

**Facultad 1**

**Aplicación informática para el monitoreo de variables en vehículo mediante el dispositivo OBD-2.**

Trabajo de diploma para optar por el título de   
Ingeniero en Ciencias Informáticas

**Autor(es):** Yandry Roger García Pérez

**Tutor(es):** Ing. Julio Alberto Leyva Durán

**Co-tutor:** Ing. Osberto Prieto Pérez.

**Consultante:** <nombre consultante (opcional de existir)>

**Asesor:** <nombre asesor (opcional de existir)>

La Habana, diciembre de 2022

Año 62 de la Revolución

**DECLARACIÓN DE AUTORÍA**

El(Los) autor(es) del trabajo de diploma con título “***Aplicación informática para el monitoreo de variables en vehículo mediante el dispositivo OBD-2***” concede(n) a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara(n) como único(s) autores de su contenido. Para que así conste firma(n) la presente a los <día> días del mes de <mes> del año <año>.

|  |
| --- |
| **Yandry Roger García Pérez** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Autor |
| **Julio Alberto Leyva Durán**. | **<nombre del tutor>** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Tutor | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Tutor |

**DATOS DE CONTACTO**

**Autor:**

Yandry Roger García Pérez

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: [yandryrgp@estudiantes.uci.cu](mailto:yandryrgp@estudiantes.uci.cu)

**Tutor(es):**

Ing. Julio Alberto Leyva Durán.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email:

Ing. Osberto Prieto Pérez.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email:

**Consultante:**

**AGRADECIMIENTOS**

A mis padres y hermana, que sin ellos nada de esto hubiera sido posible.

A mis profesores.

A mi tutor y los demas trabajadores del proyecto por el apoyo dado.

Gracias a todos

**DEDICATORIA**

A mi mama que ha dado todo lo que tiene y no porque yo logre mis sueños, y enseñarme que puedo y siempre me va apoyar, no importa me decisión.

A mi padre que se ha sacrificado y me ha enseñado a luchar para conseguir mis metas, el cual ha luchado por mis sueños y ha sido mi modelo de hombre.

A mi hermana que ha estado hay a mi lado apoyándome siempre.

A mis amigos los que están aún en mi vida y a los que no pero que me han marcado y me han ayudado a ser quien soy. En especial a Álvaro, Adiel, Hugo, Javier, Felo los principales responsables de mi vida en la universidad allá sido tan especial. Daylin quien desde el primer día fue mi amiga y como una hermana que me ayudo en los momentos malos y buenos, quien me hizo luchar y no rendirme en ningún momento.

A Leidy Laura la principal responsable del hombre que soy hoy, en los años que me acompaño me enseño a creer en mí, a no rendirme y seguir para adelante con mis sueños.

Gracias a todos y esto es especialmente para ustedes.

**RESUMEN**

El presente trabajo detalla el desarrollo de un prototipo que utiliza placas arduinos y un dispositivo de tipo abordo OBD2, el cual permite ver los datos al conductor de un vehículo mediante el monitoreo de las variables del mismo, por medio de una aplicación informática escritorio que refleja los datos obtenidos (variables) del automóvil como: Revoluciones por minuto, temperatura, voltaje de la batería, aceleración y fallas que presente. Para el diseño se utiliza la metodología de tipo inductiva que hace referencia a las revisiones de proyectos similares o anteriores a este documento, con lo cual se obtiene una información más concreta y detallada de los resultados que se estudiaron.

PALABRAS CLAVE

*Arduinos, aplicación informática escritorio, automóvil, OBD2, variables*

***ABSTRACT***

The present work details the development of a prototype that uses arduous plates and an OBD2 aboard-type device, which allows the data to be seen to the driver of a vehicle by monitoring the variables thereof, by means of a desktop computer application that reflects the data obtained For the design, the inductive type methodology that refers to the revisions of similar projects or prior to this document is used, with which more concrete and detailed information is obtained from the results that were studied.

*KEYWORDS*

Arduinos, desktop computing application, car, OBD2, variables

**TABLA DE CONTENIDOS**

[ABREVIATURAS xi](#_Toc102658658)

[INTRODUCCIÓN 14](#_Toc102658659)

[CAPÍTULO I: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio 17](#_Toc102658660)

[I.1 Historia del OBD 18](#_Toc102658661)

[I.2 Sistema OBD-1 19](#_Toc102658662)

[I.3 Sistema OBD-2 20](#_Toc102658663)

[I.4 Computadora del automóvil 25](#_Toc102658664)

[I.5 Plataforma Arduino 26](#_Toc102658665)

[I.6 Variables 27](#_Toc102658666)

[I.7 Protocolos de Comunicación 30](#_Toc102658667)

[I.8 Modos de Medición 32](#_Toc102658668)

[I.9 Tecnologías 32](#_Toc102658669)

[I.10 Lenguajes 33](#_Toc102658670)

[I.11 Herramientas 34](#_Toc102658671)

[I.12 Metodologías de desarrollo de software. Las metodologías de desarrollo de software están conformadas por un grupo de procedimientos que ayudan al correcto desarrollo del ciclo de vida de un software de gran envergadura. Son de suma importancia, pues con su uso se puede llevar un control de cadauna de las fases de un proyecto y los resultados que se obtienen. Existen actualmente varias metodologías, cada una de ellas con características peculiares que las hacen distinguirse, son clasificadas en dos grupos: metodologías ágiles y tradicionales. Se deben tener en cuenta varios aspectos en el momento de escoger la que se va a utilizar en el proceso de desarrollo de un proyecto. Programación extrema (XP). Es una metodología ágil que se basa fundamentalmente en promover el trabajo en equipo. El principal objetivo que persigue, es lograr que el producto se termine en el tiempo requerido por los clientes, obteniendo la plena satisfacción del mismo. Es una metodología en la cual existe comunicación entre el cliente y los desarrolladores, pues el cliente forma parte del equipo de trabajo y este tiene el derecho a decidir qué es lo que se debe de implementar, hacer cambios en los requerimientos y exigir por la entrega de una parte del trabajo cada cierto tiempo. Esto demuestra la retroalimentación existente entre ambas partes, la correcta comunicación y forma de trabajo. Para la especificación de requisitos se utilizan las historias de usuarios, las cuales pueden ser totalmente flexibles a cambios o reemplazos. La programación puede llevarse a cabo por parejas de programadores, disminuyendo la cantidad de errores y también se realizan pruebas unitarias. Las cuatro fases que se definen en esta son: planificación, diseño, desarrollo y prueba (Beck, 2000). Proceso unificado de desarrollo de software (RUP). RUP pertenece al grupo de las metodologías tradicionales, es un proceso que puede ser utilizado tanto en proyectos de grandes dimensiones, como en pequeños proyectos. Responde al qué, quién, cómo, y al cuándo. El objetivo principal es el desarrollo de un producto con gran calidad que cumpla con los requisitos establecidos por el cliente. Posee tres características fundamentales: dirigido por casos de uso, centrado en la arquitectura e iterativo incremental. Para el modelado visual utiliza UML y es orientado a objetos. Entre las principales ventajas que presenta esta metodología se destacan la reducción del riesgo que existe de que el producto no sea terminado en el tiempo planificado, el desarrollo es sumamente acelerado y se puede adaptar fácilmente a las necesidades del cliente. El trabajo es particionado en nueve flujos y cuatro fases (Inicio, Elaboración, Construcción y Transición), donde cada una es un ciclo de desarrollo. En esta metodología se identifican 6 prácticas que permiten al equipo de desarrollo realizar un trabajo de forma eficaz (Jacobson et al., 2006): **⮚**Gestión de requisitos. **⮚**Desarrollo de software iterativo. **⮚**Desarrollo basado en componentes. **⮚**Modelado visual. **⮚**Verificación continúa de la calidad. **⮚**Gestión de los cambios. 36](#_Toc102658672)

