UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA E INDUSTRIAS

CARRERA DE INGENIERÍA AUTOMOTRIZ

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INTERACTIVO DE MANTENIMIENTO PREVENTIVO PARA VEHÍCULOS CON PROTOCOLO OBD-II.

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO AUTOMOTRIZ

JOSÉ LUIS TRUJILLO REDROBÁN

DIRECTOR: ING. ALEXANDER PERALVO. MSc.

Quito, junio 2016

FORMULARIO DE REGISTRO BIBLIOGRÁFICO

PROYECTO DE TITULACIÓN

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	**1721521662**
APELLIDO Y NOMBRES:	**TRUJILLO REDROBAN JOSE LUIS**
DIRECCIÓN:	**23 DE MAYO 0E8-216 Y S35C**
EMAIL:	**joseluis__jl@hotmail.com**
TELÉFONO FIJO:	**3032256**
TELÉFONO MOVIL:	**0982627585**

DATOS DE LA OBRA	
TITULO:	**Diseño e implementación de un sistema interactivo de mantenimiento preventivo para vehículos con protocolo OBD-II.**
AUTOR O AUTORES:	**Jose Luis Trujillo Redroban**
FECHA DE ENTREGA DEL PROYECTO DE TITULACIÓN:	**08 de junio 2016**
DIRECTOR DEL PROYECTO DE TITULACIÓN:	**Ing. Alexander Peralvo MSc.**
PROGRAMA	**PREGRADO** [x] **POSGRADO** [ ]
TITULO POR EL QUE OPTA:	**Ingeniero Automotriz**
RESUMEN:	**El presente proyecto se desarrolló mediante el diseño de un sistema interactivo que permite dar a conocer al conductor cuando el vehículo necesita mantenimiento o revisión de sus componentes, evitando así el daño prematuro de sus partes, aumentando la confiabilidad y seguridad del vehículo. El proyecto fue enfocado en vehículos que trabajen bajo protocolo OBD-II ya que el sistema extrae información de la ECU del vehículo para así por medio de una aplicación desarrollada en Android se pueda determinar los periodos correctos de mantenimiento del vehículo. La**

	implementación del sistema se realizó mediante la conexión del puerto OBD-II del vehículo a una placa de software libre Arduino Leonardo que a su vez por medio de tecnología de comunicación bluetooth transmite los datos a una tablet con sistema operativo Android, este equipo tiene instalado una aplicación que contiene los periodos de mantenimiento y permite administrar cuando se realizan estos. La aplicación da aviso al conductor mediante alertas visuales y sonoras, también cuenta con la opción de posponer las alertas que se generen para así poder llegar al lugar de servicio y poder dar mantenimiento al vehículo y reestablecer los periodos para su próximo mantenimiento. El sistema cuenta con varias seguridades para el control del buen uso del aplicativo en el vehículo, cuenta con la implementación del sistema de encendido en la aplicación, ya que el auto se enciende desde un botón en la tablet, este botón se encuentra condicionado al número de revoluciones que genera el motor, cuando las revoluciones sean iguales a cero se active el botón de encendido y cuando sean diferentes a cero el botón quede inhabilitado. Las alarmas que se ingresen en la aplicación dependerán del vehículo al cual se conecte ya que los intervalos de mantenimiento de sus partes vienen estipulados en el manual de usuario. Con la implementación de este sistema en el vehículo logramos reducir los daños imprevistos, alargando así la vida útil del vehículo haciéndolo más seguro y

	confiable para su uso diario.
PALABRAS CLAVES:	• **Sistema Interactivo de mantenimiento preventivo** • **OBD-II** • **Arduino Leonardo** • **App Inventor Android** • **ECU**
ABSTRACT:	**This project is developed by designing an interactive system that allows to inform the driver when the vehicle needs maintenance or its components, thus avoiding premature damage of its parts, increasing reliability and safety of the vehicle. The project was focused on working under vehicles OBD-II protocol as the system extracts information from the vehicle's ECU so through an application developed on Android can determine the correct vehicle maintenance periods. The implementation of the system was performed by connecting the OBD-II port of the vehicle to a board open source Arduino Leonardo which in turn through communication technology bluetooth transmits data to a tablet with Android operating system, this computer has installed an application It is containing maintenance periods and in turn allows you to manage when they are made. The application gives warning to the driver through visual and audio alerts, also has the option of postponing the alerts that are generated in order to reach the place of service and to maintain the vehicle and reset periods for their next maintenance. The system has several safeguards to**

