CLAVE: 13DIT0001E

Titulación Integral

Tesis

"Usos del Puerto OBD2 para diagnóstico del motor de un vehículo desde un dispositivo móvil"

Para obtener el Título de

Ingeniería en Sistemas Computacionales

Integrante(s)

José Antonio Contreras Ramírez

Director

M.S.I María Guadalupe Rivera García

Codirector

M. en C. Rosi Areli Hernández Cruz

Fecha: Enero 2020

Agradecimientos

A Dios

Por darme vida, salud y bendiciones que me permiten salir a delante con mucho esfuerzo y dedicación.

A mis padres

Artemia Ramírez Hernández y Genaro Contreras Martínez por todo el esfuerzo que hicieron para que pudiera culminar con éxito mi carrera y por su apoyo constante e incondicional.

A mis amigos

Por apoyarme en los momentos difíciles en mi vida, por compartir grandes aventuras juntos y sobre todo por estar conmigo en todos mis proyectos y brindarme su apoyo.

A mis asesores

Por su apoyo constante durante este proyecto y también a lo largo de toda mi formación académica al brindarme sus conocimientos en sus respectivos campos.

Resumen

Actualmente, las innovaciones tecnológicas de los automóviles avanzan aceleradamente, mejorando el rendimiento, comodidad de los mismos, y sobre todo la seguridad de sus usuarios. Entre las innovaciones más destacadas se encuentra la inclusión de los sistemas OBD-ll, que hoy por hoy se encargan de monitorear en tiempo real los sensores del motor para verificar el correcto funcionamiento del automóvil.

El OBD II por sus siglas en ingles de "On-Board Diagnostics ll Generation", es una normativa que fue diseñada y creada para la disminución de los niveles de contaminación emitidas por los vehículos. Esta norma fue creada e implementada por los Estados Unidos de América en 1996, con ayuda de esta normativa y a través de un conector normalizado se pueden detectar gran cantidad de fallas e irregularidades en los sensores del motor, intercambiando información con la ECU (Electronic Control Unit). Los precios de dichos dispositivos oscilan entre los 1,000 y los 50,000 pesos que abarcan los escáneres automotrices básicos hasta los sistemas de diagnóstico profesionales, a excepción de otros que por su uso industrial su precio es más elevado y muy variado dependiendo la marca. Sin embargo, a lo largo de los años se han desarrollado diferentes dispositivos que son capaces de leer y enviar los códigos de error a través de la tecnología Bluetooth, ofreciendo así una manera más económica para el usuario(conductor).

En el presente proyecto se definen los usos y aplicaciones del puerto OBD2 y muestra el desarrollo y diseño de una aplicación móvil, dicha aplicación está desarrollada en el lenguaje de programación java (Android Studio) la cual tiene como propósito realizar diagnósticos al motor de un automóvil, ofreciendo una interfaz amigable y fácil de utilizar al usuario final (conductor).

La aplicación móvil es compatible con la mayoría de dispositivos(Android) existentes en el mercado y permite visualizar los resultados del diagnóstico de manera más legible para el usuario(conductor).

ÍNDICE

Capítulo I. Generalidades del Proyecto 1

1.1 Problemas a resolver 2

1.2 Objetivos 3

1.2.1 General: 3

1.2.2 Específicos: 3

1.3 Justificación 4

Capítulo II. Estado del Arte 5

2.1 Antecedentes 6

2.2 Protocolos de comunicación 8

2.3 Dispositivo ELM327 10

2.4 OBDI 11

2.5 OBDII 11

2.6 ELM327 Bluetooth 13

2.7 Descripción de los pines 14

2.8 Características del ELM327 17

2.9 El C.I. ELM327 en el mercado 18

2.10 Conector OBD2 (Diagnostic Link Conector - DLC) 21

Capítulo III. Metodología 23

3.1 Modelo en cascada 24

Capítulo IV. Procedimiento 28

4.1 Procedimiento y descripción de las actividades realizadas 29

4.1.1 Análisis de los protocolos de comunicación. 29

4.1.2 Diseño de la interfaz para la aplicación móvil. 30

4.1.3 Diseño de una base de datos local. 32

4.1.4 Programación de la comunicación Bluetooth. 33
4.1.5 Programación para interpretar información del puerto OBDII. 35
4.1.6 Pruebas de verificación. 36
4.2 Evaluación o Impacto Económico 40
4.2.1 Factibilidad operativa 40
4.2.3 Factibilidad Técnica 41
4.2.1 Factibilidad económica. 42
Capítulo V. Conclusiones y recomendaciones 45
5.1 Conclusiones del Proyecto 46
5.2 Recomendaciones 46
Capítulo VI. Competencias desarrolladas 47
6.1 Competencias desarrolladas y/o aplicadas 48
Capítulo VII. Fuentes de Información 49
Fuentes de consulta 50
Capítulo VIII. Anexos 52
Anexo A. cronograma de las actividades realizadas 53
Anexo B. Tabla de códigos DTC 54
Anexo C. Datasheet del microcontrolador ELM327 57

Índice de Figuras

Figura 1. Emisiones de gases de automóvil. 7
Figura 2. Dispositivo ELM327. 10
Figura 3. Conector ALDL. 11
Figura 4. Calcomanía de información de control de emisiones. 12
Figura 5. Características Físicas del CI ELM327. 13
Figura 6. Diagrama de Bloques del C.I. ELM327. 17
Figura 7. J1962 Conector Vehículo, Tipo A. 21
Figura 8. Pines del DLC. 22
Figura 9. Fases de la metodología en cascada. 24
Figura 10. Posibles localizaciones del puerto OBDII. 30
Figura 11. Listado de Marcas y Fabricantes. 31
Figura 12. Botón que permite realizar diagnósticos al motor del vehículo. 31
Figura 13. Método para Construir Base de Datos. 32
Figura 14. Método que rellena el Layout con información de la base de datos. 32
Figura 15. Método que permite seleccionar el dispositivo Bluetooth. 33
Figura 16. Método que verifica el estado de la conexión para evitar errores. 33
Figura 17. Método para que el Bluetooth se ejecute en segundo plano. 34
Figura 18. Método .Trim() Para Eliminar Espacios. 35
Figura 19. Metodo . Split() Para Separar e Identificar Cadenas. 35
Figura 20. Código para convertir valores Hexadecimales a valores decimales. 38
Figura 21. Switch para envió de PID's 39
Figura 22. Datasheet del microcontrolador ELM327 57

Índice de Tablas

Tabla 1. Comparativa entre distintos software y escáneres para el sistema OBD2. . 20
Tabla 2. Comparativa entre los 5 protocolos de comunicación. 29
Tabla 3. Calcular parámetros de los sensores del motor 37
Tabla 4. Personal requerido para la elaboración del proyecto. 40
Tabla 5. Materiales requeridos. 41
Tabla 6. Presupuesto para llevar a cabo el proyecto. 42
Tabla 7. Presupuesto para el personal requerido. 43
Tabla 8. Inversión total del proyecto. 44
Tabla 9. Cronograma de actividades 53
Tabla 10. Códigos DTC y descripción. 56

Introducción

En la industria automotriz mexicana se representa como el segundo sector económico más importante de este país, además significa el elemento primordial de la modernización y globalización del mismo.

A través del tiempo la industria automotriz ha logrado grandes avances en cuanto a tecnología se refiere, a pesar de ello muchos de los usuarios (conductores) no hacen uso de ellas, ya que estas son muy costosas o están orientadas más a aquellas personas con un conocimiento más amplio en mecánica automotriz.

Gracias a los avances dentro del campo automotriz se han logrado crear dispositivos que se encargan de la lectura e interpretación de los diferentes códigos de error que se pudieran encontrar en el vehículo a través de los distintos protocolos de comunicación existentes en la actualidad, uno de los más destacados y con el cual se trabajará en el presente proyecto es el microcontrolador ELM327, este microcontrolador es capaz de leer una amplia variedad de protocolos de comunicación, los cuales pueden ser captados a través de la tecnología bluetooth para su posterior interpretación.

Se darán a conocer los diversos usos y aplicaciones del puerto OBD-ll así mismo como se conforma y de qué manera se puede obtener la información de los sensores del motor de un automóvil. Posteriormente desarrollará y diseñará una aplicación móvil pensada especialmente en los usuarios (conductores), la cual se conectará a través de tecnología Bluetooth al dispositivo (ELM327) que se encarga de leer los códigos de error que pueden presentarse en los sensores del motor de un vehículo. Todo esto buscando crear una interfaz agradable para el usuario y de fácil manejo.

En este documento de investigación se desglosa todo el procedimiento y metodología usada para lograr los alcances y metas establecidas, así como las indagaciones realizadas con el fin de desarrollar una aplicación lo más completa y eficiente posible.

Capítulo I. Generalidades del Proyecto

1.1 Problemas a resolver

Dentro del sector automotriz se han logrado crear diversos métodos para diagnosticar el motor de un automóvil, uno de ellos es haber integrado sensores de lectura en el motor de los automóviles, esto con el fin de monitorear o diagnosticar las fallas que pudieran presentarse en el motor de una manera más rápida y precisa. Desafortunadamente la mayoría de los métodos que se usan para la lectura de estos sensores son muy complejos y requieren del manejo de una persona con los conocimientos necesarios en el área, otra de la desventaja de estos métodos es que los aparatos que se usan para extraer la información de la computadora central del automóvil tienden a ser demasiado costosos y difíciles de adquirir.

Considerando estudios previos en el proceso de adquisición de datos para determinar el estado del funcionamiento del motor, se observó que este proceso no se realiza de manera óptima y legible para los usuarios (conductores) que no poseen los conocimientos adecuados para interpretar dichos datos, a pesar de que en la actualidad existen muchas aplicaciones que se encargan de interpretar los datos que se obtienen del puerto OBD2, pero estas aplicaciones solo se concentran en algunos protocolos de comunicación o en algunas marcas en específico así mismo han sido desarrolladas u orientadas para personas como mecánicos, ingenieros automotrices o personas que posean un conocimiento básico sobre mecánica automotriz.

La propuesta de este proyecto es dar a conocer los usos y aplicaciones del puerto OBD2 y desarrollar una aplicación móvil capaz de interpretar una amplia variedad de protocolos de dicho puerto(OBD2). A partir de los datos obtenidos, brindar al usuario(conductor) información acerca de los motivos de la alerta y/o falla que presente el motor del vehículo, así como también encontrar las posibles soluciones que se pudieran implementar para solucionar el problema.

1.2 Objetivos

1.2.1 General:

Emplear el puerto OBD2 a través de un microcontrolador para realizar el diagnostico de motores de automóviles con el fin de monitorear su estado y enviar la información obtenida mediante tecnología Bluetooth a una aplicación móvil.

1.2.2 Específicos:

- Identificar los diferentes usos del puerto OBD2 para diagnosticar el motor de un automóvil.
- Analizar los diferentes tipos de protocolos de comunicación de acuerdo con las distintas marcas y modelos de autos.
- Diseñar una aplicación móvil para interpretar los datos del puerto OBD2.

1.3 Justificación

En la actualidad con las diferentes tecnologías existentes en el mercado se han logrado desarrollar distintos tipos de sistemas de monitoreo y diagnóstico para automóviles, la gran desventaja de esto es que dichos sistemas no muestran datos concretos, es decir, la información se muestra como un conjunto de datos los cuales tienen que ser interpretados por personas con experiencia y conocimiento en el campo automotriz, cabe destacar que el equipo que se utiliza para realizar este tipo de diagnósticos suele ser muy costoso y no está al alcance de cualquier persona.

Una gran parte de la población no hace uso del puerto OBD2 ya que no sabe cómo interpretar los datos o simplemente no sabe cómo funciona, con el desarrollo de nuevas aplicaciones móviles se ha logrado decodificar la mayoría de los protocolos de comunicación que se manejan dentro del puerto OBD2, la principal desventaja de esto es que dichas aplicaciones también han sido orientadas hacia personas con experiencia en mecánica automotriz y las pocas que son más legibles son costosas y no cualquiera puede adquirirlas.