[Conclusiones del capítulo 37](#_Toc102658673)

[CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO 38](#_Toc102658674)

[II.1 Nombre del Epígrafe II.1 38](#_Toc102658675)

[II.2 Nombre del Epígrafe II.2 38](#_Toc102658676)

[II.3 Nombre del Epígrafe II.3 38](#_Toc102658677)

[II.4 Nombre del Epígrafe II.4 38](#_Toc102658678)

[Conclusiones del capítulo 38](#_Toc102658679)

[CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA 40](#_Toc102658680)

[III.1 Nombre del Epígrafe III.1 40](#_Toc102658681)

[III.2 Nombre del Epígrafe III.2 40](#_Toc102658682)

[III.3 Nombre del Epígrafe III.3 40](#_Toc102658683)

[Conclusiones del capítulo 40](#_Toc102658684)

[CONCLUSIONES FINALES 42](#_Toc102658685)

[RECOMENDACIONES 43](#_Toc102658686)

[REFERENCIAS BIBLIOGRÁFICAS 44](#_Toc102658687)

[ANEXOS 46](#_Toc102658688)

**ÍNDICE DE TABLAS**

[Tabla 1 Funciones de pines de la interfaz DLC [Tomado de: (Socasi, 2016)]. 21](#_Toc102653046)

[Tabla 2 Comparativa entre distintos software y escáneres para el sistema OBD2.[Tomada de :(Contrera Ramírez, 2020) 22](#_Toc102653047)

[Tabla 3 Comparativa entre los 5 protocolos de comunicación [Tomada de: (García Montero, 2013)] 30](#_Toc102653048)

**ÍNDICE DE FIGURAS**

[Figura 1 Interfaz OBD-1 19](#_Toc102653081)

[Figura 2 Componentes de sistema OBD-2 20](#_Toc102653082)

[Figura 3 Interfaz DLC y conector 20](#_Toc102653083)

[Figura 4 Numeración de pines de la interfaz DLC (Socasi, 2016). 21](#_Toc102653084)

[Figura 5 Computadora del automóvil 24](#_Toc102653085)

[Figura 6 Etapas de funcionamiento de ECU(Dimaté Cáceres & González Castillo, 2013) 25](#_Toc102653086)

[Figura 7 Placas Arduinos 26](#_Toc102653087)

[Figura 8 RPM del motor 27](#_Toc102653088)

[Figura 9 Temperatura refrigerante del motor [Tomada de : (Tenorio Córdova & Coronel Magallanes, 2020)] 27](#_Toc102653089)

[Figura 10 Funcionamiento del sensor VSS [Tomada de : (Tenorio Córdova & Coronel Magallanes, 2020)] 28](#_Toc102653090)

[Figura 11 Luz Mil 29](#_Toc102653091)

[Figura 12 Pines de comunicacion [Tomada de:(Tenorio Córdova & Coronel Magallanes, 2020)] 29](#_Toc102653092)

[Figura 13 Modos de medición [Tomada de:(Tenorio Córdova & Coronel Magallanes, 2020)] 31](#_Toc102653093)

# ABREVIATURAS

|  |  |
| --- | --- |
| OBD | Diagnóstico a bordo |
| RPM | Revoluciones por minuto |
| MIL | Indicador de Averías |
| DTC | Códigos de error almacenados |
| DLC | Conector de Diagnóstico |
| ECU | Unidad de control de motor |
| GPS | Sistema de posicionamiento global |
| PIDS | Parámetros de Identificación |
| VSS | Velocidad del automóvil |
| ING. | Ingeniero |

**OPINIÓN DEL(OS) TUTOR(ES)**

<Contenido de la opinión de los tutores>

**AVAL DEL CLIENTE**

<Contenido del aval del cliente sobre la solución desarrollada>

# INTRODUCCIÓN

En la década de los 80 los vehículos contaban con sistemas de platinos y carburadores, un sistema muy antiguo y bastante contaminante. Por lo cual no había un monitoreo constante y completo del vehículo, no se podía saber con claridad cómo se hallaban o funcionaban cada una de las piezas internas del automotor siendo este un factor importante.

En 1969, Volkswagen diseña el primer sistema de monitoreo on-board. Pronto, distintos armadores como Datsun, General Motors, entre otros desarrollaron sus propios sistemas de monitoreo on-board. Fue hasta 1988 cuando la Sociedad de Ingenieros Automotrices (SAE) recomendó crear un conector estándar para todas las marcas para comunicar a los escáneres automotrices con los microcontroladores del auto y así obtener la información del sistema on-board. Así es como nace el estándar OBD (On-board Diagnostics). Sin embargo, aunque el conector fue estandarizado, los protocolos de comunicación y los códigos generados siguieron siendo exclusivos de cada marca.(Ramos Coria, 2014)

Fue hasta 1996 cuando apareció el sucesor del OBD, el OBD-2, que además de mantener un conector estándar, estandarizó los protocolos de comunicación y los códigos, obligando a todos los diseñadores de los sistemas de diagnóstico on-board a utilizar los mismos códigos alfanuméricos para diagnosticar los mismos problemas. Existe sólo una letra dentro del código de OBD-2 que puede ser utilizada libremente por las diferentes marcas para diagnosticar problemas específicos de cada marca o modelo. Problemas de inyección, del sistema de dirección, del sistema eléctrico, etc. son diagnosticados de la misma forma para todas las marcas y generan un código idéntico que puede ser leído por un escáner automotriz de cualquier marca.(Ramos Coria, 2014)

EI OBD-2 por sus siglas en ingles de "On-Board Diagnostics Il 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).(Contrera Ramírez, 2020, p. ii)

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-2, que hoy por hoy se encargan de monitorear en tiempo  
real los sensores del motor para verificar el correcto funcionamiento del automóvil.

**Situación Problemática:**

A partir de la situación problemática descrita anteriormente se plantea como **problema a resolver** ¿cómo establecer un mecanismo que permita…..?

La presente investigación tiene como objetivo general dar conocer los diversos usos y aplicaciones del puerto OBD-2 así mismo como se conforma y de qué manera se puede obtener la información de los sensores de un automóvil. Posteriormente desarrollará y diseñará una aplicación de escritorio pensada especialmente en los usuarios (conductores), la cual se conectará a través de tecnología de una placa Arduino y un sistema ODB-2 que se encarga de leer los códigos de error que pueden presentarse en los sensores de un vehículo. Todo esto buscando crear una interfaz agradable para el usuario y de fácil manejo.

