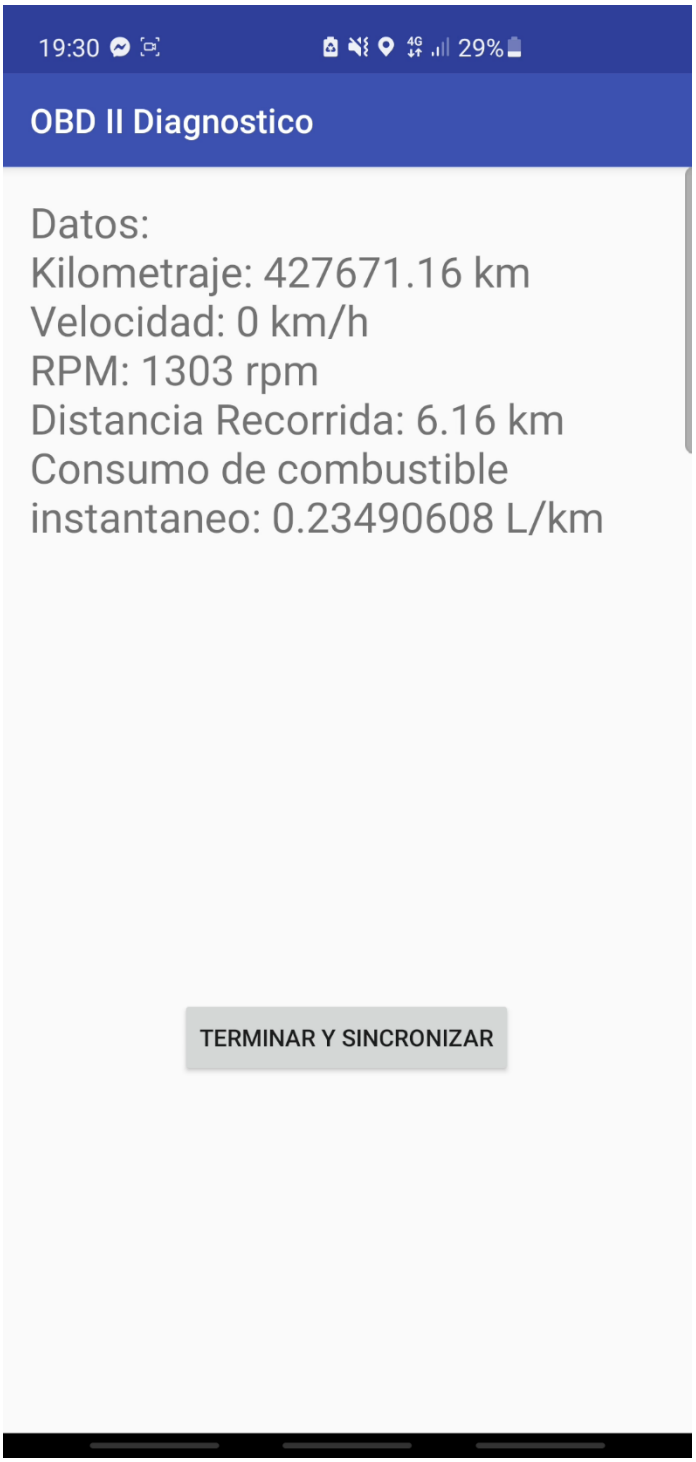


Figura 4.10. Visualización de datos terminado el recorrido.
Fuente: Autores.

La figura 4.10 muestra los datos ya cuando el vehículo ha terminado el recorrido, el cual fue de 6.16 km, como vemos esta distancia recorrida ha sido sumada al kilometraje con el que se ingresó de manera manual y estos datos finalmente serán enviados a la base de datos de Firebase.

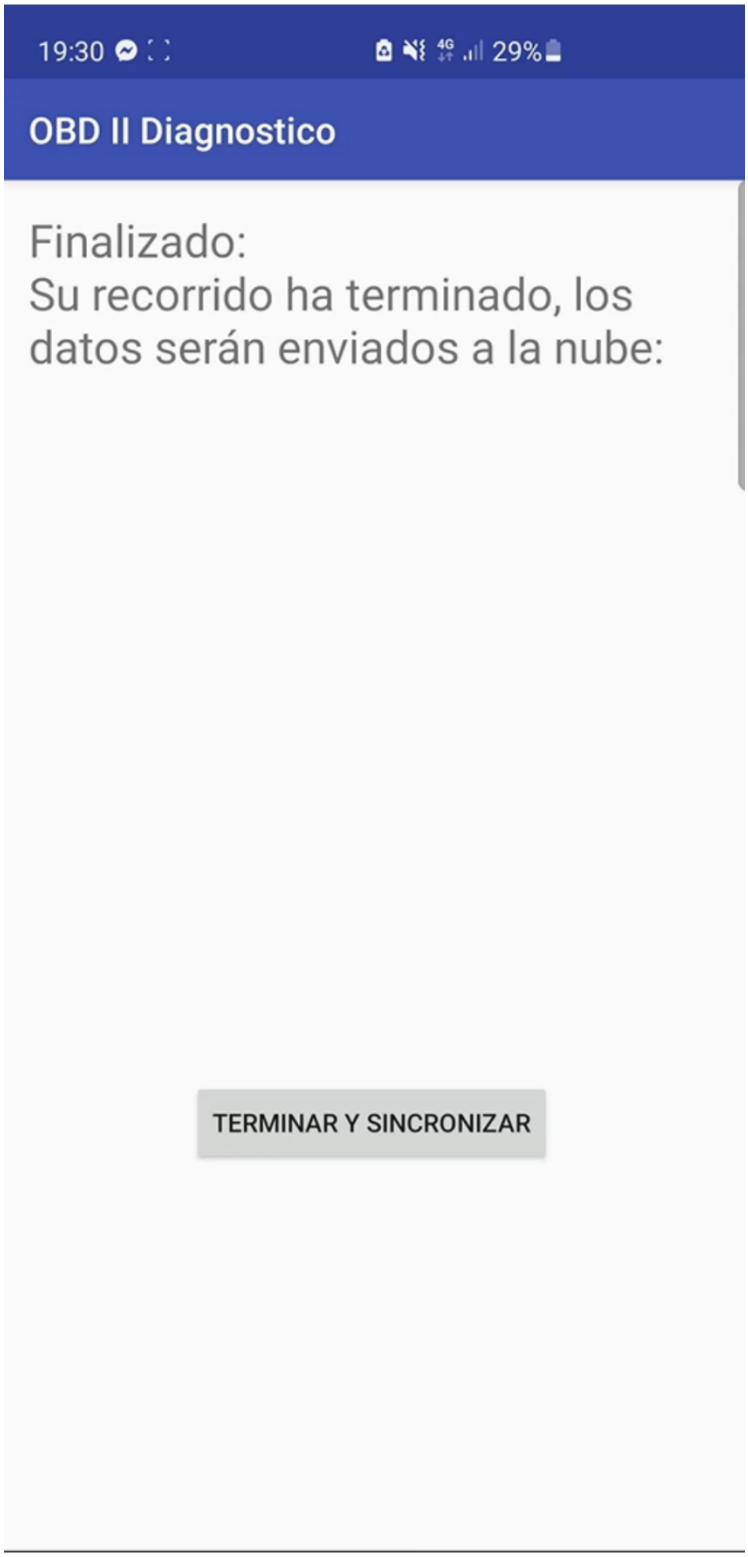


Figura 4.11. Envío de los datos.
Fuente: Autores.

Finalmente, cuando se haya terminado el recorrido el usuario verá un mensaje en la pantalla del dispositivo Android como se muestra en la figura 4.11. La aplicación móvil cuenta con dos respaldos para su correcto envío de los datos a Firebase el primer respaldo es una configuración implementada por Firebase que es la persistencia de

datos la cual funciona cuando el dispositivo Android se encuentra con una conexión inestable o inexistente a internet, permitiendo así crear una cola de solicitudes que se almacenan en la memoria caché del dispositivo Android, la cual tiene un límite de capacidad que al momento de llenarse empieza a eliminar dichas colas por tal motivo se ha creado una base de datos local que almacena todos los datos de los sensores del automóvil, entonces al sincronizar la base de datos local con la base de datos de Firebase se hace una comparación de la cantidad de datos que hay en Firebase con la base local, en caso de encontrarse una inconformidad en la cantidad de datos debido a que hubo perdida de datos durante el envío de datos en tiempo real, la aplicación móvil elimina todos los datos de Firebase que se hayan creado con el registro de ese recorrido y subirá con el mismo registro los datos de la base local. En el caso de que no se haya encontrado inconformidades en la cantidad de datos entre la base local y la de Firebase la aplicación no elimina los datos que hayan sido registrados en Firebase así evitando escribir más registros de forma innecesaria y también se evitaría salirse de los rangos gratuitos que ofrece Firebase de manera prematura al momento de escribir documentos.

4.2.3 TERMINAL DE ANDROID STUDIO

En esta sección se presentan todos los resultados que se muestran en el terminal de Android Studio durante la ejecución de la aplicación móvil.

A continuación, se muestra en la figura 4.12 y 4.13 respectivamente, el terminal de Android Studio donde se puede observar que se ha iniciado los servicios.