Con la aplicación móvil se logrará que los usuarios tengan la posibilidad de realizar diagnósticos al motor de sus vehículos, de una manera más sencilla ya que la aplicación móvil contará con una interfaz muy amigable y de fácil manejo para que incluso aquellos usuarios que no tengan un conocimiento amplio de mecánica automotriz puedan comprender lo que el sistema está interpretando, además, la aplicación estará adaptada a la mayoría de protocolos de comunicación para cubrir una amplia gama de marcas y modelos ya que si bien una gran parte de las marcas o modelos de autos funcionan con protocolos estandarizados existen muchas otras que no funcionan con los mismo protocolos de comunicación o utilizan protocolos en específico para diferentes modelos.

Capítulo II. Estado del Arte

2.1 Antecedentes

El sistema de diagnóstico abordo incorporado como estándar en los automóviles actualmente fabricados, nace como una solución para regular la emisión de gases generados por la combustión en los automóviles. La estandarización fue trabajo entre fabricantes, gobierno y entidades preocupadas por el medio ambiente, la cual llevó varias décadas de investigación para poder obtener un sistema eficiente y de altas prestaciones.

En 1975, en los Estados Unidos de América el congreso identificó la creciente industria automotriz. Debido a esto, se observó la creciente contaminación del aire por los gases emitidos por los automóviles. En el acta de contaminación del congreso, este reconoció que la contaminación del aire perjudica la salud de las personas a nivel nacional. Esta acta sería para promover la investigación y asistencia técnica en relación al control de la contaminación del aire.

Después de ocho años de investigación se concluye que es necesario crear una forma de estandarización de regulación por el gobierno en pro de proteger la salud de las personas. En 1963, el congreso aprobó The Clean Air ACT(CAA). El acta establece la mejora, fortalecimiento y aceleración de programas para prevención y reducción de la contaminación del aire. Además de la emisión de gases; algo que influyó la creación del acta fue la contaminación creada por el aceite el cual contenía altos niveles de sulfuro. La investigación de la reducción de sulfuro en lubricantes y combustibles fue establecida luego del acta.

En la década de los años setenta se establece una nueva acta en la cual se divide en regiones a Estados Unidos de América para monitorear y tener un control de la calidad del aire. El presidente Richard Nixon forma el Enviromental Protection Agency (EPA) con el interés de proteger la salud y medio ambiente del país. A EPA se le asignaron treinta millones de dólares para el desarrollo y refuerzo de los estándares de regulación de emisión de gases emitidos por los motores de combustión interna de los automóviles. EPA desarrollo regulaciones según las cuales todos los automóviles

producidos en 1975 tuvieran 90 por ciento menos de emisiones de hidrocarburos y monóxido de carbono en correlación con los modelos producidos en 1970.

Para asegurarse que la industria automotriz se apegara a las nuevas regulaciones de emisiones, EPA estableció una multa de diez mil dólares por cada automóvil producido que no estuviera dentro del rango de las regulaciones. Desde 1975 hasta 1980, EPA trabajo fuertemente en conjunto con la industria automotriz para la reducción de hidrocarburos y monóxido de carbono.

Figura 1. Emisiones de gases de automóvil.

2.2 Protocolos de comunicación

Es un conjunto de reglas y normas establecidas que permiten una comunicación exitosa entre dos o más dispositivos para el intercambio de información. Si se quiere acceder a la ECU para diagnosticar el automóvil se hace indispensable una herramienta de diagnóstico con el protocolo de comunicación que resida dentro del automóvil. Los tipos de protocolo se originan de dos organizaciones: ISO, SAE. Los protocolos estandarizados por ISO se dividen en tres: ISO 15765-4 (CAN,Controller Area Network), ISO 91412, ISO 14230-4 (KWP2000, Keyword Protocol 2000).

ISO 9141-2

El Protocolo de comunicación ISO-9141-2 es el más antiguo de todos y fue definido por ISO en 1989 en respuesta a la solicitud de CARB. Está basado en la comunicación en serie asíncrona representado el bit 0 con cero voltios y el bit 1 con 12 voltios. La velocidad de transmisión de ISO 9141-2 es de 10400 baudios. Generalmente es utilizado por Chrysler, automóviles europeos y asiáticos.

ISO 14230-4

Protocolo de comunicación en serie asíncrona con velocidades de transmisión de 1 200 a 10 400 baudios. También llamado KWP2000. Este protocolo utiliza la capa física de modelo OSI para redes computacionales, así como la capa de sesión en términos de inicialización, establecimiento y finalización de la comunicación. La capa física de este protocolo es idéntica a la del protocolo ISO9141-2.

ISO 15765-4

Protocolo desarrollado por Bosch para la industria automotriz, aeroespacial, industrial y de equipo médico. Conocido ampliamente como Controller Area Network (CAN). Permite la comunicación entre dispositivos sin una computadora host. La velocidad máxima de transferencia es de hasta 1Mbit/s o 1 000 000 baudios en redes con distancia menor a los 40 metros de distancia. Al disminuir la velocidad de transmisión aumenta la distancia entre dispositivos. CAN sigue el modelo OSI de redes computacionales; con la capa de datos (LLC y MAC) y unos aspectos de la capa física.

CAN fue estandarizado en 1986, y a partir del 2008, todos los automóviles livianos utilizarán este protocolo sustituyendo gradualmente a los demás.

SAE J1939

Protocolo basado en CAN, originalmente utilizado en la industria de camiones pesados. Actualmente este se utiliza en gran variedad de aplicaciones propulsadas por motores diésel, entre ellos vehículos de carretera, todo terreno, propulsión marina, bombas industriales y generación de energía. SAE J1939 ha sido adoptado por muchos fabricantes de motores diésel por la creciente demanda de control de emisión de emisiones

SAE J1850

Protocolo de comunicación estandarizado de arquitectura abierta, de bajo costo, utilizado en vehículos terrestres de carretera y todo terreno. Se encuentra en aplicaciones de motor, transmisión, ABS, e instrumentación de automóviles debido a su bajo costo. Este protocolo tiene dos variantes: modulación por pulso variable (VPW) y modulación por ancho de pulso (PWM).

PWM

Técnica de modulación en la cual se modifica el ciclo de trabajo de una señal periódica para transmitir por un canal de comunicación. PWM se relaciona con OBDII debido a que es la codificación utilizada para transmitir a 41,7 Kbps. Este estándar utiliza dos líneas de comunicación y una de referencia. Utiliza Carrier Sense Multiple Access (CSMA) para asegurar la integridad de los datos. CSMA es utilizado por los nodos de una red para detectar y verificar la ausencia de tráfico en el canal de comunicación, antes de poder iniciar la transmisión. PWM en OBDII tiene una velocidad de transmisión de 41,6 Kbps. PWM es el protocolo estándar utilizado por Ford Motor Company.

VPW

Modulación en donde la señal tiene un periodo variable para representar un bit 0 y 1. VPW utiliza para hacer la transición de bit 1 a 0 y de 0 a 1, 64 y 128 microsegundos respectivamente. VPW se utiliza en OBDII con velocidad de transmisión de 10400 baudios y es el protocolo estándar para General Motors Company.

2.3 Dispositivo ELM327

El dispositivo ELM tiene un módulo de comunicación serial RS232 que permite realizar la comunicación con el OBDII del vehículo, transfiriendo bytes de información que sirve de comunicación entre este dispositivo y la ECU, cada comando de bytes esta en lenguaje ASCII los cuales serían comprobados con el fin de asegurar que solo se envíen dígitos hexadecimales, convirtiéndose a dígitos decimales siendo estos los bytes de datos que serían la información de los sensores del vehículo.

Este dispositivo también incluye un convertidor de señal analógica a digital para mediciones de voltaje y un módulo multiprotocolo que es capaz de detectar automáticamente los protocolos OBD que hay en el mercado automotriz.

Figura 2. Dispositivo ELM327.

2.4 OBDI

Debido a las rigurosas regulaciones de principios de la década de los años ochenta, The General Motors Company fue el primer fabricante que incorporó el sistema de diagnóstico abordo en el cual se monitoreó los sistemas de inyección con capacidades y ajustes simples. El sistema se denominó ALCL (Assembly Line Communications Link) más conocido como ALDL (Assembly Line Diagnostic Link). El sistema de diagnóstico abordo, ALDL, incluye doce pines, de los cuales se utilizan nueve, como se muestra en la figura 2: la forma y distribución de los pines en el conector. Cada pin tiene una función específica usando un protocolo no estandarizado. El primer paso para estandarizar el sistema de diagnóstico abordo tuvo la participación de The California Air Resource Board (CARB) y EPA. Estas dos entidades analizaron el caso del ALDL u OBD y concluyeron en estandarizar el sistema de diagnóstico.

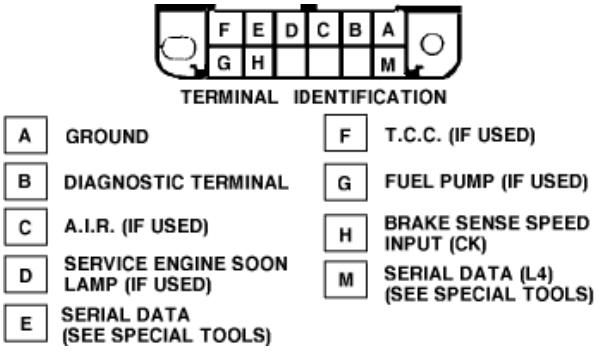


Figura 3. Conector ALDL.

2.5 OBDII

A mediados de la década de los noventas, después de casi seis años de investigación entre SAE (Society of Automotive Engineer), CARB y EPA, la nueva generación sistemas de diagnóstico fue lanzada con el nombre de OBDII (Sistema de diagnóstico abordo versión dos). A partir del 1 enero de 1996, todos los vehículos vendidos tendrían que estar equipados con OBDII.

El estándar OBDII incorporó un conector de diagnóstico, así como la ubicación del conector dentro de la cabina del automóvil. El estándar OBDII definió las partes del

motor que obligadamente debían ser monitoreadas y bajo qué parámetros. Si uno de los sensores detectaba cualquier mal funcionamiento el sistema lo indicaría en el tablero del automóvil por medio de una luz indicadora de funcionamiento anormal o Malfunction Indicator Light (MIL). Los distintos tipos de fallos se estandarizaron de tal forma que ayudaría a reparar la falla en menos tiempo y de forma correcta.

OBDII estandarizó los protocolos de comunicación con la Unidad de Control del Motor o ECU (Engine Control Unit.). La estandarización de protocolo de comunicación facilitó el diagnóstico de los automóviles, ya que no se necesita de herramienta de diagnóstico propia del fabricante. OBDII tiene como fin controlar de una manera más rigurosa la emisión de gases y la vez diagnosticar el funcionamiento anormal de automóviles para cumplir con las reglas establecidas por EPA.

Todos los automóviles equipados con OBDII tienen una calcomanía ubicada debajo del capó. Esta calcomanía indica la información de control de emisiones del automóvil. En la figura 3 se muestra un ejemplo de la calcomanía.