Los métodos teóricos utilizados son:  
Análisis-Sintético: Este método facilita la comprensión del problema planteado, a través del análisis se hace una descomposición del problema en partes, permitiendo una mayor comprensión. Posteriormente las partes son unidas a través de la síntesis logrando describir relaciones entre ellas y características.  
Histórico-Lógico: Permite realizar un estudio exhaustivo de los antecedentes históricos de los sistemas de gestión de variables mediante sistema ODB-2 teniendo en cuenta su evolución hasta la actualidad. Esto proporciona que se realice una investigación acerca del funcionamiento de estos sistemas y su desarrollo, basándose en datos obtenidos a partir del estudio histórico.

Los métodos empíricos utilizados son:  
Entrevista: Es un método que está dirigido a la recopilación de información sobre las variables que se desean controlar en el sistema que se desea desarrollar para ir recopilando los requisitos que se desea que cuente el sistema.

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.

El documento estará estructurado en tres capítulos, conformados con la información que se presenta a continuación.  
Capítulo 1: Fundamentación Teórica.Este capítulo contiene una detallada descripción de los sistemas de información, se analizan sus características y funcionalidades, específicamente los Sistemas de Información para sistemas OBD-2. Se realiza un análisis a sistemas homólogos para identificar características comunes, que puedan aplicarse a la solución. Se muestra un estudio de las tecnologías a utilizar en el desarrollo del sistema.  
Capítulo 2. Propuesta de solución.Se realiza un estudio sobre la información que se maneja actualmente en los sistemas usando ODB-2. Se especifican los requisitos funcionales y no funcionales, así como los casos de uso necesarios. Se especifican las clases que posteriormente son utilizadas en la fase de implementación y se propone el despliegue de la aplicación.  
Capítulo 3. Pruebas.En este capítulo se realizan pruebas a la solución final, para encontrar no conformidades que puedan estar presentes en las funcionalidades desarrolladas.

# CAPÍTULO I: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio

En base a este proyecto de creación de una Aplicación informática para el monitoreo de variables en vehículo mediante el dispositivo OBD-2, se realizó una búsqueda sobre temas y estudios antes desarrollados semejantes al proyecto, lo cual permitió obtener un análisis e instrucciones para crear una idea clara y obtener conceptos más específicos.

Como se refleja en las siguientes investigaciones acerca del tema:

Según el tema de tesis: “*ANÁLISIS Y DISEÑO DE UN SISTEMA PARA EL PROCESO DE ESCANEO DE LOS VEHICULOS HYUNDAI TUCSON (2005-2013), FORD EXPLORER( 2003-2006), CHEVROLET AVEO ACTIVO (2002-2013), CON SISTEMA OBD2 II, FUNDAMENTADO  
EN EL PROYECTO VISUALOBDJAR*”. Un conector estandarizado, es el Data Link Conector, de 16 pines, sirve para cualquier marca de vehículo con sistema OBD2, su denominación de estandarización es S.E.A. J2962, la forma de su configuración de comunicación no va a cambiar  
jamás, lo que significa que sin tomar en cuenta la marca del vehículo, si este posee el sistema OBD2, el técnico se conectará a él para efectuar un diagnóstico con un equipo de diagnóstico cualquiera.(Garcia Alcivar, 2015)

Este proyecto de titulación rinde a nuestro proyecto en el escaneo de los vehículos que constan con diferentes lectores de código dependiendo del año de su fabricación. Desde 1996, los automóviles ya vienen con sistema de lector de código OBD-2.

Según el trabajo de titulación: “*DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN TELEMÁTICA BASADA EN OBD2 (ON-BOARD DIAGNOSTIC) QUE PERMITA OBTENER Y PROCESAR LA INFORMACIÓN DE LOS SENSORES DEL MOTOR DE UN AUTOMÓVIL*” La contaminación del medio ambiente por la emisión de gases de los automóviles ha sido uno de los principales problemas de las grandes ciudades; debido a eso se desarrollaron sistemas de diagnóstico automotriz, que en principio no resultaron muy eficientes, ya que le restaban potencia al motor del automóvil. (Simbaña Coyago, 2015)

Este proyecto de titulación rinde a nuestro proyecto en la importancia de la contaminación de los automóviles, brindado gran importancia el dispositivo OBD-2, para controlar la cantidad de contaminación al medio ambiente y evitando que se destruyan componentes del motor.

Según el proyecto de titulación: “*ELABORACIÓN DE UN MANUAL TÉCNICO PARA EL ANÁLISIS DEL DIAGNÓSTICO ELECTRÓNICO DE VEHÍCULOS CON SISTEMA OBD2 PARA EL TALLER AMBAMAZDA S.A. DURANTE EL AÑO 2012*” Los protocolos de comunicación soportados por el sistema OBD2 incluyen el SAE J1850 PWM (Modulación por ancho de pulso a 41.6 Kbps), el SAE J1850 VPW (Ancho de pulso variable a 10.4 - 41.6 Kbps), el ISO 9141-2 (Comunicación serial asincrónica a 10.4 Kbaud), el ISO 14230 KWP (Comunicación serial sincrónica hasta 10.4 Kbaud), y el ISO 15765 CAN (250 - 500 Kbps) (García Montero, 2013).

Este proyecto rinde a nuestro trabajo permitiéndonos conocer acerca de los protocolos de comunicación que soporta OBD-2, permitiendo el acceso a los datos en tiempo real. Dependiendo del protocolo de comunicación utilizado va a ser la cantidad de PIDS (códigos de operación de cada estándar).

## I.1 Historia del OBD

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.(Contrera Ramírez, 2020).

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 (Contrera Ramírez, 2020).  
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 (Contrera Ramírez, 2020).

Para reducir la contaminación del aire, la "California Air Resources Board" (CARB) determinó en 1988 que todos los automóviles a gasolina contaran con OBD (On Board Diagnostics), que controlará los límites máximos de emisiones y además un autocontrol. Para que el conductor detecte un mal funcionamiento del OBD se impuso la obligación de tener una lámpara que indique fallos (MIL – Malfunction Indicator Lamp). Medidas más estrictas en los límites de emisiones en 1996 llevó a la creación del OBD2. En Europa se introdujo el OBD ajustándose al OBD2 americano. Desde 1996 el OBD2 es un requisito legal para automóviles nuevos en Estados Unidos. Con base en esta regla americana se impuso en los noventa la inclusión de sistemas de diagnóstico también para los automóviles destinados al mercado europeo (Tenorio Córdova & Coronel Magallanes, 2020)

La interfaz estándar del OBD-2 no solamente es utilizada por el fabricante para sus funciones avanzadas de diagnóstico sino también por aquellos que van más allá de lo que la ley exige. La siguiente etapa planeada es el OBD3, en el que los propios automóviles se comunican con las autoridades si se produce un empeoramiento de las emisiones de gases nocivos mientras está en marcha. Si esto sucede, se pedirá a través de una tarjeta indicativa, que se corrijan los defectos (ISRAEL CERVANTES ALONSO & SAÚL OSBOR ́N ESPINOSA SOLÍS, 2010)

## I.2 Sistema OBD-1

El OBD (diagnóstico a bordo, del término On board Diagnostics) no tuvo lugar hasta que los autos estuvieron equipados con controles computarizados. La compañía General Motors implemento una versión primeriza de OBD en algunos de sus vehículos de 1980. (ISRAEL CERVANTES ALONSO & SAÚL OSBOR ́N ESPINOSA SOLÍS, 2010)  
El objetivo del sistema OBD original era promover el aire limpio por medio de asegurar que los componentes del control de emisiones siguieran funcionando. Muchos estados de la unión americana empezaron a incluir una revisión de emisiones en su programa de revisión vehicular de seguridad anual o bienal. Pero, en la práctica mucho puede suceder al vehículo en un año entre revisiones, y las pruebas simplificadas de entonces solamente “olían” el escape mientras el vehículo estaba estacionado, no cuando estaba en movimiento. (ISRAEL CERVANTES ALONSO & SAÚL OSBOR ́N ESPINOSA SOLÍS, 2010).