	control the proper use of the application in the vehicle, has the implementation of the ignition system in the application, as the car starts from a button from the tablet, this button is conditioned to the number of revolutions generated by the engine, the ECU sends by a connector OBD-II the number of revolutions every half second to Arduino Leonardo that by a bluetooth transmits the data to the application to validating the condition, when revolutions are equal to zero the power button is activated and are different from zero when the button is disabled. Alarms that are entered in the application depend on the vehicle to which it is connected as change depending on vehicle maintenance intervals of parts which are specified in the manual. With the implementation of this system in the vehicle we reduce unforeseen damage, thus extending the life of the vehicle making it more secure and reliable for everyday use.
KEYWORDS	• **Interactive maintenance system** • **OBD-II** • **Arduino Leonardo** • **App Inventor Android** • **ECU**

Se autoriza la publicación de este Proyecto de Titulación en el Repositorio Digital de la Institución.

f:_____

José Luis Trujillo Redrobán

C.I:1721521662

DECLARACIÓN Y AUTORIZACIÓN

Yo, **JOSÉ LUIS TRUJILLO REDROBÁN,** CI: 1721521662 autor/a del proyecto titulado: **Diseño e implementación de un sistema interactivo de mantenimiento preventivo para vehículos con protocolo OBD-II**, previo a la obtención del título de **Ingeniero Automotriz** en la Universidad Tecnológica Equinoccial.

1. Declaro tener pleno conocimiento de la obligación que tienen las Instituciones de Educación Superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Autorizo a la BIBLIOTECA de la Universidad Tecnológica Equinoccial a tener una copia del referido trabajo de graduación con el propósito de generar un Repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, 08 de junio del 2016

f:_____

José Luis Trujillo Redrobán

C.I:1721521662

DECLARACIÓN

Yo **JOSÉ LUIS TRUJILLO REDROBÁN**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

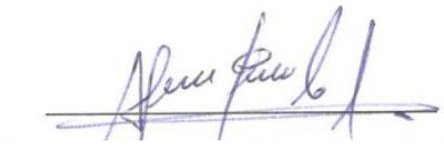
f: 

José Luis Trujillo Redrobán

C.I:1721521662

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título **"Diseño e implementación de un sistema interactivo de mantenimiento preventivo para vehículos con protocolo OBD-II."**, que, para aspirar al título de **Ingeniero Automotriz** fue desarrollado por **José Luis Trujillo Redroban**, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería e Industrias; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 19, 27 y 28.



Ing. Alexander Peralvo MSc.

DIRECTOR DEL TRABAJO

C.I. 1718133448

DEDICATORIA

Dedico esta tesis a mi familia, mi esposa e hijo que han sido mi gran motivación para cada día tener un espíritu de superación, a mis padres que siempre me demostraron la confianza y el orgullo que tienen hacia mí, mis abuelitos que con su cariño y apoyo han llegado a ser mis segundos padres y a mis tíos que siempre han estado ahí para darme una mano y brindarme un consejo.

José Luis Trujillo Redrobán

AGRADECIMIENTO

Doy gracias a Dios por brindarme salud y sabiduría para poder seguir adelante.

A mi esposa por ser un pilar fundamental en mi vida, le agradezco por siempre centrarme en la realidad y motivarme para mi superación.

A mi hijo José Antonio que con su inocencia y alegría me ayuda a olvidarme de los problemas y renovarme el ánimo para salir adelante.

A mi madre que nunca ha dejado de apoyarme y siempre ha estado conmigo dándome sabios consejos y brindándome todo su cariño.

A mi padre que siempre me enseñó a ser el mejor en lo que te depare la vida, nunca rendirte y siempre buscar superarte cada día.

A mi tío Héctor que siempre confió en mí y me dio todo su apoyo durante mis estudios y realización de esta tesis.

A mis amigos y compañeros de trabajo que supieron aconsejarme y ayudarme para poder finalizar mi tesis.

Al Ing. Alexander Peralvo por brindarme su apoyo, conocimientos y ser una guía importante para poder culminar este trabajo.