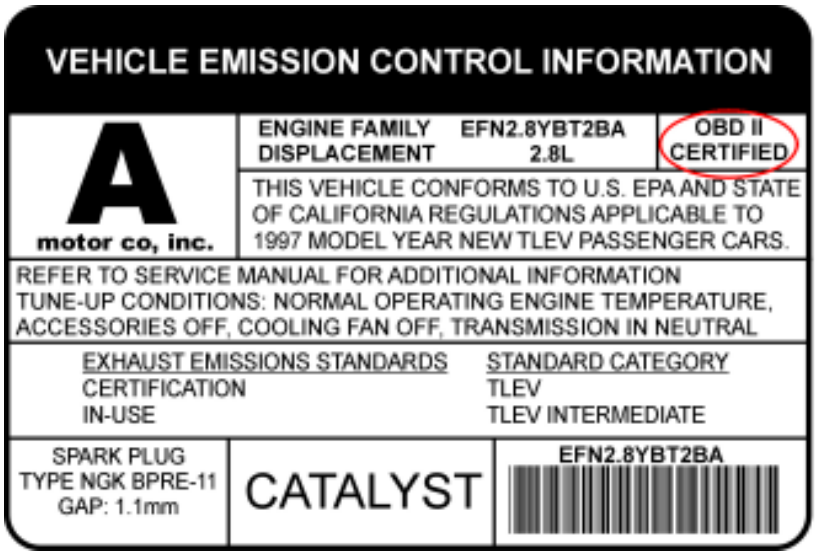


Figura 4. Calcomanía de información de control de emisiones.

2.6 ELM327 Bluetooth

La interfaz ELM327 Bluetooth es una herramienta sin cables desarrollada por ELM Electronics, una empresa canadiense que desarrolla microcontroladores, infrarrojos y además productos enfocados a la seguridad en el hogar, sin embargo, su mayor éxito son los microcontroladores ELM para su uso automotriz como un sistema OBD2. Dicha herramienta procesa los datos recibidos de la computadora del vehículo dependiendo del protocolo que este maneje.

En la hoja de datos del ELM327 podemos encontrar las especificaciones a detalle y los métodos de comunicación para este circuito integrado. Cabe mencionar que el ELM327 Bluetooth cuanta ya con todos los componentes electrónicos para el funcionamiento del microcontrolador, además de que ya cuenta con la interfaz Bluetooth, fue por eso que se seleccionó esta herramienta, además de que su precio es accesible, mientras que una interfaz bluetooth se encuentra entre los 400 a 600 pesos dependiendo del fabricante o proveedor, sin mencionar los componentes que se utilizan para el acopamiento del ELM327 y este realice una comunicación.

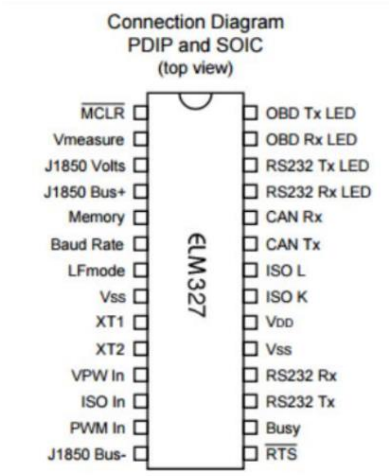


Figura 5. Características Físicas del CI ELM327.

2.7 Descripción de los pines

MLRC (pin 1)

Aplicando una carga lógica baja en esta entrada resetea el C.I. si no es usado, debe ser conectado a una entrada lógica alta.

Vmeasure (pin 2)

Esta entrada analógica es usada para medir la señal de 0 a 5v que es aplicada. Se debe de tener cuidado de sobrepasar los límites de voltaje del C.I. debido a que puede provocar un daño al mismo. Sino es usado debe conectarse a una entrada lógica alta.

J1850 Volts (pin 3)

Esta salida puede ser usada para el control de voltaje del BUS+ de la salida del J1850. El pin tendrá una salida lógica en estado alto cuando se requiera una salida de 8v en el caso de usar el PWM. En caso de no ser requerida para el proyecto, deberá estar conectada en corto circuito.

J1850 Bus+ (pin4)

Esta salida activa es usada para manejar el Bus+ del J1850 a un modo activo.

Memory (pin 5)

Esta entrada controla el estado por defecto de la memoria. Se ésta entrada está activada durante el encendido o en reset, la memoria se habilitará en el modo por defecto. Si está en un modo desactivado, entonces, el modo de memoria se deshabilitará. La memoria puede estar siempre habilitada o deshabilitada con el comando AT M1 Y AT M0.

Baud Rate (pin 6)

Esta entrada controla el rango de baudios de la interface RS232. Si está activo durante el encendido o reset, el rango de baudios se configurará a 38400 (o el rango que se haya configurado por PP 0C) si está desactivo, el rango será de 9600.

LFmode (pin 7)

esta entrada es usada para selección por defecto del modo de la línea de alimentación después del encendido o el modo reset. Esta activo, entonces por defecto los mensajes para el ELM327 serán terminados con un retorno de carro (/r) y un carácter de línea de alimentación, en caso de estar desactivado, las líneas serán terminadas únicamente por un retorno de carro. Esta característica puede ser siempre modificada usando AT L1 o AT L0.

Vss (pines 8 y 19)

El circuito común debe ser conectado a estos pines.

XT1 (pin 9) y XT2 (pin 10)

Un oscilador de cristal de 4.000 MHz es conectado entre otros dos pines. Usando unos capacitores para carga de 27pF, los cuales deben se conectados entre sí y al circuito común (Vss).

VPW In (pin 11)

Esta es la entrada actica para la señal de datos del J1850 VPW. Cuando el bus recibe, este pin debe estar conectado en un modo lógico bajo. Esta entrada tiene un pulsador de forma de onda Schmitt, por eso, no se requiere de una amplificación especial.

ISO In (pin 12)

Esta es la entrada activa baja para la señal de datos del protocolo ISO 9141 e ISO 14230. Está derivado de la línea K y debe de estar en modo lógico activo cuando el bus recibe. No necesita de una amplificación especial.

PWM In (pin 13)

Esta es la entrada activa baja de los datos y la señal del J1950 PWM. Debe estar normalmente en modo activo cuando el bus recibe. Tampoco requiere de una amplificación especial.

J1850 Bus- (pin 14)

Esta salida activa alta es usada para manejar el Bus- del J1850 en su modo uso con PWM. Si no se usa, esta salida puede dejarse en circuito abierto.

RTS (pin 15)

Esta entrada actica baja “Petición para enviar” puede ser usada para la interrupción del procesamiento del OBD, para poder enviar un nuevo comando. Normalmente en alto, la línea es traída en bajo para prestarle atención, debe permanecer hasta que la línea de Busy (pin 16) indique que el ELM327 no está ocupado.

Busy (pin 16)

Esta salida activa alta muestra el estado actual del ELM327. Si está en un nivel bajo, el procesador está listo para recibir comandos ASCII y caracteres, pero si está en nivel alto, los comandos están siendo procesados.

RS232Rx (pin 18)

Esta es ña entrada de datos del RS232. El nivel de la señal es compatible con la mayoría de los C.I. (Cuando está desocupado, el modo es normalmente alto), pero puede ser usado con otras interfaces también, desde que la entrada tiene un pulsador de forma de onda tipo Schmitt.

Vdd (pin 20)

Este es el pin de suplemento positivo, debe ser siempre el punto más positivo del circuito. Las conexiones internas conectadas a este pin son usadas para proveer de energía en el reset del microprocesador, entonces una señal de reset extrema no será requerida.

ISO K (pin 21) e ISO L (pin 22)

Estas son las salidas en activo alto las cuales son usadas para manejar el bus del protocolo ISO 9141 e ISO 14230 para un modo activo (dominante). La moyoria de los nuevos vehículos no requieren la línea L, en este caso se deja en circuito abierto.

CAN TX (pin 23) y CAN Rx (pin 24)

Estas son dos interfaces de señal que deben ser conectadas al transceptor CAN del C.I., en caso de no ser necesario debe de ser conectado a un estado lógico alto de Vdd.

RS232 Rx LED (pin 25), RS232 Tx LED (pin 26), OBD Rx LED (pin 27) y OBD Tx LED (pin 28)

Estas cuatro salidas están normalmente en estado alto, son manejadas en estado bajo cuando el ELM327 está transmitiendo o recibiendo datos. Estas salidas están controlando directamente los LEDs a través de las resistencias, o conectando con el circuito abierto. El pin 28 puede ser usado también para apagar los parámetros programables.

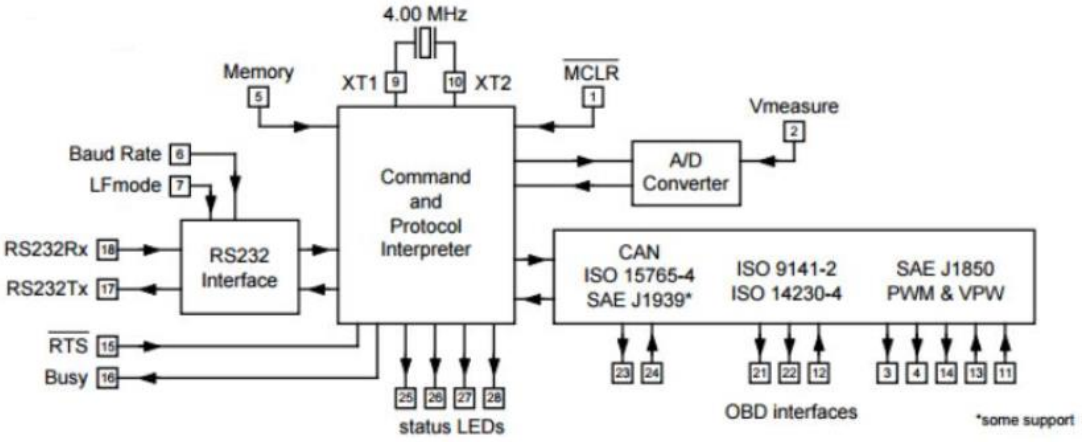


Figura 6. Diagrama de Bloques del C.I. ELM327.

2.8 Características del ELM327

El circuito integrado ha sido comercializado desde 2005, por esto, han sido creadas diferentes versiones en este periodo de tiempo, sin embargo, hay características que se mantienen entre estas versiones como son: lectura de fallas en el motor del vehículo, arrojo de datos de sensores en tiempo real, conexión directa a interface OBD2 de 16 pines, completamente configurable con comando AT, entrada de voltaje para el monitoreo de la batería y soporte en los distintos protocolos de comunicación.

2.9 El C.I. ELM327 en el mercado

El componente se puede adquirir a través de diferentes tiendas en internet o en algunas tiendas de accesorios para vehículos, sin embargo, un tema a considerar es la veracidad de que el dispositivo es el original y no una copia. Se conoció este dato debido a que después de analizar las diferentes aplicaciones para dispositivos móviles no funcionan correctamente debido a que el microcontrolador puede ser una copia.

A continuación, se muestra en la siguiente tabla una recopilación de aplicaciones y herramientas existentes en el mercado para para el uso del puerto OBD2.