La idea del OBD era tener un vehículo que realizara su propio monitoreo del control de emisiones, todo el tiempo, y más que eso asignar números de código que pudiesen identificar el problema, y finalmente guardar esos códigos de problema en la computadora. Una luz indicadora de problema en el tablero del vehículo podría indicar al conductor que hay un problema en el sistema de emisiones, y cuando el auto estuviese en el taller, el técnico podría extraer los códigos y saber que partes del sistema debería revisar y reparar (ISRAEL CERVANTES ALONSO & SAÚL OSBOR ́N ESPINOSA SOLÍS, 2010).

Hubo algunas dificultades naturales con el sistema original OBD, hoy conocido como OBD 1 asociadas a la curva de aprendizaje y desarrollo de toda nueva tecnología. Cada fabricante tenía su propia versión de códigos, haciendo más difícil para los técnicos leer y diagnosticar. Si se le agrega que en ocasiones la luz se apagaba después de un tiempo, al conductor se le olvidaba pensando que tal vez el vehículo se había arreglado solo.(Tenorio Córdova & Coronel Magallanes, 2020)



Figura 1 Interfaz OBD-1 [Tomada de: (Tenorio Córdova & Coronel Magallanes, 2020)]

## I.3 Sistema OBD-2

El sistema OBD-2 se comenzó a utilizar de forma obligatoria por los nuevos automóviles en los Estados Unidos desde 1996, teniendo como objetivo monitorear los componentes que afecten el sistema de control de emisiones de gases contaminantes y medir parámetros en tiempo real como:  
temperaturas, presiones, velocidades entre otros (Anastasio & Enrique, 2013).

Cuando el sistema OBD-2 detecta algún problema a través de los transductores, se enciende una luz de advertencia en el tablero alertando al conductor de la falla existente. Además se guarda la información sobre las fallas detectadas en la memoria de la computadora del automóvil, facilitando al técnico automotriz encontrar los problemas para corregirlos posteriormente (Anastasio & Enrique, 2013).

Los componentes del sistema OBD-2 son: la ECU (*Engine Control Unit*) conocida como la computadora del automóvil, los transductores encargados de enviar los datos hacia la ECU, la luz indicadora de fallas (MIL, *Malfunction Indicator Light*) ubicado en el tablero, y el conector de diagnóstico (DLC, *Data Link Connector*) que sirve de interfaz entre la ECU y los dispositivos de diagnóstico automotriz.

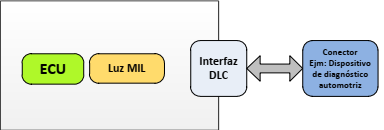


Figura 2 Componentes de sistema OBD-2 [Tomada de:(Socasi, 2016)]

El OBD2 no tiene mucha diferencia con el sistema OBD1, ya que constituye un perfeccionamiento de este sistema, lógicamente con ciertas diferencias bien definidas, como factores que entre ellos pueden destacar:  
los convertidores catalíticos, las líneas CAN, sensores, sistemas y forma de correr los diagnósticos computarizados, lo que representa una ayuda valiosa para la efectividad de este sistema en relación a los niveles de emisión (Tenorio Córdova & Coronel Magallanes, 2020).

Este sistema es el encargado de informar y almacenar los errores detectados por las unidades de control que tienen un automóvil, adicionalmente permite el acceso a dicha información a través de la interfaz DLC (Data Link Connector) y de un único conector. En la Figura 3 se muestra la interfaz DLC y su respectivo conector. (Socasi, 2016).



Figura 3 Interfaz DLC y conector [Tomada de: (Socasi, 2016)]

Como se puede observar en la anterior figura, la interfaz DLC cuenta con 16 pines que cumplen funciones específicas. En la Figura 4 se muestra la numeración de los pines, mientras que en la Tabla 1 se puede observar la función de cada pin.

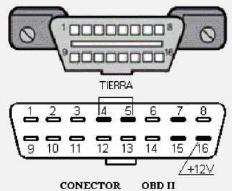


Figura 4 Numeración de pines de la interfaz DLC [Tomada de: (Socasi, 2016)].

Tabla 1 Funciones de pines de la interfaz DLC [Tomado de: (Socasi, 2016)].

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

**CARACTERÍSTICAS** (Tenorio Córdova & Coronel Magallanes, 2020)  
• El sistema OBD2 controla virtualmente todos los sistemas de control deemisiones y componentes que pueden afectar los gases de escape o **emisiones evaporativas.**  
• En muchos casos, un mal funcionamiento puede ser detectado antes quelas emisiones excedan en 1.5 veces los niveles estándar para emisionesa **50 mil millas o 100 mil millas.**  
• Si un sistema o componente ocasiona que se supere el umbral máximode emisiones o no operadentro de las especificaciones del fabricante unDTC(Diagnostic trouble code) debe ser almacenado y la lámpara MILdeberá encenderse dentro **de dos ciclos de conducción.**

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.

Tabla 2 Comparativa entre distintos software y escáneres para el sistema OBD2.[Tomada de :(Contrera Ramírez, 2020)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 | adquisición Fácil |
| 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 | Adquisición. Fácil |

## I.4 Computadora del automóvil

La ECU es la computadora del automóvil y su función principal es obtener y manejar los datos provenientes de todos los transductores del motor del automóvil. Es un dispositivo que se encuentra generalmente debajo del tablero en la parte del conductor (ISRAEL CERVANTES ALONSO & SAÚL OSBOR ́N ESPINOSA SOLÍS, 2010).



Figura 5 Computadora del automóvil [Tomada de:(Tenorio Córdova & Coronel Magallanes, 2020)]

La ECU está formada por tres etapas de funcionamiento (Dimaté Cáceres & González Castillo, 2013) como se puede apreciar en la figura 6.  
La etapa de entrada se encarga de recibir todas las señales provenientes de los sensores sean analógicas o digitales. El convertidor Analógico/Digital y el acondicionador de señales forman parte de esta etapa, y tienen la finalidad de digitalizar todas las señales para su posterior procesamiento (Dimaté Cáceres & González Castillo, 2013).  
La etapa de procesamiento está encargado de procesar los datos mediante la utilización de procesadores, memorias, unidades aritméticas y señales de sincronización (Dimaté Cáceres & González Castillo, 2013).  
La etapa de salida se constituyen por las señales de control de los actuadores como: la chispa de ignición, el pulso de control de inyectores, el encendido del ventilador, entre otras (Dimaté Cáceres & González Castillo, 2013).