José Luis Trujillo Redrobán

ÍNDICE DE CONTENIDOS

PÁGINA

ÍNDICE DE TABLAS

PÁGINA

ÍNDICE DE FIGURAS

PÁGINA

ÍNDICE DE ANEXOS

PÁGINA

RESUMEN

El presente proyecto se desarrolló mediante el diseño de un sistema interactivo que permite dar a conocer al conductor cuando el vehículo necesita mantenimiento o revisión de sus componentes, evitando así el daño prematuro de sus partes, aumentando la confiabilidad y seguridad del vehículo. El proyecto fue enfocado en vehículos que trabajen bajo protocolo OBD-II ya que el sistema extrae información de la ECU del vehículo para así por medio de una aplicación desarrollada en Android se pueda determinar los periodos correctos de mantenimiento del vehículo. La implementación del sistema se realizó mediante la conexión del puerto OBD-II del vehículo a una placa de software libre Arduino Leonardo que a su vez por medio de tecnología de comunicación bluetooth transmite los datos a una tablet con sistema operativo Android, este equipo tiene instalado una aplicación que contiene los periodos de mantenimiento y permite administrar cuando se realizan estos. La aplicación da aviso al conductor mediante alertas visuales y sonoras, también cuenta con la opción de posponer las alertas que se generen para así poder llegar al lugar de servicio y poder dar mantenimiento al vehículo y reestablecer los periodos para su próximo mantenimiento. El sistema cuenta con varias seguridades para el control del buen uso del aplicativo en el vehículo, cuenta con la implementación del sistema de encendido en la aplicación, ya que el auto se enciende desde un botón en la tablet, este botón se encuentra condicionado al número de revoluciones que genera el motor, cuando las revoluciones sean iguales a cero se active el botón de encendido y cuando sean diferentes a cero el botón quede inhabilitado. Las alarmas que se ingresen en la aplicación dependerán del vehículo al cual se conecte ya que los intervalos de mantenimiento de sus partes vienen estipulados en el manual de usuario. Con la implementación de este sistema en el vehículo logramos reducir los daños imprevistos, alargando así la vida útil del vehículo haciéndolo más seguro y confiable para su uso diario.

ABSTRACT

This project is developed by designing an interactive system that allows to inform the driver when the vehicle needs maintenance or its components, thus avoiding premature damage of its parts, increasing reliability and safety of the vehicle. The project was focused on working under vehicles OBD-II protocol as the system extracts information from the vehicle's ECU so through an application developed on Android can determine the correct vehicle maintenance periods. The implementation of the system was performed by connecting the OBD-II port of the vehicle to a board open source Arduino Leonardo which in turn through communication technology bluetooth transmits data to a tablet with Android operating system, this computer has installed an application It is containing maintenance periods and in turn allows you to manage when they are made. The application gives warning to the driver through visual and audio alerts, also has the option of postponing the alerts that are generated in order to reach the place of service and to maintain the vehicle and reset periods for their next maintenance. The system has several safeguards to control the proper use of the application in the vehicle, has the implementation of the ignition system in the application, as the car starts from a button from the tablet, this button is conditioned to the number of revolutions generated by the engine, the ECU sends by a connector OBD-II the number of revolutions every half second to Arduino Leonardo that by a bluetooth transmits the data to the application to validating the condition, when revolutions are equal to zero the power button is activated and are different from zero when the button is disabled. Alarms that are entered in the application depend on the vehicle to which it is connected as change depending on vehicle maintenance intervals of parts which are specified in the manual. With the implementation of this system in the vehicle we reduce unforeseen damage, thus extending the life of the vehicle making it more secure and reliable for everyday use.

INTRODUCCIÓN

1. INTRODUCCIÓN

En la actualidad el parque automotor en todo el mundo se está incrementando y a su vez la tasa de conductores inexpertos que adquieren vehículos es alta, no existe un sistema práctico que informe y alerte al dueño del automotor sobre los próximos mantenimientos que su vehículo necesita, en la mayoría de casos estos mantenimientos no se realizan por descuido o falta de conocimiento, provocando así daños en los vehículos que pudieron evitarse al dar un mantenimiento periódico adecuado. Si se logra controlar que los conductores estén al tanto de los periodos correctos de mantenimientos preventivos en un vehículo se ayudará a disminuir la contaminación vehicular y evitar así daños a futuro en los automotores. Un sistema que cuente con alertas visuales y sonoras en cuanto a los periodos de mantenimiento y partes a ser controladas o reemplazadas es indispensable.

Un sistema interactivo de mantenimiento preventivo para un vehículo es indispensable, ya que la mayoría de conductores no conocen con exactitud los periodos correctos para reemplazar o controlar ciertas partes del vehículo. Con la creación de este sistema se obtendrá información de la ECU en cuanto al kilometraje que recorre el vehículo y así con un plan de mantenimiento instalado en el dispositivo táctil se logrará dar aviso al conductor de los periodos correctos de mantenimiento de su vehículo con alertas visuales y sonoras de ser el caso. El sistema controlará el encendido del vehículo con solo presionar un botón.