NOMBRE DE LA HERRAMIENTA	INTERFAZ O SOPORTE	VENTAJAS	DESVENTAJAS	PRECIO	APLICACIÓN A DESARROLLAR
 Par Lite (OBD2 y coches)	Aplicación Android	Restablecimiento de códigos DTC. Datos de rendimiento de motor.	Contiene candados que limitan la lectura del OBD2	Gratuita	Mayor visibilidad de parámetros
 Torque Pro (OBD2 / coche)	Aplicación Android	Visualización de Toque de Motor Visualización de Caballos de Potencia del Motor.	Costo por su adquisición	$61.91	Bajo costo de adquisición
 Diagnostico OBD2 – ELM327	Aplicación Android	Visualización de revoluciones por minuto. Visualización de Velocidad.	Pocos parámetros	Gratuita	Mayor visibilidad de parámetros
 OBD2 Diagnostico de coches	Aplicación iOS	Buscador y explorador de códigos.	Compatible solo con códigos SAE.	$36.00	Bajo costo de adquisición

 OBD AutoDoctor Lite	Aplicación Symbian	Soporte para pruebas de monitoreo. Presentación de informe de parámetros.	Pocos parámetros.	Gratuita	Mayor visibilidad de parámetros
 OBD AutoDoctor Pro	Aplicación Symbian. Software para PC	Monitoreo de rendimiento de motor. Lectura y descripción de códigos de diagnóstico.	Costo para su adquisición	$75.00	Bajo costo de adquisición
 OBD2 Problemas Búsqueda de Código	Aplicación para Windows Phone	Buscador y explorador de códigos	Pocos parámetros	$1.99	Mayor visibilidad de parámetros
 OBDLink MX Scan Tool | Interfaz OBD	OBD2 Bluetooth	Amplia gama de vehículos. Compatibilidad con software PC, Smartphones y Tablets.	Costo por adquisición. Dificultad para su adquisición.	$3329.00	Bajo costo de adquisición
 ScanXLProfesional	Software PC	Administrador de vehículos	Costo por adquisición. Capacitación para manipularlo.	$1730.95	No se necesita capacitación
 ScanMaster DAB Software	Software PC	Lector de códigos DTC. Protocolo de detección.	Se necesita software en PC.	$935.00	No se necesita una PC
 OBD Link WiFi herramienta de análisis	Herramienta de Diagnostico	Compatibilidad con dispositivos WIFI.	Dificultad para adquisición	$3063.95	Fácil adquisición

 ELM327 USB	Herramienta de Diagnostico	Compatibilidad con una gama de software. Interfaz USB.	Se necesita Sotware en PC. Alámbrico.	$319.00	Inalámbrico
 ELM327 Bluetooth Interfaz Universal OBD2 ELM327	Herramienta de análisis	Interfaz Bluetooth. Inalámbrico.	Complemento con software	$645.95	Herramienta a utilizar para interpretar datos OBD2
 EasyOBDII	Software PC	Detección de código de errores. Códigos de diagnóstico.	Contiene candados que limitan el software	Gratuita	No se necesita una PC
 INNOVA 3120 Lector de Códigos de Diagnostico OBDI y OBDII	Escáner	Recuperación de problemas de diagnóstico. Protocolo OBDI y OBDII.	Costo por adquisición. Capacitación para manipularlo.	$3500.00	No se necesita capacitación para su uso.
 Escáner Automotriz EOBD CAN BUS Launch CREADER VI	Escáner	Pruebas de sensores de oxígeno. Lectura de códigos DTC.	Dificultad para adquisición.	$1299.00	Fácil adquisición.

Tabla 1. Comparativa entre distintos software y escáneres para el sistema OBD2.

2.10 Conector OBD2 (Diagnostic Link Conector - DLC)

El conector de diagnóstico DLC permite la conexión con la herramienta de diagnóstico (escáner).

Como ahora sabe, al comienzo de la etapa automotriz cada fabricante usaba su propio sistema de autodiagnóstico a bordo, cada fabricante estableció su propio protocolo de comunicación y un conector único para el sistema de diagnóstico.

La EPA (Agencia de Protección al Ambiente) estableció una norma que dicta de que todos los vehículos que sean vendidos a partir de 1996 deberán contar con un conector trapezoidal de 16 pines para el sistema de auto diagnostico OBD2. Existen dos tipos de conectores de alcance de diagnóstico (DLC) definidos por la norma SAE J1962, el tipo A y tipo B, la principal diferencia entre estos conectores es en la forma de lengüeta de la alineación.

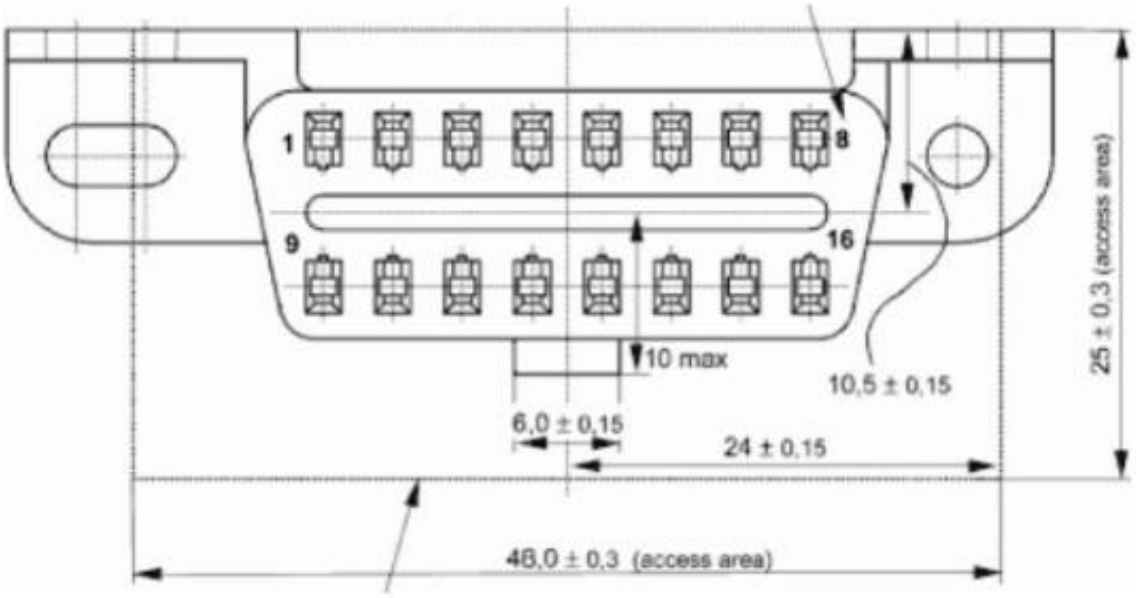


Figura 7. J1962 Conector Vehículo, Tipo A.

Descripción de los pines

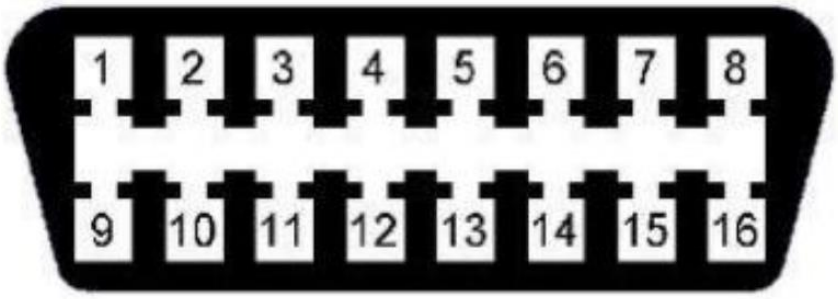


Figura 8. Pines del DLC.

1,3,8,9 – Opcional según el vendedor

2 – Comunicación SAE VPW/PWM, SAE J18850

4 – Tierra (Vehículo)

5 – Tierra (Señal)

6 – CAN, línea alta, SAE J2284

7 – Comunicación ISO 9141-2 (línea K)

10 – Comunicación PWM, SAE J1850

14 – CAN, línea baja, SAE J2284

15 – Comunicación ISO 9141-2 (línea L)

16 – Batería

La comunicación con el escáner que existe es básicamente de tres tipos, que pueden ser utilizadas y son escogidas por la armadora:

SAE VPW – Modulación por ancho de pulso variable

SAE PWM – Modulación por ancho de pulso

ISO 9141-2 – Comunicación serial

Capítulo III. Metodología

3.1 Modelo en cascada

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo de software se concibe como un conjunto de etapas que se ejecutan una tras otra. Se le denomina así por las posiciones que ocupan las diferentes fases que componen el proyecto, colocadas una encima de otra, y siguiendo un flujo de ejecución de arriba hacia abajo, como una cascada.

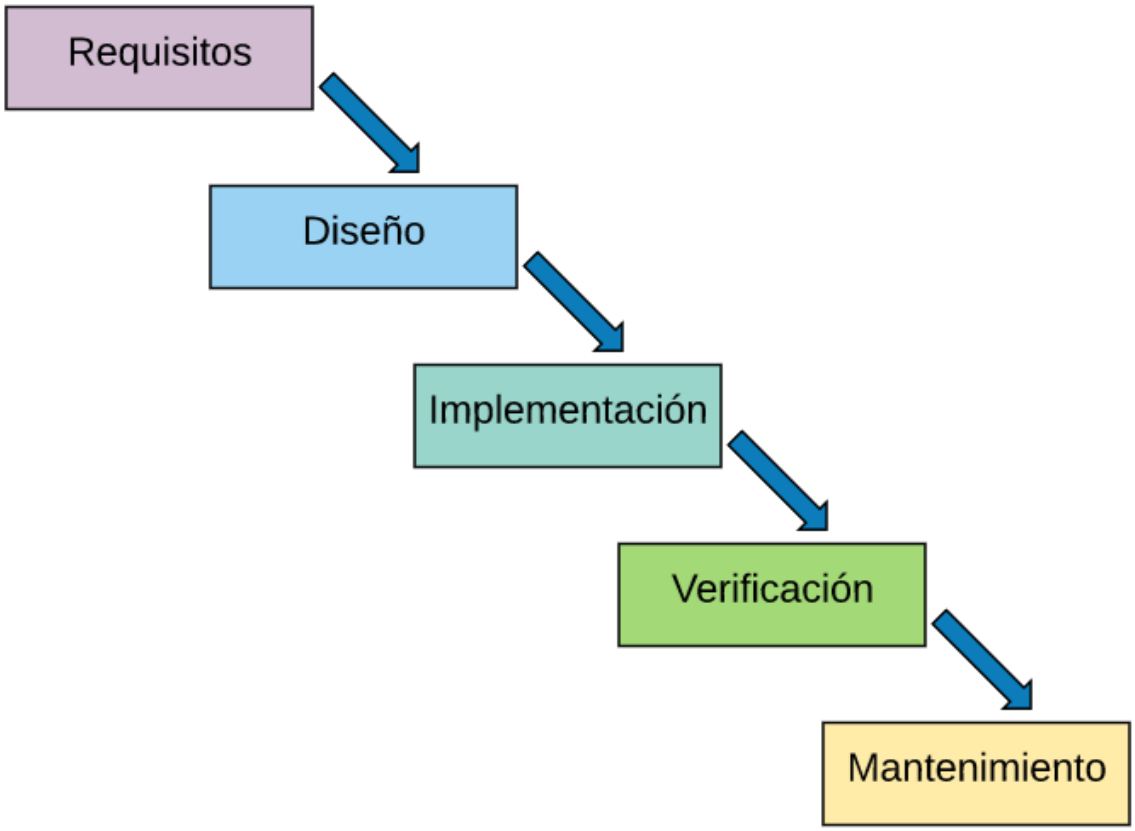


Figura 9. Fases de la metodología en cascada.

El modelo de desarrollo en cascada se originó en la industria y la construcción, donde los cambios a posteriori son caros y difíciles de implementar. Cuando estás creando un producto material, realizar cambios en lo ya construido es mucho más difícil que en un programa informático. En el mundo del software, todavía no se habían implantado otras metodologías de desarrollo por lo que se adaptó el modelo en cascada que se utilizaba en otros sectores.

Fases del modelo

El modelo de desarrollo en cascada sigue una serie de etapas de forma sucesiva, la etapa siguiente empieza cuando termina la etapa anterior.

Las fases que componen el modelo son las siguientes:

Requisitos

En esta fase se hace un análisis de las necesidades del cliente para determinar las características del software a desarrollar, y se especifica todo lo que debe hacer el sistema sin entrar en detalles técnicos. Hay que ser especialmente cuidadoso en esta primera fase, ya que en este modelo no se pueden añadir nuevos requisitos en mitad del proceso de desarrollo.

Por lo tanto, esta es la etapa en la que se lleva a cabo una descripción de los requisitos del software, y se acuerda entre el cliente y la empresa desarrolladora lo que el producto deberá hacer. Disponer de una especificación de los requisitos permite estimar de forma rigurosa las necesidades del software antes de su diseño. Además, permite tener una base a partir de la cual estimar el coste del producto, los riesgos y los plazos.

En el documento en el que se especifican los requisitos, se establece una lista de los requerimientos acordados. Los desarrolladores deben comprender de forma clara el producto que van a desarrollar. Esto se consigue teniendo una lista detallada de los requisitos, y con una comunicación fluida con el cliente hasta que termine el tiempo de desarrollo.