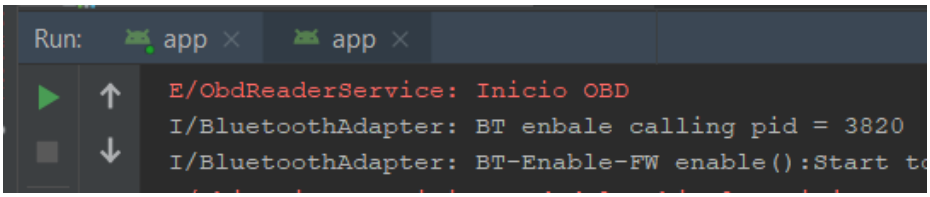


Figura 4.12. Servicio ObdReaderService.
Fuente: Autores.

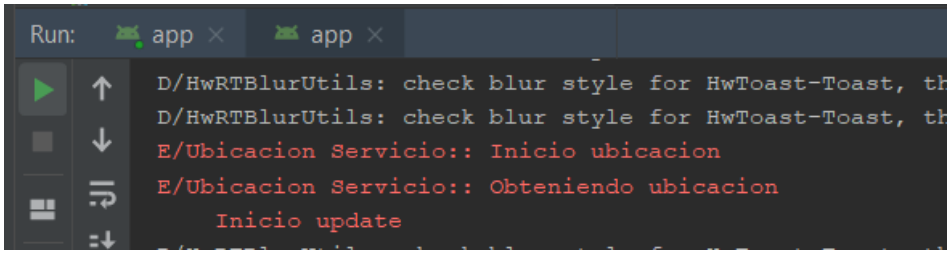


Figura 4.13. Servicio Ubicación Servicio.
Fuente: Autores.

En la figura 4.13 se muestra como el servicio para obtener la ubicación ha iniciado sin embargo este empezará a enviar las actualizaciones de la ubicación cuando el vehículo sea puesto en marcha.

El servicio de la figura 4.12 empezará establecer la conexión del dispositivo de diagnóstico ELM327, pero antes de esto el dispositivo debe emparejarse por lo general el dispositivo trae como nombre OBDII y la clave predeterminada es 1234, una vez que estén emparejados el ELM327 con el dispositivo Android el servicio establecerá la conexión con el mismo teniendo estos mensajes en el terminal de Android Studio.

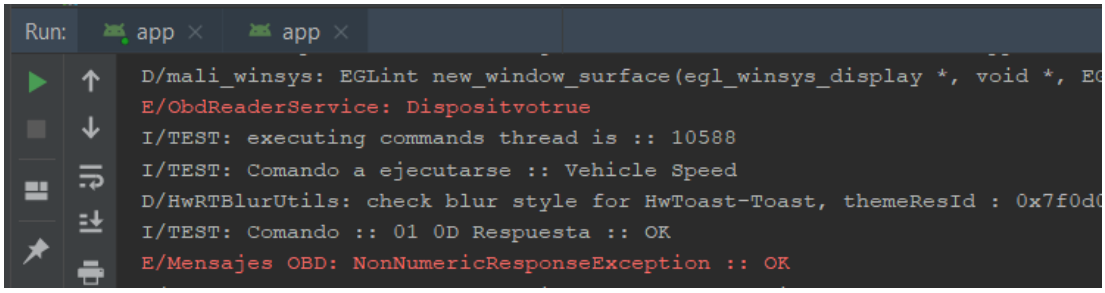


Figura 4.14. Conexión ELM327.
Fuente: Autores.

Como se muestra en la figura 4.14 el primer mensaje en color rojo nos indica que el dispositivo ha sido encontrado y establecido la conexión con el ELM327 y el segundo mensaje nos indica que no se ha encontrado errores durante la conexión con el dispositivo de diagnóstico.

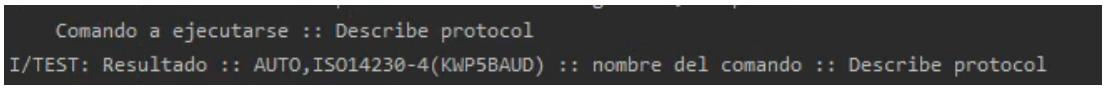


Figura 4.15. Descripción del protocolo.
Fuente: Autores.

La figura 4.15 nos muestra el resultado del protocolo con el cual el ELM327 ha establecido la comunicación con la ECU del vehículo.

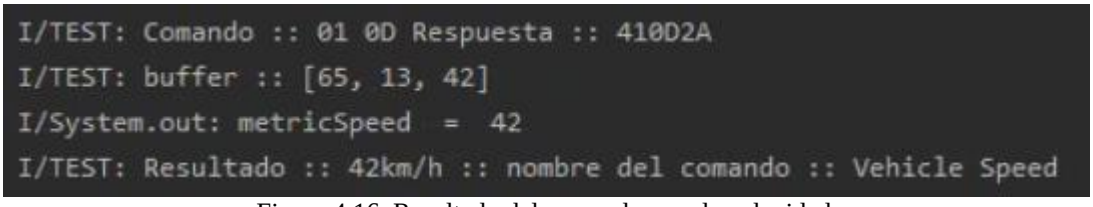


Figura 4.16. Resultado del comando para la velocidad.
Fuente: Autores.

En la figura 4.16 se muestra el test donde se visualiza el comando que se ha enviado para la velocidad “01 0D” y obteniendo su respuesta, la cual es separada en dos caracteres y almacenarse en un buffer para luego ser convertido de hexadecimal a

decimal, como se mencionó en el capítulo 2 que para obtener la velocidad se deben omitir los dos primeros pares de caracteres del buffer (65 13) e interpretar el tercer par de caracteres (42), cada PID tiene su fórmula como se indicó en la capítulo 3 en la tabla 3.1, en la cual indica que tiene como bytes de respuesta 1 y la fórmula es A dónde A equivale al byte de respuesta obtenido en el buffer que es 42 Km/h.

```
I/TEST: Comando :: 01 0C Respuesta :: 410C1E1B
    buffer :: [65, 12, 30, 27]
    Resultado :: 1926RPM :: nombre del comando :: Engine RPM
```

Figura 4.17. Resultado del comando para las RPM.
Fuente: Autores.

Para el caso de las revoluciones por minuto como se muestra en la figura 4.17 se debe aplicar la misma metodología que la velocidad, teniendo para las RPM como bytes de respuesta 2 y su fórmula mencionada en la tabla 3.1 es la siguiente:

$$\frac{256 * A + B}{4}$$

Donde A es el primer byte de respuesta y B el segundo byte de respuesta.

Aplicando la formula anterior se tiene el valor de las revoluciones del motor:

$$RPM = \frac{256 * 30 + 27}{4} \approx 1923\ RPM$$

```
I/TEST: Comando :: 01 10 Respuesta :: 41100BA1
    buffer :: [65, 16, 11, 161]
I/TEST: Resultado :: 29.77 :: nombre del comando :: Mass Air Flow
```

Figura 4.18. Resultado del comando para el sensor MAF.
Fuente: Autores.

Para obtener el resultado del sensor MAF observando la tabla 3.1 que para el MAF se tiene 2 bytes de respuesta y su fórmula es la siguiente:

$$MAF = \frac{256 * A + B}{100}$$

Donde A es el primer byte de respuesta y B el segundo byte de respuesta.

Aplicando la fórmula se obtiene:

$$MAF = \frac{256 * 11 + 161}{100} = 29.77\ g/s$$

Como siguiente paso es calcular el consumo de combustible con la ecuación 3.1

$$Fuel_{Consumption} = \left(\frac{MAF}{AFR * D * V}\right) * 3600$$

El tipo de combustible es diésel del vehículo usado para las pruebas del proyecto por lo tanto la relación aire/combustible AFR equivale a 14.5:1 y la densidad es de 850 kg/m^3.

$$Fuel_{Consumption} = \left(\frac{29.77}{14.5 * 850 * 42}\right) * 3600$$

$$Fuel_{Consumption} = 0.2070\ L/Km$$

La respuesta anterior nos indica el consumo de gasolina del vehículo en ese instante, por lo cual el consumo es muy variable ya que depende del MAF y la velocidad, para obtener un consumo aproximado durante el recorrido que se realice, la aplicación móvil guarda todos estos valores del consumo instantáneo para cuando termine el recorrido calcule un consumo promedio y este valor es enviado a la base de datos de Firebase.

El servicio de la figura 4.13 empieza a recibir actualizaciones de la ubicación cuando se pone en movimiento en el vehículo, como resultados se tiene la siguiente figura.

```
E/MyActivity: Registra Latitud: -3.2821564 Registra Longitud: -79.3132276
    Distancia total: 0.9225313549395651
E/Ubicacion Servicio:: Latitud: -3.2820677 Longitud: -79.3131744
D/Ubicacion Servicio:: VELOCIDAD: 40
    Nuevas coordenadas Latirud: -3.2820677 Longitud: -79.3131744
    Viejas coordenadas Latirud: -3.2821564 Longitud: -79.3132276
E/Ubicacion Servicio:: Distancia entre dos puntos: 11.452519
    Distancia total: 0.9339838742744178
D/MyActivity: Distancia total: 0.93
    Kilometraje: 427640.93
```

Figura 4.19. Obtención de la ubicación.
Fuente: Autores.