El objetivo del proyecto es diseñar e implementar un sistema interactivo de mantenimiento preventivo de vehículos con protocolo OBD-II.

Para poder lograr el objetivo del proyecto se desarrollará los siguientes puntos:

- Analizar el estado del arte para la transmisión de datos vía protocolo OBD-II.
- Investigar los planes de mantenimiento preventivo para vehículos.

- Diseñar un sistema amigable con el usuario para mantenimiento interactivo del vehículo.
- Implementar un sistema de mantenimiento preventivo en un vehículo con protocolo OBD-II.
- Realizar pruebas de funcionamiento del sistema interactivo de mantenimiento preventivo para vehículos.

El sistema interactivo en Android, tendrá la opción de crear diferentes planes de mantenimiento de vehículos con protocolo OBD-II, esta aplicación recibirá información de una placa Arduino Leonardo mediante conexión bluetooth y a su vez la placa Arduino Leonardo recibirá información de la ECU mediante un cable de conexión OBD-II de Arduino.

Este sistema se instalará en un vehículo Kia Cerato Koup y presentará la información del plan de mantenimiento por realizarse, adicional generará una base de datos con el registro de las alarmas activas y generadas para ese vehículo.

MARCO TEÓRICO

2. MARCO TEÓRICO

2.1 OBD-II

OBD-II, diagnóstico a bordo segunda generación (On board diagnostics second generation), es una normativa que rige para disminuir las emisiones en los vehículos, en adelante se usara las siglas OBD-II. (Agueda, 2009)

2.1.1 DEFINICIÓN

OBD-II es la segunda generación del sistema originalmente creado OBD (On Board Diagnostics), el sistema permite obtener un diagnostico electrónico del vehículo, también permite evaluar y supervisar el funcionamiento del motor para evitar la excesiva generación de emisiones, cuando algo está mal en el funcionamiento este sistema genera un código de falla y enciende la luz de check engine, toda la información que se recolecta para poder lograr este control viene de sensores instalados en el vehículo, el sistema OBD-II permite también generar protocolos de seguridad o emergencia en caso de que el vehículo llegase a fallar y la PCM se vea en la necesidad de controlarlo para la seguridad de los ocupantes hasta llegar al centro de servicio. (Agueda, 2009)

2.1.2 PROTOCOLO OBD-II

Existen 4 protocolos básicos en OBD-II, cada protocolo tiene variaciones con respecto al modelo de comunicación entre la PCM y el escáner. A pesar de existir algunos cambios de fabricante entre protocolos, como regla se tiene, los vehículos Chrysler, europeos y asiáticos usan protocolo ISO 9141. Los vehículos GM usan protocolo SAE J1850 VPW (modulación de ancho de pulso variable), los vehículos Ford usan patrones de comunicación SAE J1850 PWM (modulación de ancho de pulso) y CAN, su uso inicia en el año 2003 y de allí a la fecha, algunos modelos que los aplican son: Fiesta, Eco

Sport, Lobo, Explorer (Ford), Ram, Durango (Chrysler), Vectra, Malibu (GM), Murano, Saab, entre otros. (OBDII Experts, 2015)

2.1.2.1 ISO 9141

Es el protocolo de comunicación más antiguo y fue definido por ISO en 1989 gracias a la solicitud de la CARB (Junta de Recursos del Aire de California). Fue basado en la comunicación en serie que se basa en el bit 0 con cero voltios y el bit 1 con 12 voltios. La velocidad de transmisión estándar es de 10400 baudios. (OBDII Experts, 2015)

2.1.2.2 SAE J1850

Es el protocolo de comunicación estándar de arquitectura abierta y bajo costo, que usan 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 tipos: modulación por pulso variable (VPW) y modulación por ancho de pulso (PWM). (OBDII Experts, 2015)

2.1.2.2.1 PWM

Es la técnica de modulación en la cual se modifica el ciclo de trabajo de una señal periódica para transmitir por un canal de comunicación usada por Ford. PWM en OBD-II tiene una velocidad de transmisión estándar de 41600 baudios. (OBDII Experts, 2015)

2.1.2.2.2 VPW

Modulación utilizada por GM 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 0de 0 a 1, 64 y 128 microsegundos respectivamente. La velocidad de transmisión es de 10400 baudios. (OBDII Experts, 2015)