Diseño

En esta etapa se describe la estructura interna del software, y las relaciones entre las entidades que lo componen.

Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge

el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. El segundo define los algoritmos empleados y la organización del código para comenzar la implementación.

Implementación

En esta fase se programan los requisitos especificados haciendo uso de las estructuras de datos diseñadas en la fase anterior. La programación es el proceso que lleva de la formulación de un problema de computación, a un programa que se ejecute produciendo los pasos necesarios para resolver dicho problema.

Al programar, tenemos que realizar actividades como el análisis de las condiciones, la creación de algoritmos, y la implementación de éstos en un lenguaje de programación específico.

Verificación

Como su propio nombre indica, una vez se termina la fase de implementación se verifica que todos los componentes del sistema funcionen correctamente y cumplen con los requisitos.

El objetivo de las pruebas es el de obtener información de la calidad del software, y sirven para: encontrar defectos o bugs, aumentar la calidad del software, refinar el código previamente escrito sin miedo a romperlo o introducir nuevos bugs, etc.

Mantenimiento

Una vez se han desarrollado todas las funcionalidades del software y se ha comprobado que funcionan correctamente, se inicia la fase de instalación y mantenimiento. Se instala la aplicación en el sistema y se comprueba que funcione correctamente en el entorno en que se va a utilizar.

A partir de ahora hay que asegurarse de que el software funcione y hay que destinar recursos a mantenerlo. El mantenimiento del software consiste en la modificación del producto después de haber sido entregado al cliente, ya sea para corregir errores o para mejorar el rendimiento o las características.

Para llevar a cabo correctamente la fase de mantenimiento, se necesita trazar un plan de antemano que nos prepare para todos los escenarios que puedan producirse durante esta fase. Para evitar futuros conflictos con el cliente, en el plan hay que especificar cómo los usuarios solicitarán las modificaciones o la corrección de errores, hacer una estimación del coste de la modificación de funcionalidades o corrección de errores, quién se encargará del mantenimiento, durante cuánto tiempo se dará soporte al software, etc.

Capítulo IV. Procedimiento

4.1 Procedimiento y descripción de las actividades realizadas

Para este proyecto se utilizó la metodología en cascada, ya que es una de las metodologías más eficientes para el diseño y desarrollo de software, en los siguientes puntos podremos encontrar cada una de las fases de esta metodología (Ver Anexo A en la página 53).

4.1.1 Análisis de los protocolos de comunicación.

En este punto se analizarán los diferentes protocolos de comunicación ya que cada fabricante tiene estándares diferentes. A continuación, se puede observar una tabla comparativa de los diferentes protocolos de comunicación disponibles, así como algunos datos que pueden ser útiles al momento de obtener la información y las marcas que hacen uso de estos protocolos.

Protocolo	Fabricante	Comunicación	Velocidad de transmisión	Voltaje
SAE J1850 PWM	Ford, Lincoln y Mercury	Modulación de ancho de pulso (PWM)	41.6 Kbaud/s	0 -5 V en modo diferencial
SAE J1850 VPN	General Motors	Modulación por ancho de pulso variable (VPN)	10.4 – 41.6 Kbaud/s	2.2V – 0L 8V – 1L
ISO 9141-2	Fabricantes europeos, asiáticos, Chrysler, Jeep y Dodge	Comunicación similar al estándar RS-232	10.4 Kbaud/s	0 – 12 V (Se ajustan al voltaje de la batería)
ISO 14230 KWP (Key Word Protocol)	Fabricantes europeos y asiaticos	Comunicación similar al estándar RS-232	1.2 – 10.4 Kbaud/s	0 – 12 V (Se ajustan al voltaje de la batería)
ISO 15765 CAN	Compañía Bosch	Red de Área del controlador	250 – 500 Kbps	2.5 – 5 V (CANH) 2.5 – 0 V (CANL)

Tabla 2. Comparativa entre los 5 protocolos de comunicación.

4.1.2 Diseño de la interfaz para la aplicación móvil.

Se diseñará una aplicación móvil con una interfaz más legible para los usuarios la cual es capaz de interpretar los datos obtenidos del puerto OBD2 y mostrar resultados de una manera más sencilla.

La siguiente imagen se trata de la pantalla principal de la aplicación móvil la cual muestra algunos datos que le pueden ser muy útiles a los usuarios que la usan por primera vez. La imagen muestra las posibles ubicaciones del puerto OBD2 dentro del vehículo ya que la posición puede ser diferente según el fabricante del automóvil.

Figura 10. Posibles localizaciones del puerto OBDII.

Como se pudo observar en la imagen anterior la interfaz es muy sencilla y fácil de comprender. Esta cuenta con un botón en la parte inferior el cual prosigue con el siguiente paso, conectar el dispositivo móvil con el dispositivo lector (ELM327). Para poder realizar la conexión bluetooth con el dispositivo ELM327, la aplicación móvil solo le solicitara el usuario un permiso para poder hacer uso de las configuraciones bluetooth, una vez otorgado dicho permiso, el usuario solo tiene que seleccionar el dispositivo ELM327 de la lista que despliega la aplicación.

En la siguiente pantalla el usuario tendrá que seleccionar la marca de su vehículo, como se puede observar la interfaz es muy legible para cualquier usuario.

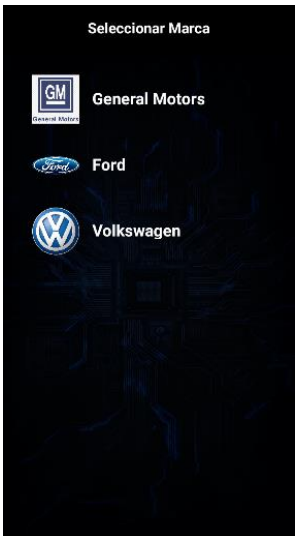


Figura 11. Listado de Marcas y Fabricantes.

Y por último se encuentra la pantalla que permitirá al usuario realizar los diagnósticos de una manera muy sencilla, ya que solo tendrá que presionar un botón para iniciar con el diagnóstico del motor de su automóvil.

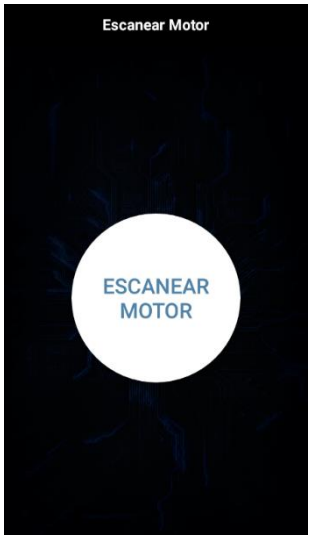


Figura 12. Botón que permite realizar diagnósticos al motor del vehículo.

4.1.3 Diseño de una base de datos local.

En este punto se diseñará e implementará una base de datos local en la aplicación para que el usuario pueda guardar los resultados de los diagnósticos para tener un registro de las fallas que puedan presentarse. En esta base de datos se encuentran registradas todos los códigos de error que puedas detectar los sensores de motor de un vehículo, así como también las posibles soluciones o recomendaciones que el usuario (conductor) pudiera implementar el su automóvil.

```
@Override
public void onCreate(SQLiteDatabase db) {
    sql = "CREATE TABLE modelos (clave INTEGER PRIMARY KEY, modelo VARCHAR(10))";
    sql = "CREATE TABLE marcas (clave INTEGER PRIMARY KEY, nombre VARCHAR(15))";
    sql = "CREATE TABLE codigos_error (clave INTEGER PRIMARY KEY, codigo VARCHAR(30), descripcion VARCHAR(500), solucion VARCHAR(500))";
    sql = "CREATE TABLE historial (clave INTEGER PRIMARY KEY, resumen VARCHAR(500), fecha VARCHAR(10))";
    db.execSQL(sql);
}
```

Figura 13. Método para Construir Base de Datos.

El siguiente segmento de código tiene la función de rellenar el layout con la información de la base de datos para que el usuario pueda proceder con el siguiente paso que es seleccionar la marca de su vehículo y el modelo que le corresponde.

```
@Override
public View getView(final int position, View convertView, ViewGroup parent) {
    Holder holder = new Holder();
    View fila;
    fila = inflater.inflate(R.layout.lista_marcas, root: null);
    holder.tv = (TextView) fila.findViewById(R.id.txt_nombre);
    holder.img = (ImageView) fila.findViewById(R.id.img_logo);
    holder.tv.setText(resultado[position]);
    holder.img.setImageResource(imgId[position]);
    fila.setOnClickListener((v) -> {
            Toast.makeText(contexto, text: "" + resultado[position], Toast.LENGTH_SHORT).show();
            Intent modelo = new Intent(contexto, seleccionar_modelo.class);
            contexto.startActivity(modelo);
    });
    return fila;
}
```

Figura 14. Método que rellena el Layout con información de la base de datos.

4.1.4 Programación de la comunicación Bluetooth.

Una vez diseñada la interfaz de la aplicación móvil se procede a realizar la programación de la comunicación bluetooth entre la aplicación móvil y el dispositivo ELM327 encargado de leer información del puerto OBDII. Esta conexión se realiza mediante un método el cual analiza los dispositivos que estén vinculados con el dispositivo móvil y los enlista para que el usuario solo tenga que seleccionar el dispositivo que corresponde.

```
private AdapterView.OnItemClickListener mDeviceClickListener = (av, v, arg2, arg3) -> {
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        Intent i = new Intent( packageContext: DispositivosBT.this, seleccionar_marca.class);
        i.putExtra(EXTRA_DEVICE_ADDRESS, address);
        startActivity(i);
};
```

Figura 15. Método que permite seleccionar el dispositivo Bluetooth.

Verificación del estado Bluetooth.

En la siguiente imagen se puede apreciar el método que verifica el estado de la conexión para determinar si la conexión se realizó correctamente y así evitar errores durante la trasferencia de la información.

```
private void VerificarEstadoBT(){
    //comprueba si esta activado el bluetooth
    mBtAdapter = BluetoothAdapter.getDefaultAdapter();
    if (mBtAdapter == null){
        Toast.makeText(getBaseContext(), text: "Dispositivo no conectado", Toast.LENGTH_SHORT).show();
    }else{
        if (mBtAdapter.isEnabled()){
            Log.d(TAG, msg: "Bluetooth Activado");
        }else{
            //solicitud para activar el bluetooth
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, requestCode: 1);
        }
    }
}
```

Figura 16. Método que verifica el estado de la conexión para evitar errores.

Método onResume.

Como último punto se encuentra el método que permite mantener el proceso de conexión bluetooth en segundo plano. Esto permite que, aunque el usuario minimice la aplicación esta se seguirá ejecutando en segundo plano para que la conexión con el dispositivo ELM327 no sea vea afectada o interrumpida, ya que en algunas ocasiones el usuario puede cerrar la aplicación por accidente y perder la conexión con el dispositivo, ocasionando que la información se pueda dañar o que la aplicación muestre resultados erróneos.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_dispositivos_bt);

    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    getSupportActionBar().setDisplayOptions(ActionBar.DISPLAY_SHOW_CUSTOM);
    getSupportActionBar().setCustomView(R.layout.txt_dispositivos_bt);
}

@Override
protected void onResume() {
    super.onResume();
    VerificarEstadoBT();

    mPaireDevicesArrayAdapter = new ArrayAdapter<String>( context: this, R.layout.nombre_dispositivos);
    lv_lista = (ListView) findViewById(R.id.lv_lista);
    lv_lista.setAdapter(mPaireDevicesArrayAdapter);
    lv_lista.setOnItemClickListener(mDeviceClickListener);

    mBtAdapter = BluetoothAdapter.getDefaultAdapter();

    Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

    if ((pairedDevices).size() > 0){
        for (BluetoothDevice device : pairedDevices){
            mPaireDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    }
}
```

Figura 17. Método para que el Bluetooth se ejecute en segundo plano.

4.1.5 Programación para interpretar información del puerto OBDII.

En este punto se realizará la programación necesaria para que la aplicación móvil pueda interpretar los datos que se obtienen a través del puerto OBDII y sea capaz de detectar el problema que presenta el motor y algunas posibles soluciones.

Poder interpretar los datos fue un gran problema, ya que, al analizar la respuesta de la herramienta, se observó que devolvía valores con espacios de más y saltos de línea, por los cual se tuvo que identificar los valores a partir de ello y tratar de eliminar algunos espacios. Para ello se utilizaron los métodos .trim() y .split().

El método .trim() se utilizó para poder eliminar los espacios al principio y al final de la respuesta.