La figura 4.19 muestra el resultado de la obtención de las actualizaciones de la ubicación, en primera instancia se obtiene un par de coordenadas y también tiene guardado el valor de la distancia que se ha recorrido, luego obtiene unas nuevas

coordenadas y verifica que el vehículo se encuentre en movimiento para comparar entre las coordenadas anteriores con las coordenadas actuales para proceder a calcular la distancia entre estos dos puntos la cual está en metros, posteriormente sumarlas a la distancia total recorrida así como también al kilometraje y finalmente ser visualizada al usuario a través de la aplicación móvil.

```
I/TEST: service onDestroy
    socket closed ::
D/BluetoothSocket: close() this: android.bluetooth.BluetoothSocket@a20164c, mSocketState: INIT
E/Ubicacion Servicio:: terminado
W/libEGL: EGLNativeWindowType 0x792964d010 disconnect failed
```

Figura 4.20. Fin de los servicios.
Fuente: Autores.

La figura 4.20 se muestra el mensaje cuando el usuario haya terminado el recorrido los servicios para la obtención de los datos de la ECU del vehículo y la ubicación han terminado de ejecutarse para así evitar que la aplicación se detenga o pueda ocurrir cualquier error al cerrarla.

4.2.4 ENVÍO DE LOS DATOS A FIREBASE

En esta sección se mostrarán varias capturas de pantalla donde se muestran cómo se estructuran los datos en Firebase a través de colecciones como se mencionó en el capítulo 3.

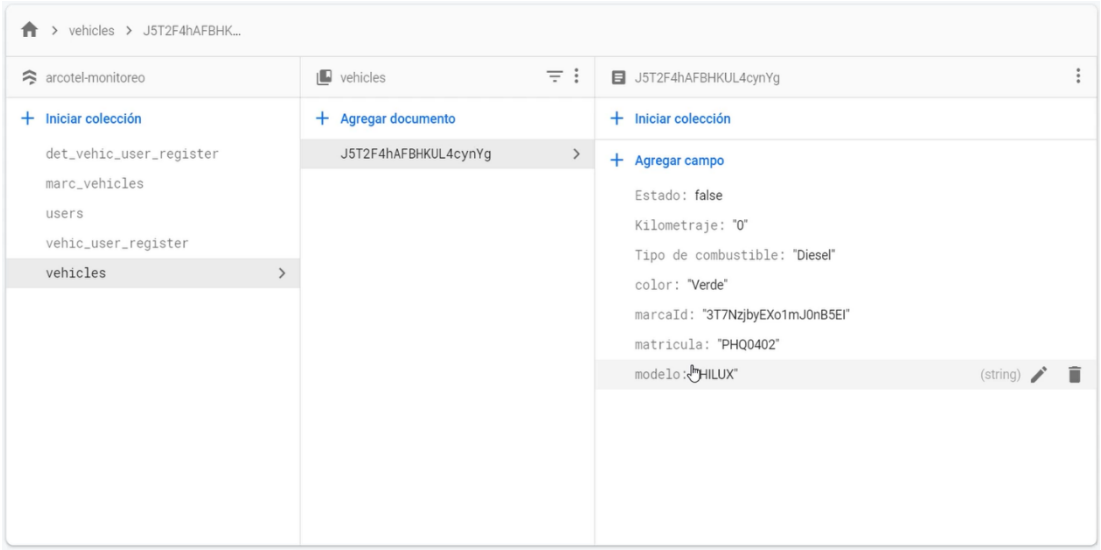


Figura 4.21. Colección vehicles.
Fuente: Autores.

La figura 4.21 nos muestra las características del vehículo, en primera instancia el dato Estado permanece en un estado falso esto indica que el vehículo aún no ha sido

seleccionado o no está en uso. El parámetro Kilometraje se encuentra en 0 debido el usuario debe ingresar por única vez y de manera manual el kilometraje del vehículo.

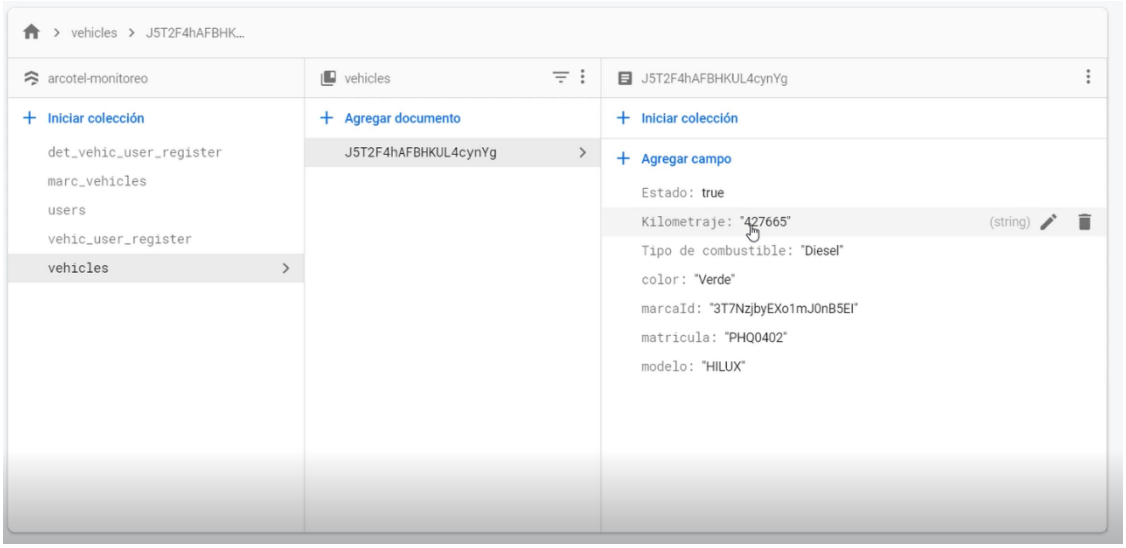


Figura 4.22. Selección del vehículo.
Fuente: Autores.

La figura 4.22 muestra que las características de la colección vehicles han cambiado luego de que el usuario ha iniciado sesión y haber seleccionado el vehículo. El dato Estado pasó de ser false a un estado true, esto indica que el vehículo está siendo usado y que otro usuario no puede seleccionar el mismo vehículo. En el parámetro Kilometraje se puede observar que el kilometraje ha sido ingresado por el usuario.

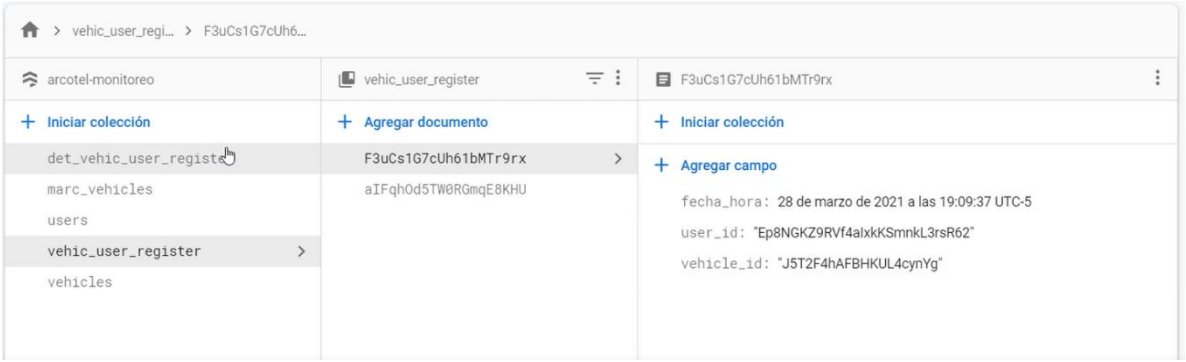


Figura 4.23. Inicio colección vehic_user_register.
Fuente: Autores.

La figura 4.23 muestra como se ha creado dentro de la colección vehic_user_register el documento donde contiene información del recorrido que se va hacer como es la fecha y hora de inicio, el usuario y el vehículo.

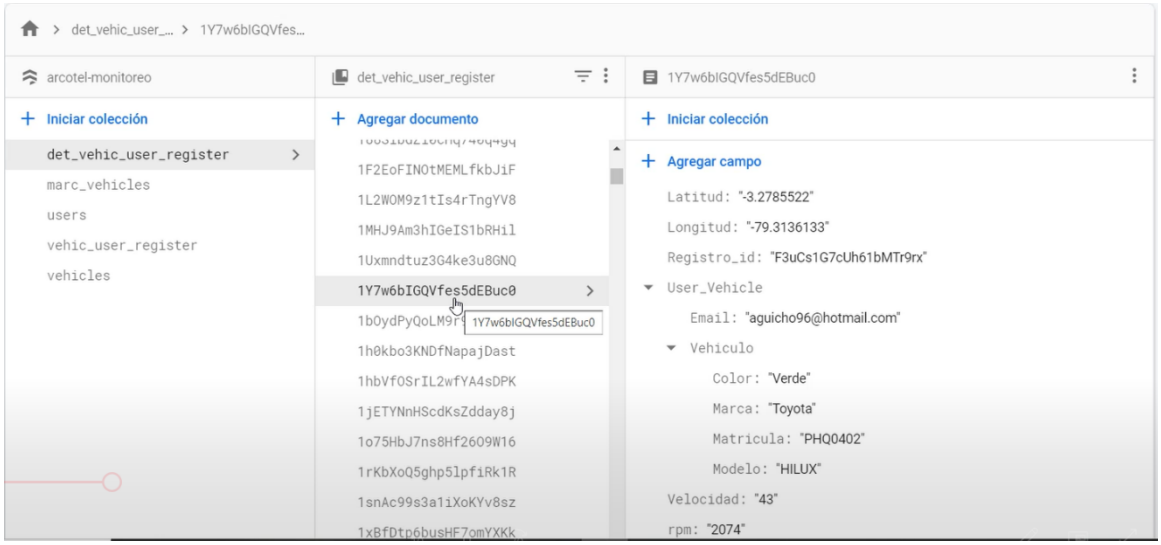


Figura 4.24. Resultados de los datos almacenados.
Fuente: Autores.