2.1.2.3 CAN

Controller Area Network ó CAN. Permite que se comuniquen varios dispositivos sin un host. La velocidad máxima de transferencia es de 1Mbit/s en redes con distancia menor a 40m. CAN fue estandarizado en 1986, y a partir del 2008, los automóviles livianos usan este protocolo. (OBDII Experts, 2015)

2.1.3 CONECTOR OBD-II

Un conector o socket de tipo OBD-II es de 16 pines no todos los pines son usados al momento de establecer una conexión. Generalmente para la transmisión de datos hacia un scanner se usa simultáneamente hasta 2 pines de los 16 como se puede observar en la figura 1. (Aficionados a la Mecánica, 2014)

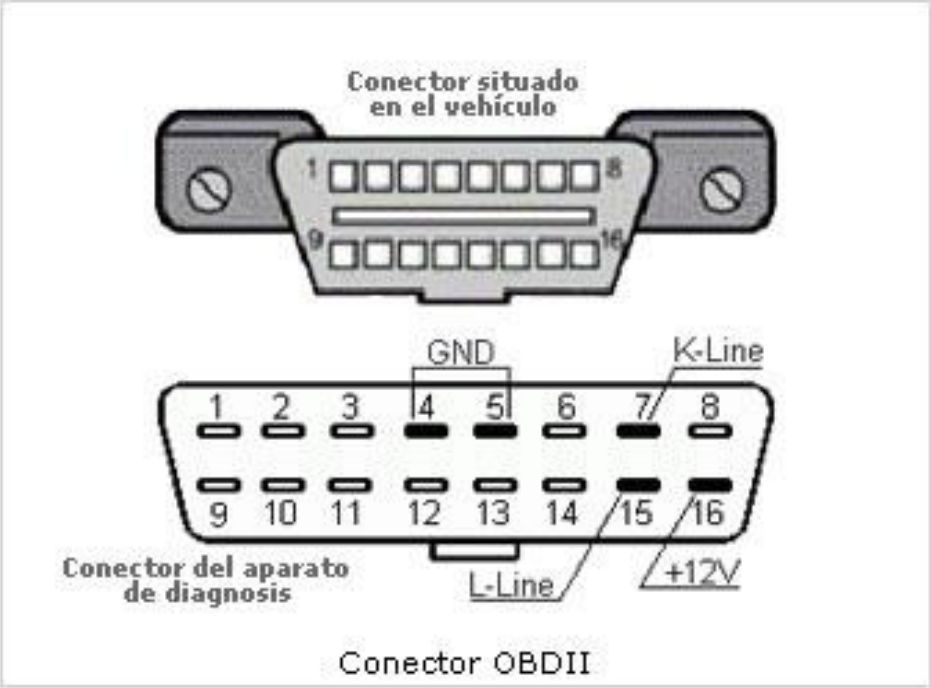


Figura 1. Configuración de pines en conector OBD-II.
(Aficionados a la mecánica, 2014)

PIN 2 - J1850 (Bus +)

PIN 4 - Masa del vehículo
PIN 5 - Masa de la señal
PIN 6 - CAN high (J-2284)
PIN 7 - ISO 9141-2 "Línea K"
PIN 10 - J1850 (Bus -)
PIN 14 - CAN low (J-2284)
PIN 15 - ISO 9141-2 "Línea L"
PIN 16 - Batería +

Después de conocer esto, se puede establecer que los pines 4, 5 y 16 son de alimentación del conector OBD-II, y el resto de pines serian para uso de la comunicación dependiendo del protocolo OBD-II que use el vehículo, el tipo de protocolo OBD-II se lo puede determinar por el fabricante del vehículo y año de fabricación del mismo, para reconocerlos se puede usar la tabla 1.

Tabla 1. Pines de comunicación en protocolos OBD-II.