```
private void metodoTrim(){
    String Str = new String( original: " < prueba de trim < ");
    String Str2 = new String( original: " < reduccion de espacios < ");
    String Str3 = new String( original: " < al inicio < ");
    String Str4 = new String( original: " < o al final del string < ");

    System.out.println("resultado:");
    System.out.println(Str.trim());
    System.out.println(Str2.trim());
    System.out.println(Str3.trim());
    System.out.println(Str4.trim());
}
```

Figura 18. Método .Trim() Para Eliminar Espacios.

Se utilizó el método .split(), este método permite separar e identificar cadenas, este se utilizó para poder realizar la identificación de los datos de respuesta de los PID's se fueron dados por el dispositivo ELM327 Bluetooth. Aquí se muestra un ejemplo del método .plit().

```
private void metodoSplit(){
    String codigo = " > 41" + "01 0b" + '\n' + " 07" + " 03";
    System.out.println(codigo);
    System.out.println("Se ha encontrado: " + codigo.split( regex: " ").length + " palabras");
    System.out.println(" - segmento 1: " + codigo.split( regex: " ")[0]);
    System.out.println(" - segmento 2: " + codigo.split( regex: " ")[1]);
    System.out.println(" - segmento 3: " + codigo.split( regex: " ")[2]);
    System.out.println(" - segmento 4: " + codigo.split( regex: " ")[3]);
    System.out.println(" - segmento 5: " + codigo.split( regex: " ")[4]);
    System.out.println(" - segmento 6: " + codigo.split( regex: " ")[5]);
    System.out.println(" - segmento 7: " + codigo.split( regex: " ")[6]);
    System.out.println(" - segmento 8: " + codigo.split( regex: " ")[7]);
}
```

Figura 19. Metodo . Split() Para Separar e Identificar Cadenas.

4.1.6 Pruebas de verificación.

Ahora con la comprensión del comportamiento del microcontrolador ELM327 y como se puede realizar una comunicación con el puerto OBD2 y la modificación del Bluetooth chat se iniciaron las pruebas de conexión con la herramienta de diagnóstico, se tuvo que investigar más a fondo el comportamiento de la conexión Bluetooth y se continuaron haciendo modificaciones.

Se realizaron pruebas con distintos PID's a través del dispositivo móvil Moto E5 Plus, esto para saber que PID's pueden ser interpretados por el ELM327 Bluetooth, y el sistema OBD2 y la aplicación diseñada en un vehículo de prueba que tiene un protocolo ISO 14230-4 KWP FAST, los datos enviados fueron interpretados y se recibió una respuesta según las condiciones en que se encontraba el motor y distintos sistemas del vehículo, en la figura se muestran algunos PID's tomados en cuenta para la prueba y aplicación.

MODO (HEX)	PID (HEX)	DESCRIPCION	UNIDADES	FORMULA
01	04	Valor calculado de carga del motor	%	A*100/255
01	05	Temperatura del refrigerante del motor	°C	A-40
01	06	% duración de combustible a corto plazo	%	(A-128)*100/128
01	07	% duración de combustible a largo plazo	%	(A-128)*100*128
01	0B	Presión de múltiple de admisión	KPa (absoluta)	A
01	0C	RPM del motor	Rpm	((A*256)+B)/4
01	0D	Velocidad del vehículo	Km/h	A
01	0E	Regulación de avance	° relativo al cilindro #1	A/2-64

01	0F	Temperatura de aire de admisión	°C	A-40
01	11	Posición de mariposa	%	A*100/255
01	14	Voltaje de sensor de oxígeno	Volts	A*0.005(B-128)*100/128
01	15	Voltaje de sensor de oxígeno	Volts	A*0.005(B-128)*100/128

Tabla 3. Calcular parámetros de los sensores del motor

Estos son algunos ejemplos de los PID's soportados y las respuestas que se obtuvieron:

Presión de múltiple admisión

Fórmula: A

Unidades: Kpa

Enviado: 01 0B

Recibido: 41 0B 1B

A=1B HEX A=27 DEC 27 KPa es la presión de múltiple admisión

Posición de mariposa (esta prueba de realizo acelerando el motor)

Fórmula: A*100/255

Unidades: %

Enviado: 01 11

Recibido: 01 11 21

A = 21 HEX A = 33 DEC 33% de apertura de la mariposa

01 11 2C

A = 2C HEX A = 44 DEC 44% de apertura de la mariposa

01 22 2ª

A = 2A HEX A = 42 DEC 42% de apertura de la mariposa

01 11 26

A = 26 HEX A = 38 DEC 38% de apertura de la mariposa

01 11 24

A = 24 HEX A = 36 DEC 36% de apertura de la mariposa

Con estas pruebas y resultados se trabajó un conversor numérico para leer los datos arrojados por el ELM327 Bluetooth y tener la interpretación de PID's de una manera en la cual se pudiera adaptar al diseño de la aplicación y el usuario pueda interpretarlos, así se empezó a practicar más en leguaje java haciendo pequeños programas y aplicaciones, como una pequeña calculadora para saber el valor decimal de un valor hexadecimal, este es parte del código de dicha calculadora.

```
public void onClick (View vista){
    TextView temp;
    TextView resultado = (TextView) findViewById(R.id.txt_1);
    temp = (TextView) findViewById(R.id.txt_2);
    try {
        int i = Integer.parseInt(temp.getText().toString(), radix: 16);
        resultado.setText("Resultado: " + (Integer.toString(i)));
    }catch (Exception e) {
        resultado.setText("Codigo no encontrado");
    }
}
```

Figura 20. Código para convertir valores Hexadecimales a valores decimales.

Posteriormente se buscó un método para enviar los PID's y recibir una respuesta, en este caso se utiliza un switch para realizar el ciclo y así poder obtener algunos parámetros para poder integrar al diseño de la aplicación.

En el caso 1 se envía el PID 04 en el modo 01, para obtener una respuesta de la carga del motor, el caso 2 para la temperatura del refrigerante, el 3 la presión que se tiene

en el múltiple de admisión, en el caso 4 se envía el PID 0C para poder obtener las RPM del motor, en el 5 se obtiene la velocidad del vehículo y en el caso 6 el porcentaje de apertura de mariposa, además de la presión del voltaje.