Una vez que el vehículo se ponga en movimiento los datos son enviados a Firebase, como resultado de cómo están llegando los datos se puede observar en la figura 4.24 donde se crean varios documentos que contienen información del registro de ese recorrido, usuario, vehículo y sus respectivos parámetros.

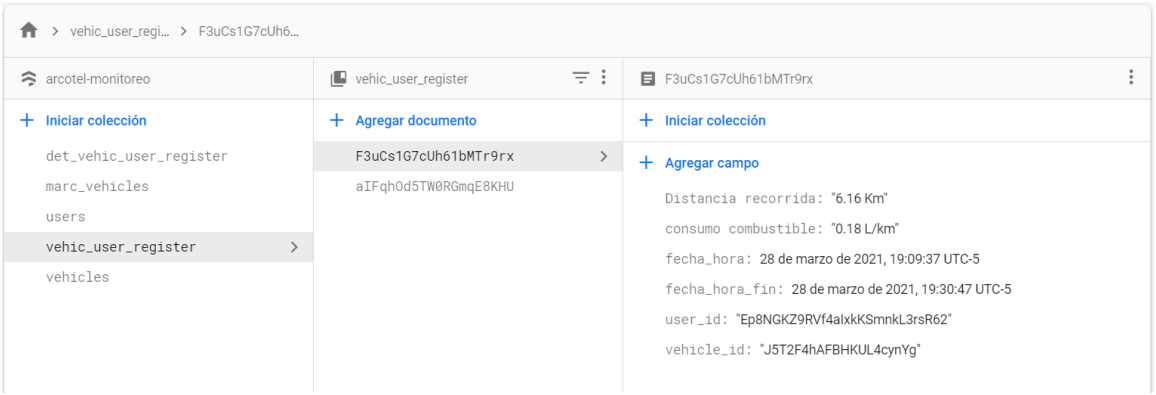


Figura 4.25. Actualización de la colección vehic_user_register.
Fuente: Autores.

Cuando el usuario termine el recorrido la colección vehic_user_register actualizará su documento en donde se agrega la distancia recorrida, consumo de combustible y por último la fecha y hora en la que se terminó el recorrido.

En la figura antes mencionada, muestra que el consumo promedio fue de 0.18 L/Km. Con este dato, al ser multiplicado por la distancia recorrida se puede obtener un cálculo aproximado del consumo total de ese recorrido. Teniendo un consumo de 1.11 L.

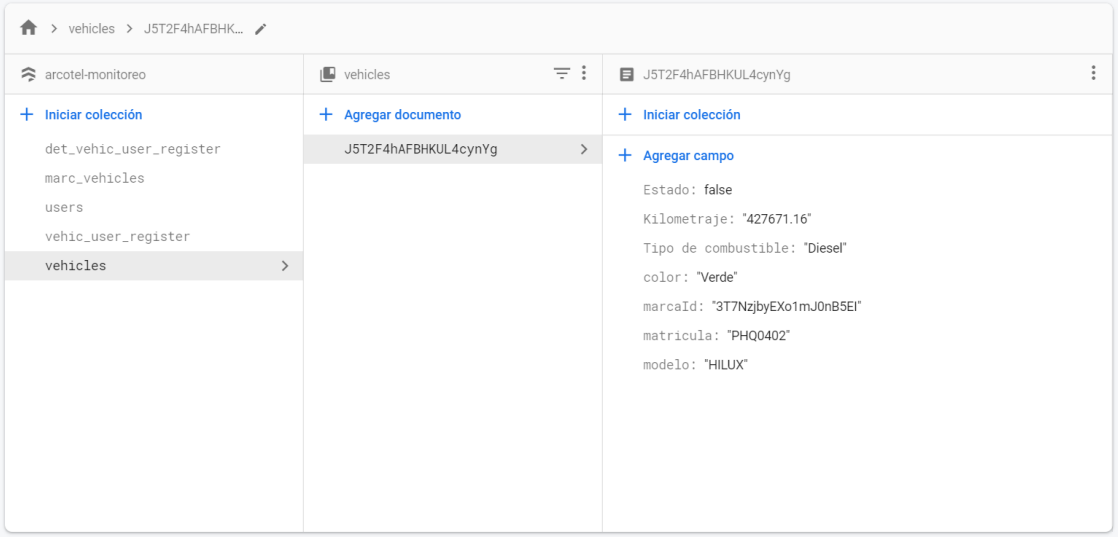


Figura 4.26. Actualización de la colección vehicules.
Fuente: Autores.

Finalmente, la colección vehicules se actualizará luego de que se haya terminado el recorrido, mostrando el kilometraje con la suma de la distancia recorrida de 6.16 Km como se muestra en la figura 4.22 y el Estado que vuelve a un estado false que indica que el vehículo no está en uso, como se puede observar en la figura 4.26.

CAPÍTULO 5:

5 CONCLUSIONES Y RECOMENDACIONES.

La aplicación fue verificada en un vehículo externo a la empresa, esto debido a que el vehículo asignado de marca Toyota modelo Prado, había sido modificado por lo que no se encontró el puerto OBD II, sino que poseía otro tipo de conector DLC.

Para comprobar el funcionamiento de la aplicación se optó en utilizar un automóvil de similares características, el automóvil es una camioneta Toyota modelo Hilux. En este vehículo se realizó con éxito todas las pruebas que se requerían, logrando obtener los parámetros deseados. Para verificar se realizó un recorrido y como evidencia se grabó videos que muestran el funcionamiento de la aplicación mientras se realizaba el recorrido.

En relación con el adaptador utilizado, siendo este de bajo costo, cabe destacar las desventajas que dependiendo de la marca del vehículo algunos datos no pueden ser leídos con el escáner ELM327 y en otros ni siquiera realizaba la conexión a la ECU.

El uso del dispositivo móvil como decodificador de datos, presentó inconvenientes en cuanto a la duración de carga de la batería, debido al gran consumo que genera procesar la información del vehículo, los datos móviles, la conexión bluetooth y el servicio de ubicación.

También el teléfono permite acceder a los servicios de posicionamiento global, logrando obtener un aproximado de la localización y distancia recorrida por el automóvil que puede ser utilizado en diferentes ámbitos de monitoreo.

La aplicación que se desarrolló puede ser instalada en diferentes dispositivos móviles Android ya que no necesita licencia alguna de desarrollador, solamente se genera un .apk, para compartir e instalar en el teléfono que se desee.

Gracias a las funcionalidades que ofrece Android Studio junto con Firebase, cuando el teléfono móvil este en zonas donde no hay conexión a internet, los datos se almacenan temporalmente en la caché de la aplicación y cuando se tenga acceso a internet se sincronice con la base de datos, evitando la pérdida de información.

Adicionalmente, la aplicación móvil cuenta con una base de datos local como respaldo de los parámetros del vehículo debido a que la memoria caché del dispositivo móvil es limitada; al saturarse empieza a eliminar datos más antiguos para dar acceso al almacenamiento de nuevos datos y para evitar esta pérdida la base de datos local se sincroniza con la base de datos de Firebase cuando se haya terminado el recorrido.

La base de datos de Firebase que se utilizó como medio para almacenar la información ayudó a tener un esquema más sistemático, logrando acceder remotamente para la consulta de datos, lo que puede facilitar la elaboración de informes.

La aplicación móvil fue diseñada únicamente para sistemas operativos Android, debido a que no es el único sistema operativo usado por los dispositivos móviles, un trabajo a futuro es la implementación de este proyecto para dispositivos con sistema operativo iOS.

Como una mejora futura se puede implementar una opción para leer los códigos de falla de vehículo, tener un historial de las fallas presentadas en la nube y también borrar dichos códigos.

El desarrollo de una página web sería un trabajo a futuro a tomar en cuenta ya que por medio de esta interfaz se podría tener una mejor visualización de la información de manera remota tales como los parámetros del vehículo, información del usuario y a través de notificaciones ayudaría a llevar acabo un mejor monitoreo y control preventivo del vehículo.

REFERENCIAS BIBLIOGRÁFICAS

[1] OBD II Scan Tool Equivalent to ISO/DIS 15031-4, SAE Standard J1978 199203, 2001.

[2] E/E Diagnostic Test Modes Equivalent to ISO/DIS 15031-5, SAE Standard J1979 200204, 2002.

[3] Pedro Mauricio González Castillo, “Diseño de una interfaz gráfica en Labview para el diagnóstico de vehículos por medio de OBD2”, Universidad Pontificia Bolivariana, Bucaramanga, 2010.

[4] Wilson Andrés Simbaña Coyago, “Diseño e implementación de una solución telemática basada en OBD-II (ON-BOARD DIAGNOSTIC) que permita obtener y procesar la información de los sensores del motor de un automóvil”, Escuela Politécnica Nacional, Quito, 2015.