PIN #	Descripción	Protocolo
2	J1850 (Bus +)	SAE J-1850
10	J1850 (Bus -)	
6	CAN High (J-2284)	CAN J-2284
14	CAN Low (J-2284)	
7	ISO 9141-2 "Línea K"	ISO 9141
15	ISO 9141-2 "Línea L"	

2.1.4 UBICACIÓN DE CONECTOR OBD-II

En el vehículo existen varios lugares en los cuales se puede encontrar el conector de 16 pines OBD-II la figura 2 muestra los más habituales. Para saber con exactitud la posición de los conectores OBD-II en un vehículo se debe investigar el manual de usuario ó a su vez el manual de taller del automóvil. (Teseo Motor, 2013)

Figura 2. Ubicaciones habituales del conector OBD-II.
(Teseo Motor, 2013)

2.2 ARDUINO

2.2.1 DEFINICIÓN

Arduino es una plataforma de electrónica abierta de software libre, es de fácil uso y flexible para crear prototipos o cualquier entorno u objeto interactivo. Arduino recolecta la información a través de sus pines de entrada por medio de sensores o alimentadores y envía señales hacia actuadores como luces, parlantes, motores, etc. Los fabricantes de las placas Arduino crearon el lenguaje de programación llamado Wiring que se usa para programar los microprocesadores de las diferentes versiones de placas Arduino que posee la marca. Al estar Arduino ya programado y en uso, no necesita que esté conectado a un ordenador ya que es independiente, su microprocesador es capaz de procesar la información y trabajar de forma estable, el uso de un computador se daría para el monitoreo de la placa y así poder realizar un test de la programación grabada. (Arduino, 2016)

2.2.2 COMUNICACIÓN

La comunicación con el computador se la puede realizar por medio de puerto serial o USB, solamente se necesita un cable serial a serial o cable mini USB. Esta comunicación se concatena a un puerto del computador el cual virtualmente es añadido en la aplicación de programación Arduino, la cual por medio de lenguaje Wiring permite generar y almacenar las rutinas que regirán al microprocesador de la placa Arduino. (Arduino, 2016)

2.2.3 ENTORNO ARDUINO

El entorno Arduino es el software que se encarga de la comunicación con la placa Arduino, este entorno permite la fácil programación de las diferentes placas diseñadas por Arduino, el entorno consta de herramientas y menús que ayudan en la creación de secuencias de programación viables para ser cargadas en el microprocesador de la placa Arduino, la presentación del entorno se puede observar en la figura 3. (Arduino, 2016)

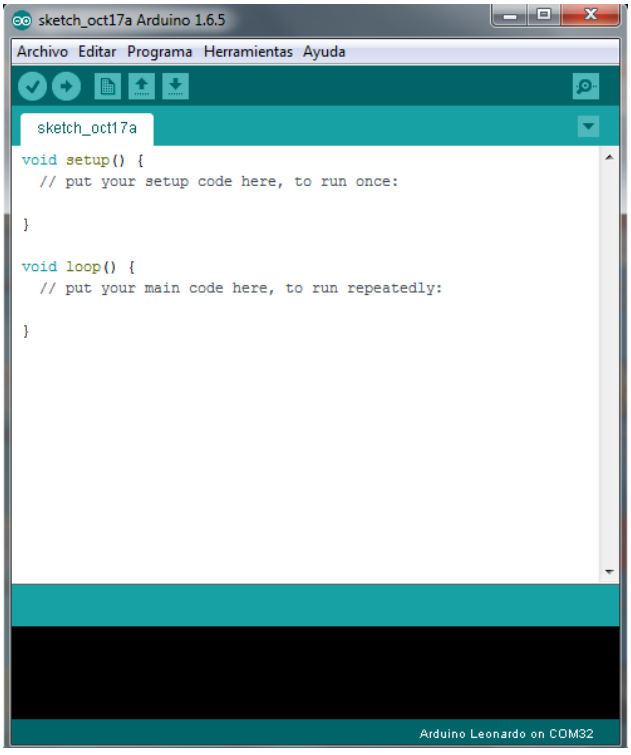


Figura 3. Entorno Arduino.

El entorno de Arduino genera bocetos de programación que podrán ser guardados y abiertos en ocasiones futuras, la extensión con la que Arduino guarda sus bocetos es “.ino” el entorno consta con todas las funcionalidades de introducción de texto, es decir copiar, pegar, reemplazar, haciendo más fácil y rápida la creación de bocetos, en la parte superior derecha consta con un icono para abrir el monitor serie, el cual está basado en hyperterminal para la transmisión de datos por un puerto COM virtual. Al momento de crear el boceto se puede verificar si la secuencia de comandos es la adecuada en cuanto a reglas y así poder subir una compilación de comandos adecuada a la placa Arduino. En la parte inferior izquierda el entorno consta con 2 botones los cuales son compilar y subir, como sus nombres lo indican la opción compilar realiza una revisión a la secuencia de comandos en busca de errores, por su parte el botón subir compila y sube la secuencia de comandos o boceto a la placa Arduino seleccionada. (Arduino, 2016)

El entorno Arduino permite también subir a su base librerías adicionales a las nativas, estas librerías pueden ser desarrolladas por terceros y servirán para poder concatenar la placa Arduino con sensores, actuadores, conectores de Arduino o externos a la marca, las bibliotecas deberán estar guardadas en “.zip” para su correcta adición a la base de datos. Antes de subir un boceto se debe configurar el puerto COM virtual generado para la conexión con la placa Arduino por parte del PC, adicional se deberá escoger el modelo de la placa Arduino que se esté usando. (Arduino, 2016)

El entorno Arduino está disponible en 30 idiomas diferentes, los cuales se encontraran en la lista de preferencias.