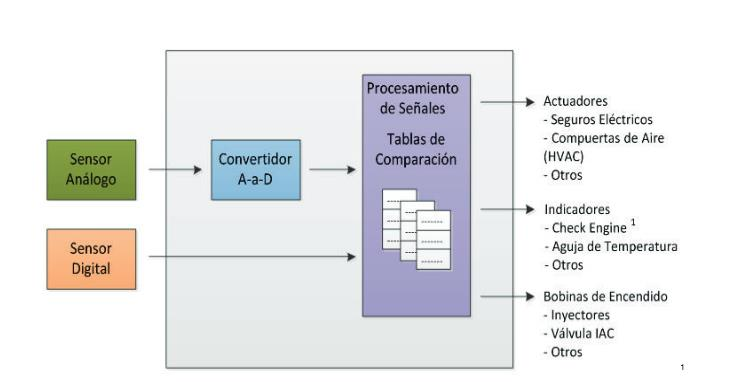


Figura 6 Etapas de funcionamiento de ECU [Tomado de:(Dimaté Cáceres & González Castillo, 2013)]

## I.5 Plataforma Arduino

Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirlo en una salida: activar un motor, encender un LED, publicar algo en línea. Puede decirle a su placa qué hacer enviando un conjunto de instrucciones al microcontrolador en la placa. Para ello se utiliza el lenguaje de programación Arduino (basado en Wiring), y el Software Arduino (IDE), basado en Processing (*What Is Arduino?*, s. f.).

A lo largo de los años, Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos hasta instrumentos científicos complejos. Una comunidad mundial de creadores (estudiantes, aficionados, artistas, programadores y profesionales) se ha reunido en torno a esta plataforma de código abierto, sus contribuciones se han sumado a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda tanto para principiantes como para expertos (*What Is Arduino?*, s. f.).

Arduino nació en el Instituto de Diseño de Interacción Ivrea como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta de simples placas de 8 bits a productos para aplicaciones IoT, wearables, impresión 3D y entornos integrados (*What Is Arduino?*, s. f.).

En la Figura 7 se observan las distintas placas de Arduino que existen:



Figura 7 Placas Arduinos [Tomado de:(Tenorio Córdova & Coronel Magallanes, 2020)]

## I.6 Variables

**RPM del Motor:**

Las Revoluciones por minuto (RPM) indican el número de vueltas completas que desarrolla el cigüeñal; mientras más altas sean las RPM, mayor es la potencia del motor. Además las RPM ayudan al conductor a saber cuándo se debe realiza un cambio de marcha, mientras el automóvil se encuentra en movimiento (*Definición de RPM - Qué es, Significado y Concepto*, s. f.).  
Las RPM se obtienen a través del sensor CKP (Crankshat Position), el cual se ubica cerca de un engrane dentado, que origina un pulso eléctrico de corriente alterna a medida que el diente del disco se acerca al sensor. Cuando se presiona el acelerador, el disco gira más rápido y como consecuencia de ello se generan mayores números de pulsos. El gráfico N. 9 muestra el funcionamiento del sensor CKP (Tenorio Córdova & Coronel Magallanes, 2020).



Figura 8 RPM del motor [Tomado de:(Tenorio Córdova & Coronel Magallanes, 2020)]

**Temperatura del motor:**

El líquido refrigerante cumple un papel importante en el automóvil, es el que absorbe el calor y mantiene al motor a una temperatura adecuada de funcionamiento. Si la temperatura del líquido refrigerante se encuentra entre 85ºC y 95ºC, el automóvil, la medida de temperatura del líquido refrigerante se obtiene a través del sensor ECT (Engine Coolant Temperature) (Tenorio Córdova & Coronel Magallanes, 2020).

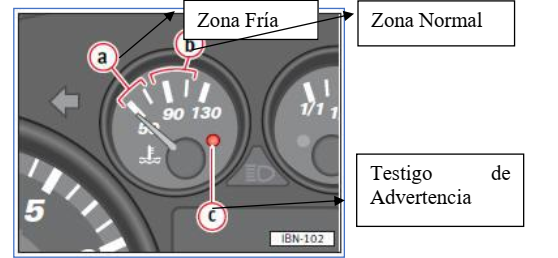


Figura 9 Temperatura refrigerante del motor [Tomada de : (Tenorio Córdova & Coronel Magallanes, 2020)]

**Posición de acelerador:**

La posición del acelerador indica el estado en el que el pedal se encuentra presionado. Esta medida es adquirida a través del sensor TPS (Throttle Position Sensor), permitiendo tener un control adecuado de la mezcla aire-combustible y de la cantidad de combustible que necesitan los inyectores. El funcionamiento del sensor TPS se basa en la variación de la resistencia provocada por el desplazamiento de un brazo cursor conectado al eje de la mariposa. El desplazamiento del brazo cursor hace que la resistencia varíe, permitiendo conocer la posición angular de la válvula de mariposa (Tenorio Córdova & Coronel Magallanes, 2020).

**Velocidad del automóvil:**

El sensor de velocidad del automóvil (VSS, Vehicle Speed Sensor) es un captador magnético que se encarga de enviar a la ECU la velocidad con la que recorre el automóvil, realizando el enriquecimiento de combustible durante la aceleración o el corte de combustible durante la desaceleración. (Tenorio Córdova & Coronel Magallanes, 2020)

En la Figura 10 se observa el funcionamiento del sensor.

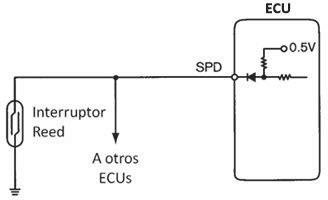


Figura 10 Funcionamiento del sensor VSS [Tomada de : (Tenorio Córdova & Coronel Magallanes, 2020)]

**Luz de fallas:**

La luz MIL es utilizada por el sistema OBD2 y se enciende cuando existe un problema o falla en el automóvil; su propósito es alertar al conductor de la necesidad de realizar una revisión del mismo. La luz MIL posee dos tipos de alerta: una es cuando la luz encendida es fija, lo que significa que se requiere una revisión inmediata del automóvil y la otra es cuando la luz encendida es intermitente, lo que significa que se requiere de una revisión del automóvil en el corto plazo (Dimaté Cáceres & González Castillo, 2013).

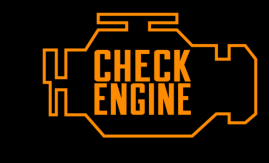


Figura 11 Luz Mil [Tomado de:(Tenorio Córdova & Coronel Magallanes, 2020)]

## I.7 Protocolos de Comunicación

En el sistema OBD2, los protocolos de comunicación permiten establecer la comunicación e intercambio de mensajes en forma bidireccional, entre una herramienta de diagnóstico escáner automotriz y la ECU del automóvil. Se puede determinar que sin tomar en cuenta la marca del vehículo, sea: Ford, Chevrolet, Hyundai, inclusive, si su procedencia es diferente: Americanos, Asiáticos o Europeos, existirá la manera de implantar comunicación con equipo de diagnóstico básico, que representa otra ventaja de este sistema, que posibilita la entrada a sus bases de datos, por los protocolos de comunicación (García Montero, 2013).

En el tabla 2 se presenta la comparación entre los cinco protocolos de comunicación que soporta el sistema OBD2.