[5] Lin, J., Chen, S. C., Shih, Y. T., and Chen, S. H., “A study on remote on-line diagnostic system for vehicles by integrating the technology of OBD, GPS, and 3G,”.

[6] White, J., Thompson, C., Turner, H., Dougherty, B., and Schmidt, D. C., “WreckWatch: automatic traffic accident detection and notification with smartphones,” Mobile Networks and Applications.

[7] Alex Xandra Albert Sim, Benhard Sitohang “OBD-II Standard Car Engine Diagnostic Software Development ", IEEE Trans. 2014.

[8] A. Cárdenas Sumba and M. Salavarría Tutivén, "Implementación de una plataforma de conectividad para el monitoreo en tiempo real del diagnóstico de automóviles para el mantenimiento preventivo", Ingeniería, Escuela Superior Politécnica del Litoral, 2018.

[9] F. Andrade Sánchez, "Análisis, diseño e implementación de un scanner automotriz para vehículos Volkswagen Gol, programado con arduino para visualizar en android", Ingeniería, Universidad Tecnológica Equinoccial, 2015.

[10] Pedro González Melis, “Electrónica del automóvil OBD II”, Escuela Técnica Superior de Ingeniería Industrial de Barcelona, Barcelona, 2008.

[11] A. Anchapaxi Socasi, "Recolección de datos del sistema OBD II de un automóvil usando un dispositivo Android.", Ingeniería, Escuela Politécnica Nacional, Quito, 2016.

[12] Datasheet ELM327, “OBD TO RS232 interpreter”.

[13] “PC-based Scan Tools”, OBD Software. [En línea]. Disponible en: https://www.scantool.net/scan-tools/pc-based/.

[14] G. Bernias, "Aplicación distribuida para la monitorización y diagnosis de automóviles.", Ingeniería, Universidad Politécnica de Madrid, 2015.

[15] L. Cangás Toapanta and C. Yanez Jácome, "Diseño e implementación de un módulo generador de señales y conversor para probar el comportamiento de una ECU (Unidad de Control Electrónico) para el automóvil chevrolet aveo.", Ingeniería, Universidad Politécnica Salesiana, 2015.

[16] González, C., Gajardo, J., Daniels, G., Daniels, G., Martin, R., Martin, R., Rgz, J., GORTAIRE, L., ANTONIO, E., Ojeda, A., Rivas, l. and Layes, G., 2021. Ubicación conector OBD2, donde está el conector obd. [online] Fallos OBD y OBD2, Guías Reparar Fallos OBD. Mecánica y tutoriales. Disponible en: https://teseomotor.com/ubicacion-conector-obd/.

[17] David Arroyo Cazorla, “Adquisición de datos remota mediante el sistema OBD-II y dispositivo móvil”, Universidad Carlos III de Madrid, Leganés, 2014.

[18] Gonzalo Bernias Vaquero, “Aplicación distribuida para la monitorización y diagnosis de automóviles”, Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, 2015.

[19] Vallejo, D., 2021. OBD II con ELM327 Interpretación de los Comandos OBD “Hablando al Vehículo”. [online] Issuu. Disponible en: https://issuu.com/diegoquemero/docs/subida_auto_elec_-_obd.

[20] Android Developers. 2021. Introducción a Android Studio | Desarrolladores de Android. [online] Disponible en: https://developer.android.com/studio/intro?hl=es-419.

[21] Android Developers. 2021. Descripción general de proyectos | Desarrolladores de Android. [online] Disponible en: https://developer.android.com/studio/projects?hl=es-419.

[22] GitHub. 2021. Build software better, together. [online] Disponible en: https://github.com/search?q=obd-reader.

[23] GitHub. 2021. md-sohrab-alam/android-obd-reader. [online] Disponible en: https://github.com/md-sohrab-alam/android-obd-reader.

[24] Malekian, R., Moloisane, N., Nair, L., Maharaj, B. and Chude-Okonkwo, U., 2017. “Design and Implementation of a Wireless OBD II Fleet Management System”. IEEE Sensors Journal, 17(4), pp.1154-1164.

[25] Yaguargos Castro, S., 2020. “Aplicación web progresiva (pwa) para la automatización de los procesos de gestión e información en liga deportiva parroquial totoras”. Universidad Técnica de Ambato.

[26] Firebase. 2021. Firebase Brand Guidelines. [online] Disponible en: https://firebase.google.com/brand-guidelines?hl=es.

[27] Moscoso Montenegro, D. and Romero Guaycha, D., 2020. “Desarrollo de un prototipo de radiobaliza para ciclistas, integrado en la bicicleta y el casco, con la capacidad de detección de impactos en la cabeza y emisión de alertas”. Universidad Politécnica Salesiana.

[28] Firebase. 2021. Google Analytics | Firebase. [online] Disponible en: https://firebase.google.com/docs/analytics?hl=es.

[29] VALENCIA SÁNCHEZ, J., 2018. “Desarrollo de una aplicación web que permita el monitoreo de incidentes a través de un gis, utilizando una aplicación a ser desarrollada para dispositivos móviles con sistema operativo android, que permita registrar la información georreferenciada”. Universidad Politécnica Salesiana.

[30] Firebase. 2021. Firebase Realtime Database. [online] Disponible en: https://firebase.google.com/docs/database.

[31] Firebase. 2021. Cloud Firestore | Firebase. [online] Disponible en: https://firebase.google.com/docs/firestore.

[32] Firebase. 2021. Cloud Storage for Firebase. [online] Disponible en: https://firebase.google.com/docs/storage.

[33] Firebase. 2021. Firebase Hosting. [online] Disponible en: https://firebase.google.com/docs/hosting.

[34] Firebase. 2021. Firebase Cloud Messaging. [online] Disponible en: https://firebase.google.com/docs/cloud-messaging/.

[35] Firebase. 2021. Modelo de datos de Cloud Firestore | Firebase. [online] Disponible en: https://firebase.google.com/docs/firestore/data-model?hl=es.

APÉNDICES

APÉNDICE A: HOJA DE ESPECIFICACIONES DEL CIRCUITO INTEGRADO ELM327 [12]

ELM327

OBD a RS232 intérprete

Descripción

Casi todos los automóviles producidos en la actualidad están obligados, por ley, para proporcionar una interfaz para la conexión de equipos de prueba de diagnóstico. La transferencia de datos en estas interfaces seguir varias normas, pero ninguno de ellos es directamente utilizable por los ordenadores o dispositivos inteligentes. El ELM327 está diseñado para actuar como un puente entre estos diagnósticos a bordo (OBD) puertos y una interfaz serie RS232 estándar.

Además de ser capaz de detectar automáticamente e interpretar nueve protocolos OBD, el ELM327 también proporciona soporte para comunicaciones de alta velocidad, un modo de sueño, el poder y el camión y J1939 estándar de bus. También es completamente personalizable, si desea modificar para que se ajuste más a sus necesidades.

Las siguientes páginas tratan sobre todo de las características del ELM327 en detalle, cómo usarlo y configurarlo, así como proporcionar información básica sobre los protocolos que son compatibles. También hay diagramas esquemáticos y consejos para ayudar a hacer interfaz con microprocesadores, construir una herramienta de exploración básica, y permite utilizar el modo de bajo consumo.

aplicaciones

- lectores de códigos de diagnóstico de problemas
- herramientas de análisis de automoción
- Material didáctico

Características

- Control de energía con el modo de espera
- serie universal interfaz (RS232)
- Busca automáticamente los protocolos
- Totalmente configurable con comandos AT
- Diseño de baja potencia CMOS

Diagrama de conexión
PDIP y SOIC
(vista superior)

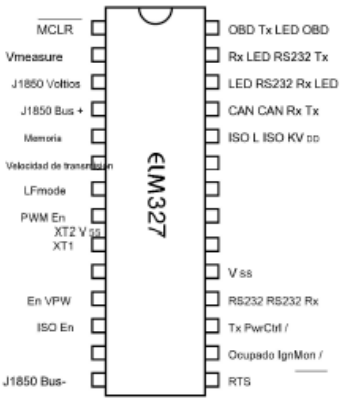
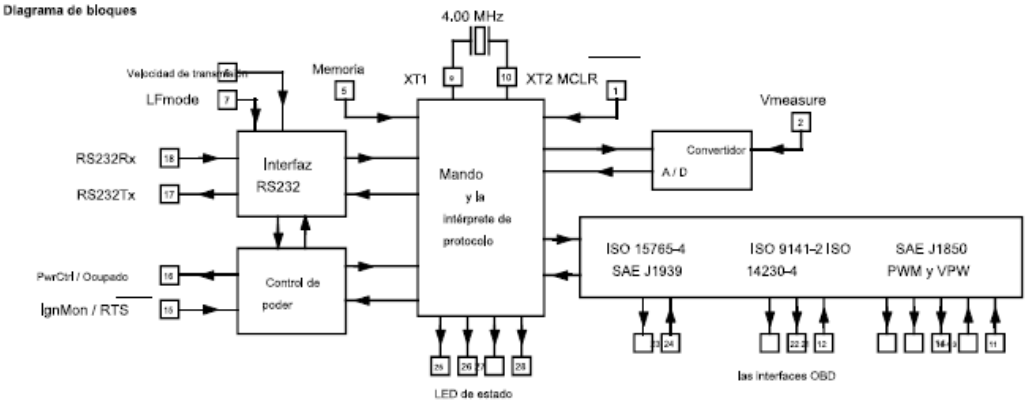


Diagrama de bloques



Características electricas

Todos los valores son para el funcionamiento a 25 ° C y una alimentación de 5V, a menos que se indique lo contrario. Para más información, consulte la nota 1 a continuación.