2.2.4 TARJETAS ARDUINO

Arduino ha creado varios modelos que se adaptan a las diferentes necesidades de velocidad y capacidad de procesador, como: (Arduino, 2016)

- Arduino UNO

- Arduino Duemilanove
- Arduino Diecimila
- Arduino Nano
- Arduino Mega
- Arduino Leonardo
- Arduino Mini
- Arduino Ethernet
- Arduino Fio
- Arduino BT
- Arduino Pro Mini
- Arduino Pro

2.3 CONECTOR OBD-II ARDUINO

2.3.1 DEFINICIÓN

El conector OBD-II de Arduino funciona como puente de datos del vehículo para Arduino, da acceso a los datos OBD-II con ayuda de la biblioteca de Arduino, así como fuente de alimentación (convertida y regulada desde el puerto OBD-II) para Arduino y los dispositivos conectados. El conector se observa en la figura 4 y su biblioteca es compatible para la conexión con todas las series de placas Arduino incluyendo los AVR de 8 bits basado en Arduino, también de 32 bits Arduino Due y Yun. (Arduinodev, 2015)

Figura 4. Conector OBD II Arduino.
(Arduinodev, 2015)

2.3.2 CARACTERÍSTICAS

- Se conecta directamente al puerto OBD-II del vehículo
- Interfaz de datos en serie (UART o I2C)
- Módulos de alto rendimiento para la salida de 5V / 3.3V DC hasta 2A
- Soporta bus CAN, KWP2000 y protocolos ISO9141/ ISO9141-2
- Fácil acceso a los PID OBD-II disponibles en la ECU del vehículo (Arduinodev, 2015)

2.3.3 COMPATIBILIDAD

El adaptador es compatible con los países y años de fabricación que se observan en la tabla 2.

Tabla 2. Compatibilidad de conector OBD-II.

País	Tipo	Año de Fabricación
Estados Unidos	Gas	1996+
Estados Unidos	Diésel	2004+
China, Korea	Gas ; Diésel	1998+
Canada	Gas	1998+
Europa + Reino Unido	Gas	2001+
Europa + Reino Unido	Diésel	2004+
Australia + Nueva Zelanda	Gas ; Diésel	2006+

El dispositivo es compatible con la mayoría de vehículos que tengan protocolo OBD-II como base para el funcionamiento de la ECU, generalmente la mayoría de vehículos fabricados desde el año 1996 pueden ser compatibles con el dispositivo, también se puede comprobar que protocolo de comunicación posee el vehículo levantando el capó y encontrando la etiqueta de la figura 5.

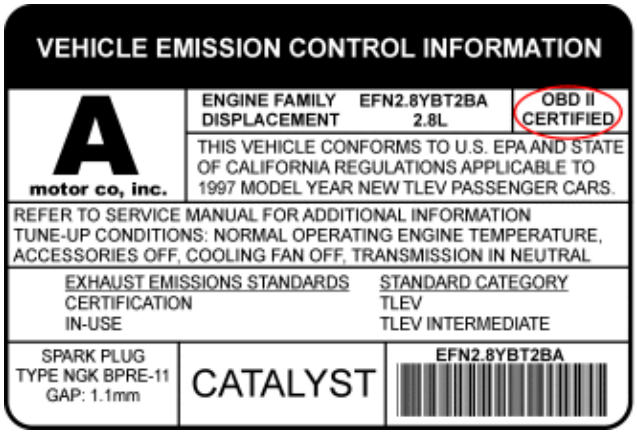


Figura 5. Etiqueta de certificación OBD-II.
(Arduinodev, 2015)

2.3.4 PROTOCOLOS DE COMUNICACIÓN

El conector OBD-II de Arduino es compatible con los protocolos especificados en la tabla 3. (Arduinodev, 2015)

Tabla 3. Protocolos de comunicación compatibles con conector OBD-II.