```
switch (numPid) {
    case 1:
        Toast.makeText( context: this,  text: "01 05" + '\r', Toast.LENGTH_SHORT).show(); //carga del motor
        envio.setText("01 05");
        numPid++;
        break;
    case 2:
        Toast.makeText( context: this,  text: "01 05" + '\n', Toast.LENGTH_SHORT).show(); //temperatura del refrigerante
        envio.setText("01 05");
        numPid++;
        break;
    case 3:
        Toast.makeText( context: this,  text: "01 08" + '\r', Toast.LENGTH_SHORT).show(); //presion del multiple admision
        envio.setText("01 0B");
        numPid++;
        break;
    case 4:
        Toast.makeText( context: this,  text: "01 0C" + '\r', Toast.LENGTH_SHORT).show(); //rpm del motor
        envio.setText("01 0C");
        numPid++;
        break;
    case 5:
        Toast.makeText( context: this,  text: "01 0D" + '\r', Toast.LENGTH_SHORT).show(); //velocidad del vehiculo
        envio.setText("01 0D");
        numPid++;
        break;
    case 6:
        Toast.makeText( context: this,  text: "01 11" + '\r', Toast.LENGTH_SHORT).show(); //posicion de mariposa
        envio.setText("01 11");
        numPid++;
        break;
    case 7:
        Toast.makeText( context: this,  text: "AT RV" + '\r', Toast.LENGTH_SHORT).show(); //voltaje
        envio.setText("AT RV");
        numPid++;
        break;
}
```

Figura 21. Switch para envió de PID's

4.2 Evaluación o Impacto Económico

4.2.1 Factibilidad operativa

Este proyecto permite a los usuarios (conductores), conocer la posibilidad de conseguir poner en marcha las nuevas tecnologías de comunicaciones inalámbricas y aplicaciones móviles, aprovechando los múltiples beneficios que ofrece.

A continuación, se puede observar una tabla con el personal requerido para llevar a cabo el proyecto de manera correcta, así como también el perfil con el que estos deberán contar.

Personal	Perfil deseado
Desarrollador	Experiencia trabajando con AndroidStudio y conocimiento del leguaje java, así como también el manejo de interfaces en XML.
Diseñador	Experiencia en diseños de interfaces de fácil manejo y agradable para el usuario final y elaboración de diseños para campañas de marketing.
Ingeniero automotriz	Conocimiento en telemetría automotriz y experiencia para extraer datos de la computadora central del automóvil.

Tabla 4. Personal requerido para la elaboración del proyecto.

En cuanto a los usuarios finales (conductores), solo necesitaran conocimientos básicos acerca del manejo de aplicaciones móviles ya que el principal objetivo de esta aplicación móvil es facilitarle a los usuarios la lectura e interpretación de los datos.

4.2.3 Factibilidad Técnica

Para llevar a cabo la elaboración del producto final es necesario percibir de algunos materiales, por lo cual algunos deben ser adquiridos. Cabe mencionar que contar con estos materiales es muy importante para la correcta elaboración del proyecto por lo cual se indagó cuáles serían los materiales adecuados para elaborar el proyecto, tomando en cuenta las características de los materiales y si estos reunían los requisitos necesarios.

La tabla que se puede observar a continuación, muestra los materiales que son necesarios para llevar a cabo la elaboración del proyecto.

Materiales	Cantidad	Requisitos	Descripción
Laptop o Equipo de Escritorio	1	4 GB en Memoria RAM Procesador Core i5	Este equipo deberá cumplir con estos requisitos para soportar el software de desarrollo que se va a utilizar.
Dispositivo Móvil con SO Android (Smartphone o Tablet)	1	Android 4.5 o superior	Ya que el software ha sido desarrollado para ser compatible con la mayoría de dispositivos Android el único requisito es que la versión este actualizada.
Android Studio	1	Versión 3.5.1 o superior	Este es el software en el que se realiza la programación y desarrollo de la aplicación móvil.
Dispositivo Bluetooth ELM327	1	Versión 1.5 o superior	El dispositivo Bluetooth es el que se encargará de la conexión inalámbrica y de leer los datos que se pueden obtener del puerto OBD2.

Tabla 5. Materiales requeridos.

Todos los materiales aquí presentados han sido previamente analizados para satisfacer las necesidades del proyecto y así poder desarrollar una aplicación más eficiente.

4.2.1 Factibilidad económica.

Material requerido.

En esta sección se incluyen los costos referenciales en la implementación de la solución propuesta, dentro de estos se pueden encontrar los costos de los materiales, los honorarios del programador y demás personal requerido para el proyecto.

En la siguiente tabla se pueden apreciar los costos de todo el material y equipo necesario para llevar el proyecto. Cabe mencionar que los costos mencionados en dicha tabla están actualizados hasta noviembre del 2019, por lo tanto, el precio de algunos materiales y/o equipos pueden variar.

Presupuesto de Material			
Equipo/Material	**Cantidad**	**Costo unitario**	**Costo total**
Dispositivo móvil con SO Android (Smarphone, Tablet)	1	$ 3,500	$ 3,500
Laptop o Equipo de Escritorio (4Gb RAM, Core i5)	1	$ 14,999	$ 14,999
Android Studio	1	Gratuito	NA
Dispositivo Bluetooth ELM327	1	$ 450	$ 450
Total			$ 18,696.029

Tabla 6. Presupuesto para llevar a cabo el proyecto.

El presupuesto del material que se utilizará ha sido actualizado a la fecha del 04 de marzo del 2020. Los precios han sido obtenidos de proveedores confiables y autorizados para distribuir este tipo de materiales. También es importante mencionar que el total presentado en la tabla anterior no refleja el costo del producto final ya que esta inversión solo es para la realización de la aplicación móvil y esta solo se realiza una vez en todo el proceso.

Personal requerido.

El recurso humano es de vital importancia para este proyecto, ya que son ellos los encargados de que todas las ideas, objetivos e investigaciones propuestas sean cumplidas. Para ello se ha hecho un análisis de todo el personal con el que será necesario contar para cada una de las tareas del proyecto.

En la siguiente tabla se puede observar los costos relacionados con el personal que se requiere para la realización del proyecto y con los cuales se lograra conseguir desarrollar una aplicación móvil de acuerdo con las especificaciones propuestas en los objetivos de este proyecto de investigación.

Presupuesto de Personal Requerido			
Personal	**Horas**	**Costo unitario**	**Costo total**
Desarrollador	280	$ 92.31	$ 25,846.8
Diseñador	200	$ 81.25	$ 16,250
Ingeniero automotriz	200	$ 98.75	$ 19,750
Total			$ 61,846.8

Tabla 7. Presupuesto para el personal requerido.

El salario de cada miembro del personal requerido, ha sido evaluado conforme al perfil y experiencia de cada uno y tomando en cuenta el salario promedio en México, estos son los costos que se tendrían que cubrir en cuanto a salario si se quiere implementar en una empresa, sin embargo, algunas necesidades pueden ser cubiertas por una sola persona si es que cuenta con los conocimientos necesarios como los es el caso del desarrollador y el diseñador ya que estas actividades van de la mano.

En la siguiente tabla se contempla la inversión total de todo el proyecto tomando en cuenta el costo de mano de obra calificada y de los materiales.

Inversión total	
Concepto	**Inversión**
Sueldo del Personal	$ 61,846.8
Costo de los materiales	$ 18,696.029
Total	$ 81,542.829

Tabla 8. Inversión total del proyecto.

Por último para que este proyecto sea redituable estos costos de producción no reflejan el costo final de la aplicación móvil, por lo cual la propuesta es trabajar con algunas de las marcas más vendidas en México para que el dispositivo ya esté integrado en la computadora del automóvil y de esta manera poder cubrir los costos de producción con el precio final del vehículo, de este modo, la aplicación móvil podrá ser adquirida por los usuarios a un precio muy económico.

Capítulo V. Conclusiones y recomendaciones

5.1 Conclusiones del Proyecto

Al culminar con este proyecto se observó una gran importancia de hacer uso de la tecnología del diagnóstico a bordo y de hacerla más amigable para los usuarios finales(conductores) ya que en algunas encuestas rápidas se obtuvieron respuestas muy positivas de parte de estos usuarios por que la mayoría no tenía idea de que el puerto OBDII se encontraba en sus vehículos. A pesar de que algunos usuarios si tenían conocimiento de este puerto, los limitaba el hecho de no saber cómo utilizarlo para realizar los diagnósticos a sus vehículos, por tal motivo hubo una gran aceptación por parte de estos usuarios para realizar este tipo de aplicaciones que son de fácil entendimiento y manejo.

También se debe que mencionar que como experiencia personal fue muy satisfactorio trabajar en el proyecto ya que tuve la oportunidad de aprender cosas nuevas acerca de la industria automotriz y de poner en practica todo el aprendizaje adquirido dentro del Instituto Tecnológico de Huejutla, las cuales me permiten crecer día con día tanto como persona y como profesionista.

5.2 Recomendaciones

Posteriormente al desarrollo y diseño de la aplicación móvil existen algunas recomendaciones de uso para el usuario, las cuales permitirán el correcto funcionamiento de la aplicación. Algunas de las recomendaciones son:

- Mantener la aplicación actualizada. Este punto es muy importante ya que en ocasiones se encuentran nuevos códigos de error y estos se integran a la aplicación a través de las actualizaciones.
- Borrar memoria cache. Borrar la memoria cache permite al usuario optimizar el rendimiento de la aplicación móvil al deshacerse de los archivos basura que se generan.

Las recomendaciones antes mencionadas permitirán que la aplicación móvil funciones de manera correcta y de una manera más eficiente.

Capítulo VI. Competencias desarrolladas

6.1 Competencias desarrolladas y/o aplicadas

A lo largo de la realización de este proyecto se aplicaron y desarrollaron distintas competencias para la resolución de problemas, así como también para lograr un alcance optimo a través del trabajo en equipo. Algunas de las competencias desarrolladas son:

- Trabajo en equipo.
- Capacidad de resolver problemas.
- Pensamiento crítico.
- Pensamiento analítico.
- Razonamiento lógico.
- Liderazgo.
- Identificación con la compañía.
- Orientación al logro.
- Impacto e influencia.
- Desarrollo de interrelaciones

Las competencias antes mencionadas me permitieron concluir satisfactoriamente este proyecto, cumpliendo con las metas y los objetivos establecidos.

Capítulo VII. Fuentes de Información

Fuentes de consulta

Android Developers. (17 de febrero de 2020). Introducción a Android Studio. 27 de febrero de 2020, de Google Sitio web: https://developer.android.com/studio/

Christian Arlén Morales Landín y Ulises Yosafat Valverde Jiménez, “scanner automotriz interfaz pc”, escuela superior de ingeniería mecánica y eléctrica unidad profesional “Adolfo López mateos”, México, 2010.

Eliana Lorena Montero Revelo, “Diseño e implementación de un sistema web para el monitoreo del estado de un motor”, Escuela Politécnica Nacional, Quito, 2016.

Elm Electronics Inc. (2019). Elm Electronics. 24-oct-2019, de Elm Electronics Inc. Sitio web: https://www.elmelectronics.com/

Elm Electronics. (2012). ELM327. 26-Oct-2019, de Elm Electronics Sitio web: http://elmelectronics.com/DSheets/ELM327DSF.pdf

Elm Electronics. (2015). ELM327 Programmable Parameters. 28-Oct-2019, de Elm Electronics Sitio web: http://elmelectronics.com/ELM327/Prog_Parameters.pdf

Emprento CA. (2019). Librería de códigos OBD2. 14-Nov-2019, de Emprento CA. Sitio web: https://codigosdtc.com/

Emprento CA. (2019). Características del protocolo OBD2. 17-Nov-2019, de Emprento CA. Sitio web: https://codigosdtc.com/obd2/