Característica		Máxima	mínimo	habitual		Unidades	condiciones
Tensión de alimentación, V_{DD}		4.2		5.0	5.5	V	
V_{DD} tasa de incremento		0.05				V / ms	nota 2
corriente media, me DD	normal			12		mamá	de carga
	bajo consumo de energía			0.15		mamá	
niveles lógicos de entrada	bajo	V_{SS}			0.8	V	Botones 5, 6, 7, y 24 solamente
	alto	3.0			V_{DD}	V	
los umbrales de entrada de disparador de Schmitt	creciente			2.9	4.0	V	Las patillas 1, 11, 12, 13, 15 y 18 solamente
	que cae	1.0		1.5		V	ELM327 solamente - no incluye a las corrientes
Salida de baja tensión				0.3		V	actual (sumidero) = 10 mA
Salida de alto voltaje				4.4		V	corriente (fuente) = 10 mA véase la
voltaje de reposición brown-out		2.65		2.79	2.93	V	
tiempo de conversión A / D				9		mseg	EN RV a principios de respuesta del dispositivo
Pin duración de pulso de 18 estela		128				microsegundo	despertar de modo de bajo consumo
tiempo de rebote IgnMon		50		sesenta y cinco		mseg	
En el LP a PwrCtrl tiempo de salida				1.0		segundo	
LP ALERT para el tiempo de salida PwrCtrl				2.0		seg	
Restablecer el tiempo	AT Z			800		mseg	Medida desde el extremo de la orden para el inicio de la ID de mensaje (ELM327 v2.1)
	EN WS			2		mseg	

Resumen de comandos AT

Comandos generales

<CR>	repetir el último comando
BRD hh	Velocidad de transmisión tratar Divisor hh
BRT hh	establecer la velocidad en baudios de tiempo de espera
re	establecer todos los valores predeterminados
E0, E1	Echo off, o *
FE	Olvídese de Eventos
yo	imprimir el identificador de versión
L0, L1	Saltos de línea desactivada o activada
LP	ir al modo de bajo consumo
M0, M1	Memoria desactivada o activada
RD	Leer los datos almacenados
SD hh	hh byte de datos Guardar
WS	Arranque en caliente (reinicio del software rápida)
z	resetear todo
@ 1	mostrar la descripción del dispositivo
@ 2	visualizar el identificador de dispositivo
@ 3 ccccccccccc	almacenar el identificador @ 2

Comandos de parámetros programables

OFF PP xx Prog desactivar Parámetro xx

PP FF OFF todos los parámetros Prog desactivados

PP XX sobre permitir xx Parámetro Prog

PP FF EN todos los parámetros Prog habilitado

PP xx yy SV Para XX PP, establezca el valor de yy

PPS imprimir un resumen PP

Los comandos de lectura de voltaje

CV dddd Calibrar el voltaje a dd.dd voltios

CV 0000 restaurar el valor CV al ajuste de fábrica

RV Leer la tensión de entrada

Otro

IGN leer el nivel de entrada IgnMon

Resumen de comandos AT (continuación)

Comandos OBD

Alabama	Permitir largas (> 7 bytes) mensajes
AMC	Recuento en pantalla Monitor de Actividad
AMT hh	establecer el tiempo de espera Actividad lun a SS
Arkansas	Recibe automáticamente

AT0, 1, 2 El tiempo de adaptación fuera, auto1 *, auto2

BD	realizar un volcado Buffer
BI	Omitir la secuencia de inicialización
DP	Describir el protocolo actual
DPN	Describir el Protocolo por el Número
H0, H1	Encabezados apagado *, o en
MAMÁ	Monitor de toda la
MR hh	Monitor para el receptor = hh
MT hh	Monitor para el transmisor = hh
NL	mensajes de longitud normal *

ordenador personal Protocolo Cerrar

R0, R1	Respuestas desactivada o activada *
RA hh	establecer el Recibir dirección a SS
S0, S1	la impresión de Espacios desactivada o activada *
xyz SH	Conjunto de la cabecera a XYZ

xxyyzz SH Conjunto de la cabecera a xxyyzz

wwxxyyzz SH Conjunto de la cabecera a wwxxyyzz

SP h	Establecer Protocolo h y guardarlo
Ah SP	Establecer Protocolo en Auto, h y guardarlo
SP 00	Borrar los protocolos almacenados
hh SR	Establecer la dirección Recibir a SS
SS	utilizar el orden de búsqueda estándar (J1978)
ST hh	Establecer tiempo de espera a HH x 4 ms
TA hh	set probador Dirección a SS
TP h	Trate Protocolo h
Ah TP	Trate Protocolo h con búsqueda automática

Comandos específicos J1850 (los protocolos 1 y 2)

IFR0, 1, 2	IFR apagado, automático *, o en
IFR H, S	IFR valor de la cabecera * o Fuente

Comandos específicos ISO (protocolos de 3 a 5)

FI	realizar una iniciación rápida
IB 10	establecer la velocidad ISO Baud a 10.400 *
IB 48	establecer la velocidad ISO Baud a 4800
IB 96	establecer la velocidad ISO Baud a 9600
IIA hh	establecer ISO (lento) Init Dirección a SS
KW	mostrar las palabras clave

KW0, KW1 Palabra clave de verificación desactivada o activada *

SI	realizar una iniciación (5 baudios) Slow
SW hh	Establecer intervalo de despertador a HH x 20 mseg
SW 00	Detener el envío de mensajes de Despertar

WM [1 - 6 bytes] establecer el mensaje de despertador

CAN (Comandos específicos protocolos de 6 a C)

CEA	apague CAN direccionamiento extendido
CEA hh	uso puede extendido hh Dirección

CAF0, CAF1 Formato automático desactivada o activada *

CF hhh	configurar el filtro de ID a hhh

CF hhhhhhhh configurar el filtro de ID a hhhhhhhh

CFC0, CFC1 Controles de flujo desactivada o activada *

hhh CM	configurar la máscara de ID a hhh

hhhhhhhh CM configurar la máscara de ID a hhhhhhhh

hh CP	PUEDE conjunto prioridad a SS (29 bits)
CRA	restablecer el Recibe filtros de direcciones
hhh CRA	el aparato puede recibir direcciones para hhh

hhhhhhhh CRA configurar la Dirección Rx a hhhhhhhh

CS	mostrar los recuentos de estado pueden

CSM0, CSM1 Supervisión silenciosa desactivada o activada *

CTM1	programar el temporizador de multiplicador de 1 *
CTM5	programar el temporizador Multiplicador a 5

Resumen de comandos AT (continuación)

PUEDE comandos específicos (continuación)

D0, D1	pantalla del DLC apagado *, o en
FC SM h	Flujo de control, ajuste el modo en h
FC SH hhh	FC, establecer la cabecera a HHH
FC SH hhhhhhhh	Establecer la cabecera de hhhhhhhh
FC SD [1 - 5 bytes]	FC, conjunto de datos a [...]
PB xx yy	Opciones Protocolo B y velocidad de transmisión
RTR	enviar un mensaje de RTR
V0, V1	uso de DLC variable apagado *, o en

Comandos específicos J1939 (CAN protocolos de A a C)

DM1	supervisar los mensajes DM1
JE	utilizar el formato de datos J1939 Elm *
JHF0, JHF1	Formateo de cabecera desactivada o activada *
JS	utilizar el formato de datos SAE J1939
JTM1	programar el temporizador de multiplicador de 1 *
JTM5	programar el temporizador Multiplicador a 5

hhhh MP Monitor para PGN 0hhhh

MP hhhh n " "Y obtener n mensajes

hhhhhh MP Monitor para PGN hhhhhh

MP HHHHHH n " "Y obtener n mensajes

APÉNDICE B: PID MODO 01 OBD II [4]

PID (HEX)	BYTES DE DATOS DEVUELTO	DESCRIPCIÓN	UNIDADES	FÓRMULA
00	4	PIDs Soportados		Bit codificado [A7..D0]==[PID 0x01..PID 0x20]
01	4	Número de código de problema e información I/M		Bit codificado. Véase más abajo.
02	8	DTC congelado		
03	2	Estado del sistema de combustible		Bit codificado. Véase más abajo.
04	1	Valor calculado de carga del motor	%	A*100/255
05	1	Temperatura del refrigerante del motor	ºC	A-40
06	1	Combustible a corto plazo %reducido-Banco 1	%	0.7812*(A-128)
07	1	Combustible a largo plazo %reducido-Banco 2	%	0.7812*(A-128)
08	1	Combustible a corto plazo %reducido-Banco 1	%	0.7812*(A-128)
09	1	Combustible a largo plazo %reducido-Banco 2	%	0.7812*(A-128)
0A	1	Presión del combustible	KPa	A*3
0B	1	Presión del colector de admisión	KPa	A
0C	2	RPM del motor	Rpm	((A*256)+B)/4
0D	1	Velocidad del automóvil	Km/h	A