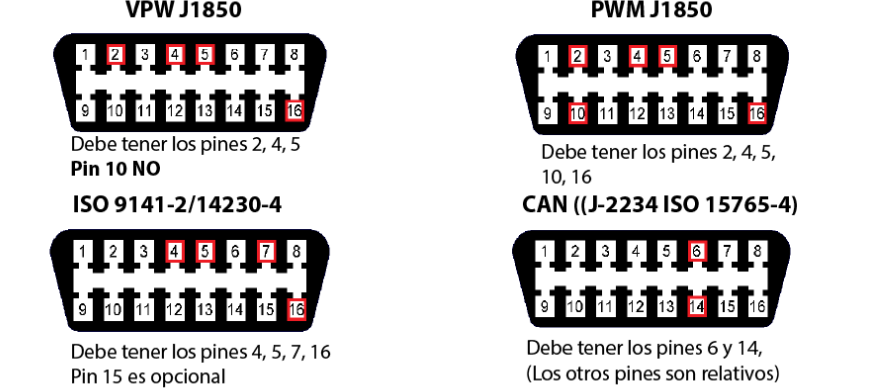


Figura 12 Pines de comunicación [Tomada de:(Tenorio Córdova & Coronel Magallanes, 2020)]

Tabla 3 Comparativa entre los 5 protocolos de comunicación [Tomada de: (García Montero, 2013)]

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

## I.8 Modos de Medición

El sistema OBD2 emplea nueve modos de medición, en el cual cada uno de los ellos permite el acceso a los datos de la ECU del vehículo. Para poder tener acceso es necesaria la utilización de códigos PID (Parámetros de identificación). Cada PID está relacionado con una medida específica de los  
modos 1 y 2 del sistema OBD2. Por ejemplo, si queremos obtener el dato en tiempo real de las revoluciones del automóvil, se debe ingresar al modo 1 y utilizar el PID 0E (ISRAEL CERVANTES ALONSO & SAÚL OSBOR ́N ESPINOSA SOLÍS, 2010).

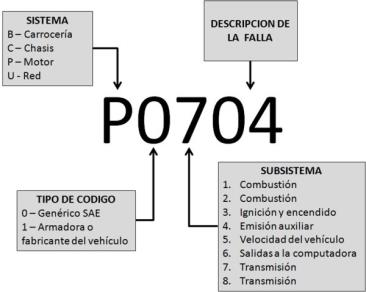


Figura 13 Modos de medición [Tomada de:(Tenorio Córdova & Coronel Magallanes, 2020)]

## I.9 Tecnologías

**Qt:**

Es un completo framework de trabajo que entrega herramientas para la creación multiplataforma de interfaces gráficas. Fue desarrollado por la empresa Quasar Technologies1, que posteriormente cambia su nombre a Trolltech, con el fin de contar con una herramienta que permitiera generar interfaces multiplataforma para proyectos escritos en C++. De esta forma, en 1994 fue lanzada la primera versión de Qt, Q por el primer nombre de la empresa que inicio su desarrollo (Quasar) y por la palabra toolkit, contando con soporte para los sistemas Linux y Windows. Producto de esto, en 2001 se añade soporte para MAC OS X, transformándose en una de las herramientas más utilizadas para la creación de entornos gráficos. En 2008 Trolltech es adquirido por Nokia, extendiéndose su aplicabilidad a sistemas m ́oviles. En cuanto a licencias de uso, Qt es distribuido bajo 3 diferentes tipos, GPL, LGPL y comercial. El primer tipo, GPL, está orientado al desarrollo de aplicaciones libres, por lo que los códigos de ellas y cualquier cambio realizado al código fuente de Qt deben estar disponibles a la comunidad. El segundo tipo, LGPL, permite la creación de aplicaciones cerradas, pero cualquier cambio realizado a los códigos del framework debe ser compartido. Por último, la licencia comercial permite realizar aplicaciones completamente cerradas. De esta forma, existen versiones de Qt gratuitas y de paga, dependiendo de la licencia bajo la que se rige el desarrollo. Así, de ser una herramienta utilizada en unos pocos proyectos, ha pasado a ser una de las m ́as utilizadas, siendo tal vez los proyectos m ́as conocido el gestor de ventanas utilizado en el proyecto KDE2, Photoshop y Google Earth. En la actualidad se encuentra disponibles para plataformas Linux, Mac OS y Windows (Orellana-Romero & Duran-Faundez, s. f.).

**PyQt:**

Conecta el marco multiplataforma Qt C++ con el lenguaje Python, es un módulo GUI.Qt es más que un conjunto de herramientas de GUI, por lo que presenta abstracciones de sockets o subprocesos de red, junto con Unicode, SQL, bases de datos, SVG, OpenGL, XML, un navegador web operativo, un sistema de servicio y una amplia gama de widgets de GUI.El principio sobre el que funciona una clase Qt está relacionado con un mecanismo de ranura responsable de ofrecer comunicación entre elementos con el fin de diseñar componentes de software reutilizables con facilidad.

Además, Qt viene con Qt Designer, una herramienta que actúa como una interfaz gráfica de usuario. PyQt puede diseñar código Python desde Qt Designer, al tiempo que agrega nuevos controles GUI cuando se usan tanto Qt Designer como el lenguaje de programación Python.(*What is PyQt? | Learn Python PyQt*, s. f.)

## I.10 Lenguajes

**Lenguaje Arduino**

El equipo que desarrolló la tarjeta Arduino creo un lenguaje propio para programar estas tarjetas. Este lenguaje está basado en un lenguaje llamado Wiring. Este es un framework (Un framework es una especie de plantilla, esquema o estructura conceptual de base tecnológica que nos permite trabajar de una manera mucho más sencilla) de código abierto para la programación de microcontroladores.(Concepción, 2020)

**Python**

Es un lenguaje de programación de **alto nivel** que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina.

Python es un lenguaje **sencillo de leer y escribir debido a su alta similitud con el lenguaje humano**. Además, se trata de un **lenguaje multiplataforma de código abierto** y, por lo tanto, gratuito, lo que permite desarrollar software sin límites. Con el paso del tiempo, Python ha ido ganando adeptos gracias a su sencillez y a sus amplias posibilidades, sobre todo en los últimos años, ya que facilita **trabajar con inteligencia artificial, big data, machine learning y data science**, entre muchos otros campos en auge. (*Python*, s. f.-a)

## I.11 Herramientas

**Qt Desinger:**

Es una herramienta de Qt que le proporciona una interfaz de usuario de lo que ve es lo que obtiene (WYSIWYG) para crear GUI para sus aplicaciones PyQt de manera productiva y eficiente. Con esta herramienta, crea GUI arrastrando y soltando objetos QWidget en un formulario vacío. Después de eso, puede organizarlos en una GUI coherente utilizando diferentes administradores de diseño.

Qt Designer también le permite obtener una vista previa de sus GUI usando diferentes estilos y resoluciones, conectar señales y ranuras, crear menús y barras de herramientas, y más.

Qt Designer es independiente de la plataforma y el lenguaje de programación. No produce código en ningún lenguaje de programación en particular, pero crea archivos .ui. Estos archivos son archivos XML con descripciones detalladas de cómo generar GUI basadas en Qt.

Puede traducir el contenido de los archivos .ui a código Python con pyuic5, que es una herramienta de línea de comandos que viene con PyQt. Entonces puede usar este código Python en sus aplicaciones GUI. También puede leer archivos .ui directamente y cargar su contenido para generar la GUI asociada.(Python, s. f.-b).