Israel Cervantes Alonso y Saúl Osborn Espinosa Solís, “Escáner automotriz de pantalla táctil”, Instituto Politécnico Nacional, México, 2010.

Miguel Ángel Álvarez. (23 de octubre de 2019). Qué es Java. 26 de febrero de 2020, de DesarrolloWeb.com Sitio web: https://desarrolloweb.com/articulos/497.php

Miriam Lucía Loachamín Loachamín, “Implementación de un sistema de administración remota para el proceso de obtención de datos del sistema OBD-II de un automóvil”, Escuela Politécnica Nacional, Quito, 2015.

ORACLE. (1995). ¿Qué es Java? Febrero 2020, de ORACLE Sitio web: https://www.java.com/es/download/faq/whatis_java.xml

Pablo Domínguez. (06 de febrero del 2020). En qué consiste el modelo en cascada. 26 de febrero del 2020, de OpenClassrooms Sitio web: https://openclassrooms.com/en/courses/4309151-gestiona-tu-proyecto-de-desarrollo/4538221-en-que-consiste-el-modelo-en-cascada.

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.

Capítulo VIII. Anexos

Anexo A. cronograma de las actividades realizadas

Actividades	Comprendido del 19 de Agosto al 06 de Diciembre														
	Semanas														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Análisis de los protocolos de comunicación															
Diseño de la interfaz para la aplicación móvil.															
Diseño de una base de datos local															
Programación de la comunicación bluetooth															
Programación para interpretar datos del puerto OBDII															
Pruebas de verificación															

Tabla 9. Cronograma de actividades

Anexo B. Tabla de códigos DTC

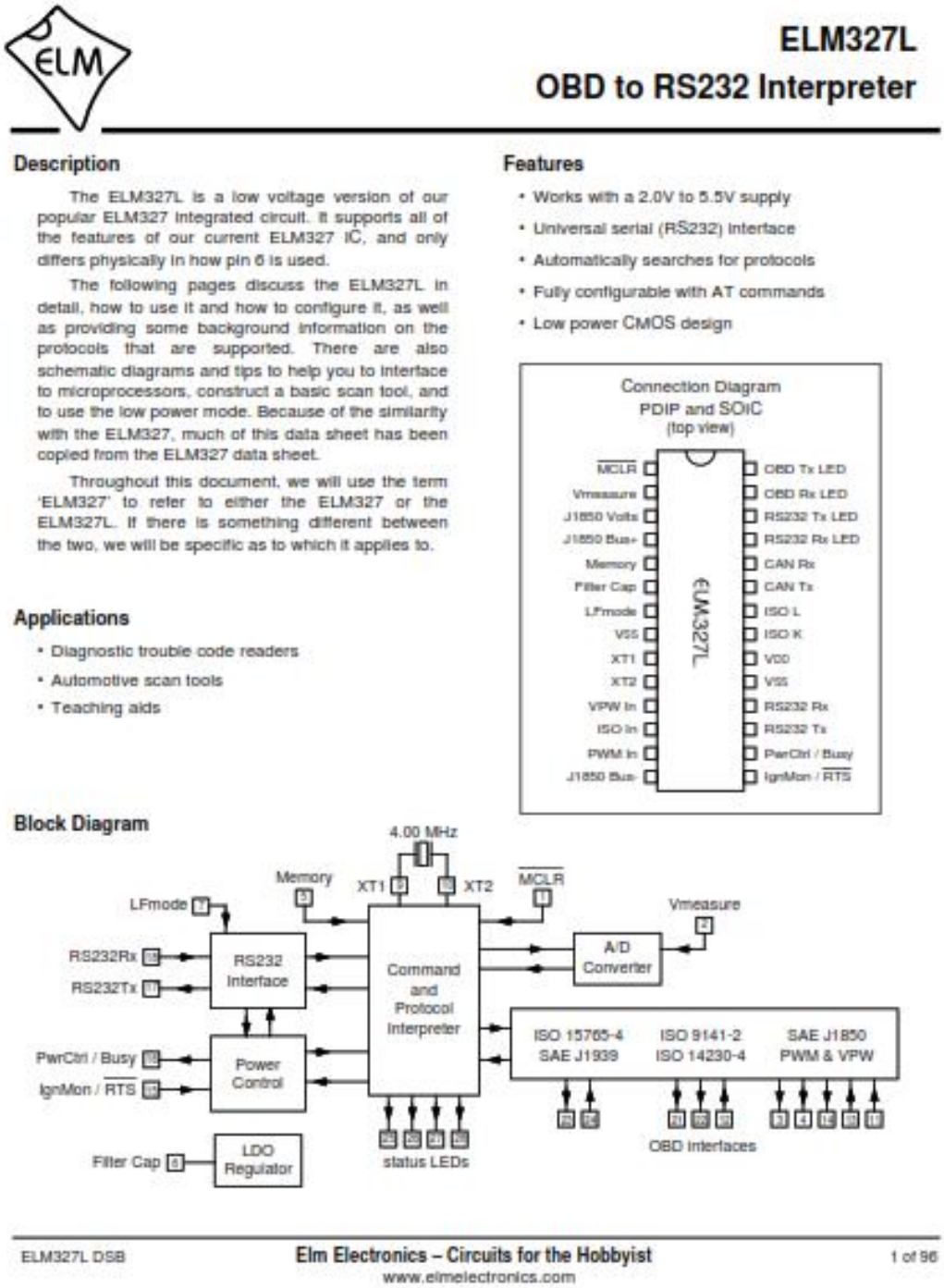
Código DTC	Descripción
P0000	NO SE ENCUENTRA NINGUNA AVERÍA
P0001	Control regulador volumen combustible - circuito abierto
P0002	Control regulador volumen combustible - rango/funcionamiento circuito
P0003	Control regulador volumen combustible - señal baja
P0004	Control regulador volumen combustible - señal alta
P0005	Válvula corte combustible - circuito abierto
P0006	Válvula corte combustible - señal baja
P0007	Válvula corte combustible - señal alta
P0008	Sistema posición motor (bloque 1) - rendimiento
P0009	Sistema posición motor (bloque 2) - rendimiento
P0010	0 Actuador posición árbol levas (bloque 1) - circuito defectuoso
P0011	1 Posición árbol levas (bloque 1) - encendido avanzado, rendimiento
P0012	2 Posición árbol levas (bloque 1) - encendido atrasado
P0013	3 Actuador posición árbol levas (bloque 1) - circuito defectuoso
P0014	4 Actuador posición árbol levas (bloque 1) - encendido avanzado, rendimiento
P0015	5 Actuador posición árbol levas (bloque 1) - encendido atrasado
P0016	6 cigüeñal-árbol levas (bloque 1 sensor A) - correlación
P0017	7 Posición cigüeñal-árbol levas (bloque 1 sensor B) - correlación
P0018	8 Posición cigüeñal-árbol levas (bloque 2 sensor A) - correlación
P0019	9 Posición cigüeñal-árbol levas (bloque 2 sensor B) - correlación
P0020	0 Actuador posición árbol levas (bloque 2) - circuito defectuoso
P0021	1 Posición árbol levas (bloque 2) - encendido avanzado, rendimiento
P0022	2 Posición árbol levas (bloque 2) - encendido atrasado
P0023	3 Actuador posición árbol levas (bloque 2) - circuito defectuoso
P0024	4 Actuador posición árbol levas (bloque 2) - encendido avanzado, rendimiento
P0025	5 Actuador posición árbol levas (bloque 2) - encendido atrasado
P0026	6 Circuito solenoide control válvula admisión (bloque 1) - rango
P0027	7 Circuito solenoide control válvula escape (bloque 1) - rango
P0028	8 Circuito solenoide control válvula admisión (bloque 2) - rango
P0029	9 Circuito solenoide control válvula escape (bloque 2) - rango
P0030	0 Sensor calentado oxígeno (Sensor 1 bloque 1) - circuito defectuoso
P0031	1 Sensor calentado oxígeno (Sensor 1 bloque 1) - señal baja
P0032	2 Sensor calentado oxígeno (Sensor 1 bloque 1) - señal alta
P0033	3 Válvula descarga turbocompresor - circuito defectuoso
P0034	4 descarga turbocompresor - señal baja
P0035	5 Válvula descarga turbocompresor - señal alta
P0036	6 Sensor calentado oxígeno (Sensor 2 bloque 1) - circuito defectuoso
P0037	7 Sensor calentado oxígeno (Sensor 2 bloque 1) - señal baja
P0038	8 Sensor calentado oxígeno (Sensor 2 bloque 1) - señal alta
P0039	9 Válvula derivación turbocompresor - rango
P0040	0 Señales sensor oxígeno cambiadas (bloque 1 sensor 1 y bloque 2 sensor 1)
P0041	1 Señales sensor oxígeno cambiadas (bloque 1 sensor 2 y bloque 2 sensor 2)
P0042	2 Sensor calentado oxígeno (Sensor 3 bloque 1) - circuito defectuoso

P0043	3 Sensor calentado oxígeno (Sensor 3 bloque 1) - señal baja
P0044	4 Sensor calentado oxígeno (Sensor 3 bloque 1) - señal alta
P0045	5 Solenoide sobrealimentación turbocompresor - circuito abierto
P0046	6 Solenoide sobrealimentación turbocompresor - rango, rendimiento
P0047	7 Solenoide sobrealimentación turbocompresor - señal baja
P0048	8 Solenoide sobrealimentación turbocompresor - señal alta
P0049	9 Turbina turbocompresor - sobre velocidad
P0050	0 Sensor calentado oxígeno (Sensor 1 bloque 2) - circuito defectuoso
P0051	1 Sensor calentado oxígeno (Sensor 1 bloque 2) - señal baja
P0052	2 Sensor calentado oxígeno (Sensor 1 bloque 2) - señal alta
P0053	3 Sensor calentado oxígeno (Sensor 1 bloque 1) - resistencia
P0054	4 Sensor calentado oxígeno (Sensor 2 bloque 1) - resistencia
P0055	5 Sensor calentado oxígeno (Sensor 3 bloque 1) - resistencia
P0056	6 Sensor calentado oxígeno (Sensor 2 bloque 2) - circuito defectuoso
P0057	7 Sensor calentado oxígeno (Sensor 2 bloque 2) - señal baja
P0058	8 Sensor calentado oxígeno (Sensor 2 bloque 2) - señal alta
P0059	9 Sensor calentado oxígeno (Sensor 1 bloque 2) - resistencia
P0060	0 Sensor calentado oxígeno (Sensor 2 bloque 2) - resistencia
P0061	1 Sensor calentado oxígeno (Sensor 3 bloque 2) - resistencia
P0062	2 Sensor calentado oxígeno (Sensor 3 bloque 2) - circuito defectuoso
P0063	3 Sensor calentado oxígeno (Sensor 3 bloque 2) - señal baja
P0064	64 Sensor calentado oxígeno (Sensor 3 bloque 2) - señal alta
P0065	65 Inyector asistido por aire - rango, funcionamiento
P0066	66 Inyector asistido por aire - circuito defectuoso, señal baja
P0067	67 Inyector asistido por aire - señal alta
P0068	68 Correlación sensor MAP/sensor MAF/Posición mariposa
P0069	69 Correlación sensor presión absoluta colector/sensor presión barométrica
P0070	70 Sensor temperatura aire ambiente - circuito defectuoso
P0071	71 Sensor temperatura aire ambiente - rango, funcionamiento
P0072	72 Sensor temperatura aire ambiente - señal baja
P0073	73 Sensor temperatura aire ambiente - señal alta
P0074	74 Sensor temperatura aire ambiente - interrupción intermitente
P0075	75 Solenoide control válvula admisión (bloque 1) - circuito defectuoso
P0076	76 Solenoide control válvula admisión (bloque 1) - señal baja
P0077	77 Solenoide control válvula admisión (bloque 1) - señal alta
P0078	78 Solenoide control válvula escape (bloque 1) - circuito defectuoso
P0079	79 Solenoide control válvula escape (bloque 1) - señal baja
P0080	80 Solenoide control válvula escape (bloque 1) - señal alta
P0081	81 Solenoide control válvula admisión (bloque 2) - circuito defectuoso
P0082	82 Solenoide control válvula admisión (bloque 2) - señal baja
P0083	83 Solenoide control válvula admisión (bloque 2) - señal alta
P0084	84 Solenoide control válvula escape (bloque 2) - circuito defectuoso
P0085	85 Solenoide control válvula escape (bloque 2) - señal baja
P0086	86 Solenoide control válvula escape (bloque 2) - señal alta
P0087	87 Rampa combustible/presión sistema demasiado baja
P0088	88 Rampa combustible/presión sistema demasiado alta

P0089	89 Regulador presión combustible 1 - funcionamiento
P0090	90 Solenoide dosificador combustible 1 - circuito abierto
P0091	91 Solenoide dosificador combustible 1 - cortocircuito a masa
P0092	92 Solenoide dosificador combustible 1 - cortocircuito a positivo
P0093	93 Fuga en sistema combustible - fuga grande
P0094	94 Fuga en sistema combustible - fuga pequeña
P0095	95 Sensor temperatura aire admisión 2 - circuito defectuoso
P0096	96 Sensor temperatura aire admisión 2 - rango, funcionamiento
P0097	97 Sensor temperatura aire admisión 2 - señal baja
P0098	98 Sensor temperatura aire admisión 2 - señal alta
P0099	99 Sensor temperatura aire admisión 2 - circuito intermitente
P0100	00 Sensor masa/volumen aire - circuito defectuoso
P0101	01 Sensor masa/volumen aire - rango, funcionamiento
P0102	02 Sensor masa/volumen aire - señal entrada baja
P0103	03 Sensor masa/volumen aire - señal entrada alta
P0104	04 Sensor masa/volumen aire - interrupción intermitente
P0105	05 Sensor presión absoluta colector/presión barométrica - circuito defectuoso
P0106	06 Sensor presión absoluta colector/presión barométrica - rango, funcionamiento
P0107	07 Sensor presión absoluta colector/presión barométrica - señal entrada baja
P0108	08 Sensor presión absoluta colector/presión barométrica - señal entrada alta
P0109	09 Sensor presión absoluta colector/presión barométrica - interrupción intermitente
P0110	10 Sensor temperatura aire admisión - circuito defectuoso
P0111	11 Sensor temperatura aire admisión - rango, funcionamiento
P0112	12 Sensor temperatura aire admisión - señal entrada baja
P0113	13 Sensor temperatura aire admisión - señal entrada alta
P0114	14 Sensor temperatura aire admisión - interrupción intermitente
P0115	15 Sensor temperatura refrigerante motor - circuito defectuoso
P0116	16 Sensor temperatura refrigerante motor - rango, funcionamiento
P0117	17 Sensor temperatura refrigerante motor - señal entrada baja
P0118	18 Sensor temperatura refrigerante motor - señal entrada alta
P0119	19 Sensor temperatura refrigerante motor - interrupción intermitente
P0120	20 Sensor posición pedal acelerador A/mariposa A - circuito defectuoso
P0121	21 Sensor posición pedal acelerador A/mariposa A - rango, funcionamiento
P0122	22 Sensor posición pedal acelerador A/mariposa A - señal entrada baja
P0123	23 Sensor posición pedal acelerador A/mariposa A - señal entrada alta
P0124	24 Sensor posición pedal acelerador A/mariposa A - interrupción intermitente
P0125	25 Temperatura refrigerante insuficiente para control combustible bucle cerr
P0126	26 Temperatura refrigerante insuficiente para funcionamiento estable
P0127	27 Temperatura aire admisión demasiado alta
P0128	28 Termostato refrigerante - circuito defectuoso
P0129	29 Presión barométrica demasiado baja
P0130	30 Sensor oxígeno (Sensor 1 bloque 1) - circuito defectuoso

Tabla 10. Códigos DTC y descripción.

Anexo C. Datasheet del microcontrolador ELM327



ELM327L

OBD to RS232 Interpreter

Description

The ELM327L is a low voltage version of our popular ELM327 integrated circuit. It supports all of the features of our current ELM327 IC, and only differs physically in how pin 6 is used.

The following pages discuss the ELM327L in detail, how to use it and how to configure it, as well as providing some background information on the protocols that are supported. There are also schematic diagrams and tips to help you to interface to microprocessors, construct a basic scan tool, and to use the low power mode. Because of the similarity with the ELM327, much of this data sheet has been copied from the ELM327 data sheet.

Throughout this document, we will use the term 'ELM327' to refer to either the ELM327 or the ELM327L. If there is something different between the two, we will be specific as to which it applies to.

Features

- Works with a 2.0V to 5.5V supply
- Universal serial (RS232) interface
- Automatically searches for protocols
- Fully configurable with AT commands
- Low power CMOS design

Applications

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

Connection Diagram
PDIP and SOIC
(top view)

Block Diagram

Figura 22. Datasheet del microcontrolador ELM327