PID (HEX)	DATOS DEVUELTO	DESCRIPCIÓN	UNIDADES	FÓRMULA
0E	1	Sincronización de avance	Relativo al cilindro #1	(A-128)/2
0F	1	Temperatura del aire de admisión	ºC	A-40
10	2	Flujo de aire del MAF	g/s	((256*A)+B)/100
11	1	Posición de la válvula reguladora	%	A*100/255
12	1	Estado del aire secundario mandado		Bit codificado. Véase más abajo
13	1	Sensores de oxígeno presentes		[A0..A3]==Banco1, Sensores 1-4. [A4..A7]==Banco2..
14	2	Banco 1, Sensor 1: Voltaje del sensor de oxígeno, ajuste a corto plazo del combustible	Voltios %	A/100 (B-128)*100/128 (Si B==0xFF, sensor no usado en el ajuste calculado)
15	2	Banco 1, Sensor 2: Voltaje del sensor de oxígeno, ajuste a corto plazo del combustible	Voltios %	A/100 (B-128)*100/128 (Si B==0xFF, sensor no usado en el ajuste calculado)
16	2	Banco 1, Sensor 3: Voltaje del sensor de oxígeno, ajuste a corto plazo del combustible	Voltios %	A/100 (B-128)*100/128 (Si B==0xFF, sensor no usado en el ajuste calculado)
17	2	Banco 1, Sensor 4: Voltaje del sensor de oxígeno, ajuste a corto plazo del combustible	Voltios %	A/100 (B-128)*100/128 (Si B==0xFF, sensor no usado en el ajuste calculado)
18	2	Banco 2, Sensor 1: Voltaje del sensor de oxígeno, ajuste a corto plazo del combustible	Voltios $	A/100 (B-128)*100/128 (Si B==0xFF, sensor no usado en el ajuste calculado)

PID (HEX)	DATOS DEVUELTO	DESCRIPCIÓN	UNIDADES	FÓRMULA
19	2	Banco 2, Sensor 2: Voltaje del sensor de oxígeno, ajuste a corto plazo del combustible	Voltios %	A/100 (B-128)*100/128 (Si B==0xFF, sensor no usado en el ajuste calculado)
1A	2	Banco 2, Sensor 3: Voltaje del sensor de oxígeno, ajuste a corto plazo del combustible	Voltios %	A/100 (B-128)*100/128 (Si B==0xFF, sensor no usado en el ajuste calculado)
1B	2	Banco 2, Sensor 4: Voltaje del sensor de oxígeno, ajuste a corto plazo del combustible	Voltios %	A/100 (B-128)*100/128 (Si B==0xFF, sensor no usado en el ajuste calculado)
1C	1	Los estándares OBD del automóvil se ajustan a		Bit codificado. Véase más abajo
1D	1	Sensores de oxígeno presentes		Similar al PID13, pero [A0..A7]==[B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2]
1E	1	Estado de la entrada auxiliar		A0==toma de alimentación apagada (PTO) estado (1==activo) [A1..A7] no usada
1F	2	Tiempo de ejecución desde el arranque del motor	Segundos	(A*256)+B
20	4	PIDs soportados 21-40		Bit codificado [A7..D0]==[PID 0x21..PID 0x40]
21	2	Distancia recorrida con la luz MIL encendida	Km	(A*256)+B

(HEX)	DATOS DEVUELTO	DESCRIPCION	UNIDADES	FORMULA
22	2	Carril de presión de combustible (en relación con el vacío del colector)	KPa	((A*256)+B)*0.079
23	2	Carril de presión de combustible (diesel)	KPa	((A*256)+B)*10
24	4	O2S1_WR_lambda (1) Relación de equivalencia de voltaje	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
25	4	O2S2_WR_lambda (1) Relación de equivalencia de voltaje	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
26	4	O2S3_WR_lambda (1) Relación de equivalencia de voltaje	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
27	4	O2S4_WR_lambda (1) Relación de equivalencia de voltaje	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
28	4	O2S5_WR_lambda (1) Relación de equivalencia de voltaje	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
29	4	O2S6_WR_lambda (1) Relación de equivalencia de voltaje	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
2A	4	O2S7_WR_lambda (1) Relación de equivalencia de voltaje	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
2B	4	O2S8_WR_lambda (1) Relación de equivalencia de voltaje	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
2C	1	Mando EGR	%	100*A/255
2D	1	Error EGR	%	(A-128)*100/128
2E	1	Mando de purgación	%	A*100/255

PID (HEX)	BYTES DE DATOS DEVUELTO	DESCRIPCIÓN	UNIDADES	FÓRMULA
2F	1	Nivel de combustible de entrada	%	A*100/255
30	1	# de calentamientos desde el borrado de códigos	N/A	A
31	2	Distancia recorrida desde el borrado de códigos	Km	(A*256)+B
32	2	Presión del sistema de evaporación de gases	Pa	((A*256)+B)/4
33	1	Presión barométrica	KPa	A
34	4	O2S1_WR_lambda (1) Relación de equivalencia de corriente	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256-128
35	4	O2S2_WR_lambda (1) Relación de equivalencia de corriente	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256-128
36	4	O2S3_WR_lambda (1) Relación de equivalencia de corriente	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256-128
37	4	O2S4_WR_lambda (1) Relación de equivalencia de corriente	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256-128
38	4	O2S5_WR_lambda (1) Relación de equivalencia de corriente	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256-128

(HEX)	DATOS DEVUELTO	DESCRIPCIÓN	UNIDADES	FÓRMULA
39	4	O2S6_WR_lambda (1) Relación de equivalencia de corriente	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256-128
3A	4	O2S7_WR_lambda (1) Relación de equivalencia de corriente	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256-128
3B	4	O2S8_WR_lambda (1) Relación de equivalencia de corriente	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256-128
3C	2	Temperatura del catalizador Banco 1, Sensor 1	ºC	((A*256)+B)/10-40
3D	2	Temperatura del catalizador Banco 2, Sensor 1	ºC	((A*256)+B)/10-40
3E	2	Temperatura del catalizador Banco 1, Sensor 2	ºC	((A*256)+B)/10-40
3F	2	Temperatura del catalizador Banco 2, Sensor 2	ºC	((A*256)+B)/10-40
40	4	PIDs soportados [41-60]		Bit codificado [A7..D0]==[PID 0x41..PID 0x60]
41	4	Estado del monitoreo del ciclo de manejo		Bit codificado
42	2	Módulo de control de voltaje	V	((A*256)+B)/1000
43	2	Valor de carga absoluta	%	((A*256)+B)*100/255

PID (HEX)	BYTES DE DATOS DEVUELTO	DESCRIPCIÓN	UNIDADES	FÓRMULA
44	2	Comando de relación de equivalencia	N/A	((A*256)+B)/32768
45	1	Posición relativa de la válvula reguladora	%	A*100/255
46	1	Temperatura del aire del ambiente	ºC	A-40
47	1	Posición absoluta B de la válvula reguladora	%	A*100/255
48	1	Posición absoluta C de la válvula reguladora	%	A*100/255
49	1	Posición D del pedal acelerador	%	A*100/255
4A	1	Posición E del pedal acelerador	%	A*100/255
4B	1	Posición F del pedal acelerador	%	A*100/255
4C	1	Comando del actuador de la válvula reguladora	%	A*100/255
4D	2	Tiempo de ejecución con la luz MIL encendida	minutos	(A*256)+B
4E	2	Problema de tiempo desde el borrado de códigos	minutos	(A*256)+B
4F	4	El valor máximo para la relación de equivalencia, el voltaje del sensor de oxígeno, la corriente del sensor de oxígeno, y el colector de admisión de presión absoluta	V mA KPa	A, B, C, D*10
50	4	El valor máximo para la tasa de flujo de aire del	g/s	A*10, B, C y D reservada para uso

PID (HEX)	BYTES DE DATOS DEVUELTO	DESCRIPCIÓN	UNIDADES	FÓRMULA
51	1	Tipo de combustible		Ver tabla de tipo de combustible
52	1	% Etanol en combustible	%	A*100/255
53	2	Presión absoluta de vapor del sistema	KPa	((A*256)+B)/200
54	2	Presión del vapor del sistema	KPa	((A*256)+B)-32767
55	2	Sensor de oxígeno secundario a corto plazo del banco de ajuste 1 y el banco 3	%	(A-128)*100/128 (B-128)*100/128
56	2	Sensor de oxígeno secundario a largo plazo del banco de ajuste 1 y el banco 3	%	(A-128)*100/128 (B-128)*100/128
57	2	Sensor de oxígeno secundario a corto plazo del banco de ajuste 2 y el banco 4	%	(A-128)*100/128 (B-128)*100/128
58	2	Sensor de oxígeno secundario a largo plazo del banco de ajuste 2 y el banco 4	%	(A-128)*100/128 (B-128)*100/128
59	2	Presión del carril de combustible (absoluta)	KPa	((A*256)+B)*10
5A	1	La posición del pedal del acelerador relativa	%	A*100/255
5B	1	Vida restante batería híbrida	%	A*100/255
5C	1	Temperatura del aceite	ºC	A-40

PID (HEX)	BYTES DE DATOS DEVUELTO	DESCRIPCIÓN	UNIDADES	FÓRMULA
5D	2	Sincronización de la inyección de combustible	º	(((A*256)+B)-26.880)/128
5E	2	Tasa de combustible del motor	L/h	((A*256)+B)*0.05
5F	1	Los requisitos sobre emisiones a la que el automóvil está diseñado		Bits codificados

APÉNDICE C: COSTO DEL PROYECTO

En esta sección se muestra el costo total del proyecto, la tabla incluye la descripción y el costo unitario.