**Visual Studio Code:**

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Es un editor de código fuente súper rápido y liviano que se puede usar para ver, editar, ejecutar y depurar código fuente para aplicaciones. Es utilizado principalmente por desarrolladores **front-end**. Visual Studio Code es un producto completamente diferente en comparación con Visual Studio.

**Visual Paradigm:**Es una herramienta de gestión y diseño potente, multiplataforma y fácil de usar para sistemas de TI. ¡Visual Paradigm proporciona a los desarrolladores de software la plataforma de desarrollo de vanguardia para crear aplicaciones de calidad más rápido, mejor y más barato! Facilita una excelente interoperabilidad con otras herramientas CASE y la mayoría de los IDE líderes, lo que supera todo su proceso de desarrollo Model-Code-Implement en esta solución integral (*Visual Paradigm Product Overview*, s. f.).

## I.12 Metodologías de desarrollo de software. Las metodologías de desarrollo de software están conformadas por un grupo de procedimientos que ayudan al correcto desarrollo del ciclo de vida de un software de gran envergadura. Son de suma importancia, pues con su uso se puede llevar un control de cadauna de las fases de un proyecto y los resultados que se obtienen. Existen actualmente varias metodologías, cada una de ellas con características peculiares que las hacen distinguirse, son clasificadas en dos grupos: metodologías ágiles y tradicionales. Se deben tener en cuenta varios aspectos en el momento de escoger la que se va a utilizar en el proceso de desarrollo de un proyecto. Programación extrema (XP). Es una metodología ágil que se basa fundamentalmente en promover el trabajo en equipo. El principal objetivo que persigue, es lograr que el producto se termine en el tiempo requerido por los clientes, obteniendo la plena satisfacción del mismo. Es una metodología en la cual existe comunicación entre el cliente y los desarrolladores, pues el cliente forma parte del equipo de trabajo y este tiene el derecho a decidir qué es lo que se debe de implementar, hacer cambios en los requerimientos y exigir por la entrega de una parte del trabajo cada cierto tiempo. Esto demuestra la retroalimentación existente entre ambas partes, la correcta comunicación y forma de trabajo. Para la especificación de requisitos se utilizan las historias de usuarios, las cuales pueden ser totalmente flexibles a cambios o reemplazos. La programación puede llevarse a cabo por parejas de programadores, disminuyendo la cantidad de errores y también se realizan pruebas unitarias. Las cuatro fases que se definen en esta son: planificación, diseño, desarrollo y prueba (Beck, 2000). Proceso unificado de desarrollo de software (RUP). RUP pertenece al grupo de las metodologías tradicionales, es un proceso que puede ser utilizado tanto en proyectos de grandes dimensiones, como en pequeños proyectos. Responde al qué, quién, cómo, y al cuándo. El objetivo principal es el desarrollo de un producto con gran calidad que cumpla con los requisitos establecidos por el cliente. Posee tres características fundamentales: dirigido por casos de uso, centrado en la arquitectura e iterativo incremental. Para el modelado visual utiliza UML y es orientado a objetos. Entre las principales ventajas que presenta esta metodología se destacan la reducción del riesgo que existe de que el producto no sea terminado en el tiempo planificado, el desarrollo es sumamente acelerado y se puede adaptar fácilmente a las necesidades del cliente. El trabajo es particionado en nueve flujos y cuatro fases (Inicio, Elaboración, Construcción y Transición), donde cada una es un ciclo de desarrollo. En esta metodología se identifican 6 prácticas que permiten al equipo de desarrollo realizar un trabajo de forma eficaz (Jacobson et al., 2006): ⮚Gestión de requisitos. ⮚Desarrollo de software iterativo. ⮚Desarrollo basado en componentes. ⮚Modelado visual. ⮚Verificación continúa de la calidad. ⮚Gestión de los cambios.

## Conclusiones del capítulo

<La lista de conclusiones en este capítulo por lo general van dirigidas a establecer los argumentos [posición científica que adopta el autor] que respaldan las decisiones tomadas en relación al objeto de estudio y el campo de acción de la investigación, principalmente referidas a los referentes escogidos como pilares de la investigación>

# CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

<Introducción del capítulo 2 con una breve explicación del objetivo que persigue el capítulo, los principales contenidos que aborda, la estructura que puede encontrar el lector en su composición y un breve texto introductorio a las temáticas principales que aborda el capítulo>

## II.1 Nombre del Epígrafe II.1

<Generalmente este epígrafe se dedica al modelado de los procesos y subprocesos que componen el objeto de estudio, por lo que debe ser consecuencia del epígrafe del capítulo 1 donde se realizó la descripción de dicho proceso en términos textuales>

## II.2 Nombre del Epígrafe II.2

<Generalmente este epígrafe consta de los artefactos resultantes de la Ingeniería de Requisitos desarrollada; así como del modelado resultante del análisis y diseño de la solución propuesta, por lo cual debe contener la descripción de la arquitectura>

## II.3 Nombre del Epígrafe II.3

<Por lo general este epígrafe presenta el diseño de los mecanismos para el almacenamiento, procesamiento y transmisión de los datos; así como ejemplos de la implementación de estos mecanismos y de las interfaces gráficas de usuario de la solución propuesta>

## II.4 Nombre del Epígrafe II.4

<Por lo general este epígrafe presenta el diseño del tratamiento de los errores, así como el diseño del despliegue de la solución propuesta>

## Conclusiones del capítulo

<La lista de conclusiones en este capítulo por lo general van dirigidas a establecer los argumentos [posición científica que adopta el autor] a partir de la aplicación del enfoque, metodología y métodos de Ingeniería de Software aplicados para lograr cumplir el objetivo, resolver el problema científico planteado y obtener un resultado práctico; destacando la pertinencia de las decisiones tomadas a lo largo del proceso de desarrollo de software descrito>

<EJEMPLO DE ESTRUCTURA Y CONTENIDO DE ESTE CAPÍTULO:

Título del Capítulo: SoftCalTest: Una aplicación web para la evaluación de la calidad de software en la empresa SOFT-PLUS+

I.1 Modelado del proceso de evaluación de la calidad de software en la empresa SOFT-PLUS+

I.2 Requisitos, análisis y diseño de la aplicación web SoftCalTest

I.3 Diseño e implementación del almacenamiento, procesamiento y transmisión de los datos en la aplicación web SoftCalTest

I.4 Tratamiento de errores y despliegue de la aplicación web SoftCalTest

Conclusiones del capítulo>

# CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

<Introducción del capítulo 3 con una breve explicación del objetivo que persigue el capítulo, los principales contenidos que aborda, la estructura que puede encontrar el lector en su composición y un breve texto introductorio a las temáticas principales que aborda el capítulo>

## III.1 Nombre del Epígrafe III.1

<En sentido general este epígrafe presenta el diseño de los mecanismos utilizados para la verificación y validación de la solución propuesta, su ejecución y los resultados obtenidos>

## III.2 Nombre del Epígrafe III.2

<Por lo general este epígrafe contiene la aplicación de los métodos y técnicas científicos que demuestran la transformación lograda por la solución propuesta en el objeto de estudio, es decir los datos que demuestren el tránsito del estado actual descrito en el capítulo 1 al estado deseado de dicho objeto>