Descripción	Cantidad	Costo Unitario	Costo Total
Samsung S10	1	$ 600,00	$ 600,00
Tarjeta SIM	1	$ 3,50	$ 3,50
Lector ELM327 Bluetooth	1	$ 25,00	$ 25,00
Plan de datos	1	$ 5,00	$ 5,00
Horas de trabajo	600	$ 4,00	$ 2400,00
Total			$ 3033,50

Tabla C. 1 Costo total del proyecto.

APÉNDICE D: MANUAL DE USUARIO

a) Instalación de la aplicación

Antes de instalar la aplicación debe asegurarse que el dispositivo móvil tiene permitido instalar aplicaciones de origen desconocido. Luego de verificar los permisos antes mencionados, se debe compartir al usuario el archivo apk ya sea a través de una computadora o compartiendo un link de descarga.

Una vez instalada la aplicación, el usuario debe seguir los siguientes pasos:

1) Conectar el dispositivo de diagnóstico ELM 327 al conector DLC del vehículo.
2) Emparejar el escáner con el dispositivo móvil con sistema operativo Android, la contraseña por defecto es 1234.
3) Activar el GPS para hacer uso de las características de localización.
4) Iniciar la aplicación móvil.

b) Interfaz de la aplicación

Al iniciar la aplicación móvil, al usuario se le mostrará la pantalla inicial donde debe dar los permisos para que la aplicación pueda acceder a la ubicación del dispositivo, como se muestra en la siguiente figura:

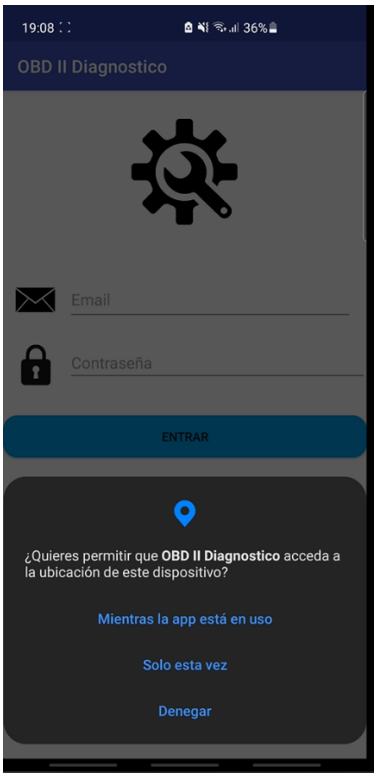


Figura D. 1 Pantalla inicial.

Luego de dar los permisos necesarios, el usuario debe ingresar sus credenciales para autentificarse, previamente el usuario ya tiene que estar registrado en la base de datos de Firebase.

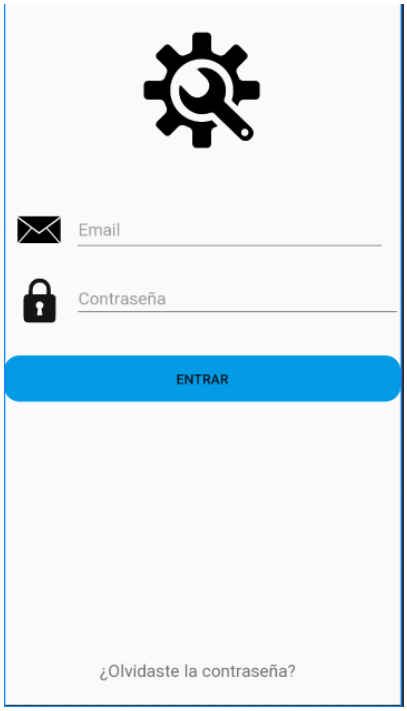


Figura D. 2 Pantalla de inicio de sesión.

Una vez autentificado el usuario se le presentará una pantalla en la cual se debe seleccionar el vehículo que se le asigno para su recorrido, luego ingresar por única vez el kilometraje, como se muestra en la figura:

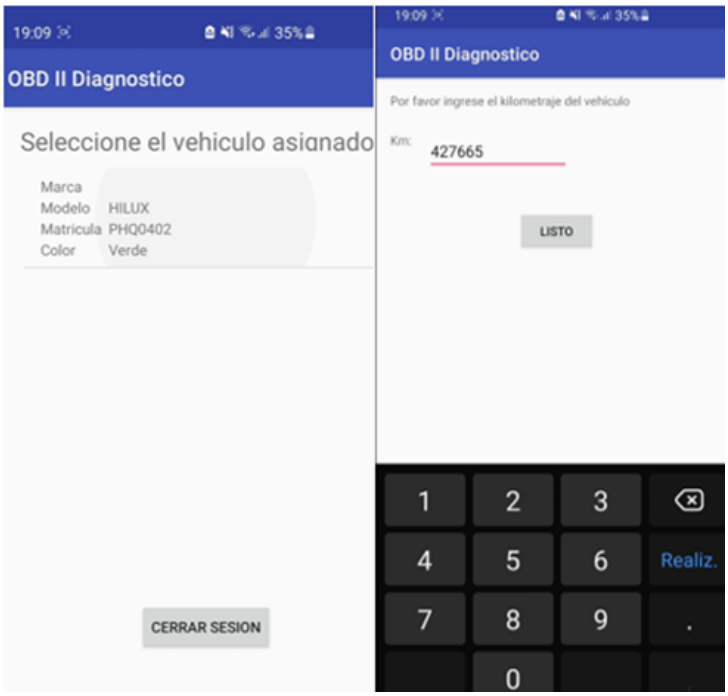


Figura D. 3 Selección del vehículo.

Después de ingresar el kilometraje, la aplicación ejecutará el proceso de configuración y cuando se realice la conexión con el escáner ELM327 se mostrará un mensaje de confirmación en la pantalla.

Figura D. 4 Mensajes de confirmación.

Luego de que se haya establecido la conexión, el usuario observará en la pantalla los datos de kilometraje, velocidad, revoluciones por minuto, consumo de combustible y la distancia recorrida, se mostrará como lo siguiente:

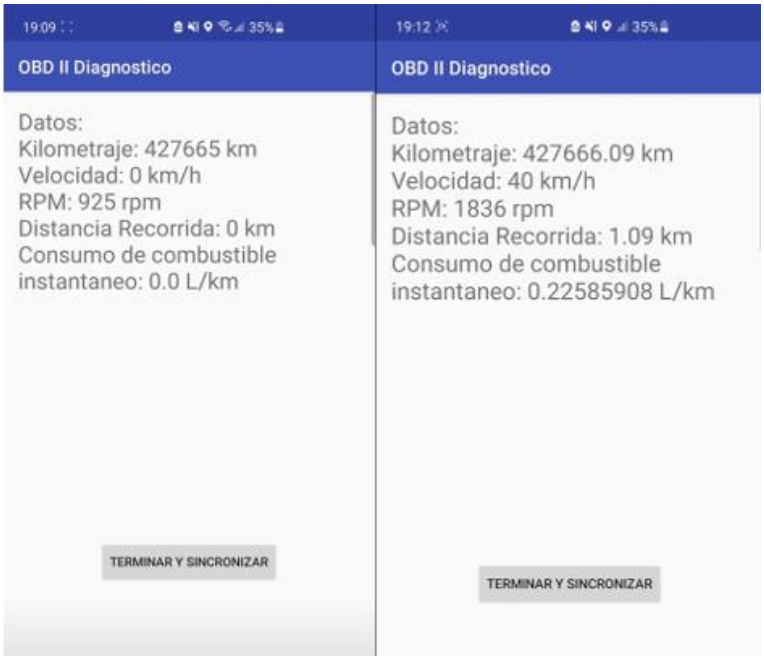


Figura D. 5 Visualización de los datos.

Finalmente, cuando el usuario termine el recorrido, el usuario deberá tocar el botón de terminar y sincronizar para que los datos finales sean enviados a la base de datos en la nube, verá algo similar a la siguiente figura:

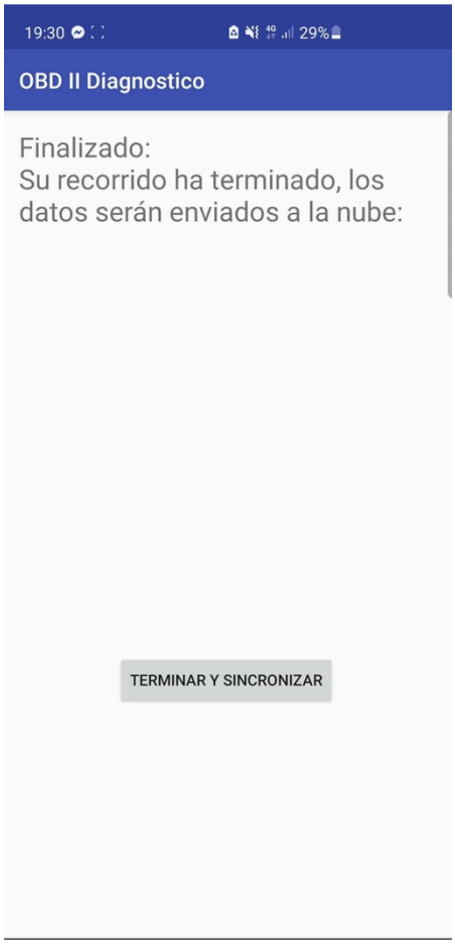


Figura D. 6 Pantalla final.