## III.3 Nombre del Epígrafe III.3

<Generalmente este epígrafe contiene el estudio de factibilidad para la realización de la solución propuesta que igualmente demuestra la viabilidad de la solución desarrollada>

## Conclusiones del capítulo

<La lista de conclusiones en este capítulo por lo general van dirigidas a establecer los argumentos y resultados que demuestran la veracidad, factibilidad y fiabilidad de la solución propuesta en términos de los datos obtenidos al aplicar técnicas y métodos de verificación y validación de software, técnicas y métodos de validación científica de la transformación o impacto sobre el objeto de estudio; así como la factibilidad económica de la solución propuesta>

<EJEMPLO DE ESTRUCTURA Y CONTENIDO DE ESTE CAPÍTULO:

Título del Capítulo: Validación de SoftCalTest como aplicación web para la evaluación de la calidad de software en la empresa SOFT-PLUS+

I.1 Verificación y validación de la aplicación web SoftCalTest

I.2 Impacto de la aplicación web SoftCalTest en la evaluación de la calidad de software en la empresa SOFT-PLUS+

I.3 Estudio de factibilidad del desarrollo de software de de la aplicación web SoftCalTest

Conclusiones del capítulo>

# CONCLUSIONES FINALES

<La lista de conclusiones finales por lo general van dirigidas a establecer los argumentos y resultados a los que se arribó en lo siguientes aspectos: (1) sistematización del estado del arte referido al objeto de estudio y el campo de acción, (2) diagnóstico del estado actual del objeto de estudio, (3) principales aspectos del análisis, diseño e implementación de la solución, (4) principales resultados de la validación de la solución propuesta. Deben apoyarse en los resultados obtenidos y descritos en la memoria y no en datos que no aparezcan en este documento. No pueden exceder una cuartilla en su extensión>

# RECOMENDACIONES

<La lista de conclusiones por lo general van dirigidas a establecer aquellos aspectos en los cuales la investigación puede continuar para su perfeccionamiento, mantenimiento o evolución en el tiempo. No deben constituir acciones no realizadas por omisión de etapas del proceso investigativo o ingenieril; ni ser demasiadas en número que cuestionen la completitud y pertinencia de la investigación realizada. No pueden exceder una cuartilla en su extensión>

# REFERENCIAS BIBLIOGRÁFICAS

* Anastasio, M., & Enrique, J. (2013). *Caracterización de los estilos de conducción mediante smartphones, dispositivos obd-ii y redes neuronales*. https://riunet.upv.es/handle/10251/21025
* Beck, K. (2000). *extreme programming eXplained: Embrace change*. Addison-Wesley.
* Concepción, R. (2020, julio 16). ¿Qué lenguaje de programación usa Arduino? *Rjconcepcion*. https://www.rjconcepcion.com/podcast/que-lenguaje-de-programacion-usa-arduino/
* Contrera Ramírez, J. A. (2020). *Usos del Puerto OBD2 para diagnóstico del motor de un vehículo desde un dispositivo móvil* [Tesis, Instituto Tecnológico de Huejutla]. https://rinacional.tecnm.mx/bitstream/TecNM/1127/1/JOSE%20ANTONIO%20CONTRERAS%20RAMIREZ.pdf
* *Definición de RPM - Qué es, Significado y Concepto*. (s. f.). Recuperado 4 de mayo de 2022, de https://definicion.de/rpm/
* Dimaté Cáceres, J. M., & González Castillo, P. M. (2013). Diseño de una interfaz gráfica en Labview para el diagnostico de vehículos por medio de OBD2. *instname:Universidad Pontificia Bolivariana*. https://repository.upb.edu.co/handle/20.500.11912/911
* Garcia Alcivar, B. X. (2015). *Analisis y deseño de un sistema para el proceso de esconaeo de los vehiculos*. https://dspace.ups.edu.ec/bitstream/123456789/10311/1/UPS-GT001210.pdf
* García Montero, M. (2013). *FACULTAD DE MECÁNICA ESCUELA DE INGENIERÍA AUTOMOTRIZ*. dspace.espoch.edu.ec/bitstream/123456789/2609/1/65T00068.pdf
* ISRAEL CERVANTES ALONSO & SAÚL OSBOR ́N ESPINOSA SOLÍS. (2010). *ESCÁNER AUTOMOTRÍZ DE PANTALLA TÁCTIL T E S I S QUE PARA OBTENER EL TÍTULO DE: INGENIERO EN COMUNICACIONES Y ELECTRONICA - PDF Free Download*. https://docplayer.es/1287330-Escaner-automotriz-de-pantalla-tactil-t-e-s-i-s-que-para-obtener-el-titulo-de-ingeniero-en-comunicaciones-y-electronica.html
* Jacobson, I., Booch, G., & Rumbaugh, J. (2006). *El proceso unificado de desarrollo de software*. Addison Wesley.
* Orellana-Romero, E., & Duran-Faundez, C. (s. f.). *Introducci´on al desarrollo de GUI’s mediante Qt*. 13.
* *Python: Qué es y por qué deberías aprender a utilizarlo*. (s. f.-a). Recuperado 5 de mayo de 2022, de https://www.becas-santander.com/es/blog/python-que-es.html
* Python, R. (s. f.-b). *Qt Designer and Python: Build Your GUI Applications Faster – Real Python*. Recuperado 5 de mayo de 2022, de https://realpython.com/qt-designer-python/
* Ramos Coria, D. A. (2014). *Diseño de un Sistema de Monitoreo OBD-II con comunicación GSM*. http://www.ptolomeo.unam.mx:8080/xmlui/handle/132.248.52.100/3407
* Simbaña Coyago, W. A. (2015). *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*. http://bibdigital.epn.edu.ec/handle/15000/10659
* Socasi, A. R. A. (2016). *RECOLECCIÓN DE DATOS DEL SISTEMA OBD II DE UN AUTOMÓVIL USANDO UN DISPOSITIVO ANDROID*. https://bibdigital.epn.edu.ec/bitstream/15000/16553/1/CD-7218.pdf
* Tenorio Córdova, B. de los Á., & Coronel Magallanes, C. E. (2020). *“Análisis y diseño de un sistema de seguridad y monitoreo vehicular usando dispositivos OBD2 GSM”* [Thesis, Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas. Carrera de Ingeniería en Networking y Telecomunicaciones]. http://repositorio.ug.edu.ec/handle/redug/49483
* *Visual Paradigm Product Overview*. (s. f.). Recuperado 5 de mayo de 2022, de https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963\_visualparadi.html
* *What is Arduino?* (s. f.). Recuperado 4 de mayo de 2022, de https://www.arduino.cc/en/Guide/Introduction
* *What is PyQt? | Learn Python PyQt*. (s. f.). Recuperado 5 de mayo de 2022, de https://pythonpyqt.com/what-is-pyqt/

# ANEXOS

<Contenido de los anexos con igual tipo de fuente Arial, pero a tamaño 11 puntos e interlineado 1.0 puntos. Debe tratar de sólo utilizarse aquellos anexos imprescindibles para complementar lo presentado en la memoria escrita y que no excedan las ocho (8) o diez (10 páginas). Deben aparecer uno a continuación del otro sin necesidad de saltos de página entre estos>