Protocolo	Velocidad
CAN	500 Kbps/11bit
	250 Kbps/11bit
	500 Kbps / Poder 29 bit
	250 Kbps / Poder 29 bit
ISO 9141/ ISO9141-2	250 / 500 Kbps
KWP2000	100 Kbps
	5 Kbps

2.3.5 CONEXIÓN

El conector OBD-II de Arduino en su otro extremo posee 4 cables con 2 sockets los cuales son para alimentación de energía y transmisión de datos

o información como se muestra en la figura 6, siendo su configuración la especificada en la tabla 4. (Arduinodev, 2015)

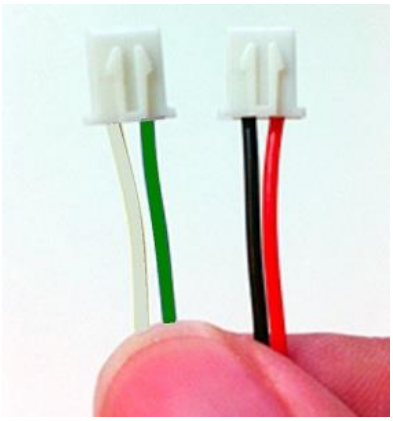


Figura 6. Sockets de conexión adaptador OBD-II Arduino.
(Arduinodev, 2015)

Tabla 4. Configuración de conectores adaptador OBD-II.

Tipo de Línea	Color	Nomenclatura	Descripción
Eléctrica	Rojo	VCC	Conexión a VCC de Arduino
Eléctrica	Negro	GND	Conexión a GND de Arduino
Datos	Blanco	RX	Cable a serie Tx de Arduino
Datos	Verde	TX	Cable a serie Rx de Arduino

2.3.6 PID'S DE COMUNICACIÓN

El conector Arduino consta de varios PID's de comunicación, el término PID viene del algoritmo de control el cual contiene 3 parámetros distintos: el proporcional, el integral, y el derivativo. Los PID's que acepta el dispositivo los designa el fabricante, limitando al programador el uso de las librerías gratuitas o bajo costo de ser el caso. Los PID's de comunicación son colocados en la secuencia de comandos, por ejemplo "PID_SPEED" para poder extraer la velocidad del vehículo. (Arduinodev, 2015)

2.4 TECNOLOGÍA BLUETOOTH

2.4.1 BLUETOOTH

Es un protocolo de comunicación de corto alcance, máximo 10 metros el cual está dentro de las tecnologías inalámbricas WPAN que permite transmitir datos en una banda ISM (banda de frecuencia industrial, científica y médica) de 2.4GHz, el bluetooth fue creado para eliminar los cables de conexión entre dispositivos y reemplazar directamente a la tecnología infrarroja. (Bluetooth, 2016)

La tecnología bluetooth tiene varios perfiles de transmisión de datos que son usados comúnmente, en la actualidad por lo menos un aparato electrónico está trabajando por medio de la tecnología bluetooth este método de transmisión es el más popular, su icono representativo se muestra en la figura 7. (PC World, 2015)

Figura 7. Símbolo de bluetooth.
(PC World, 2015)

2.4.2 PERFILES DE COMUNICACIÓN

Los perfiles de Comunicación bluetooth más usados son los que se utilizan para la transferencia de archivos o también llamados FTP, los utilizados para la transferencia de audio o como se denomina A2DP y la más popular que es la que da soporte a dispositivos de interfaces humanas o HID. (Bluetooth,2016)

2.4.3 EMPAREJAMIENTO Y SEGURIDAD

El emparejamiento en los dispositivos bluetooth es indispensable para la transmisión estable de datos, esta se la realiza una sola vez con cada dispositivo y existen varios niveles de seguridad que pueden ser configurados en nuestros equipos. Los niveles de seguridad pueden ir desde una clave de 4 dígitos hasta una encriptación avanzada de datos para poder conectarse al dispositivo. El emparejamiento se lo realiza con la MAC address del dispositivo bluetooth ya que esta es única y no puede existir una idéntica que pueda generar conflictos en la transmisión de datos. (Bluetooth,2016)

2.5 MÓDULOS BLUETOOTH PARA ARDUINO

2.5.1 INTRODUCCIÓN

Los módulos bluetooth para conexión con Arduino u otros microprocesadores son de fácil uso y programación, se pueden encontrar las versiones de bluetooth HC06 y HC05 para la integración con placas Arduino su presentación se observa en la figura 8. (Prometec, 2014)

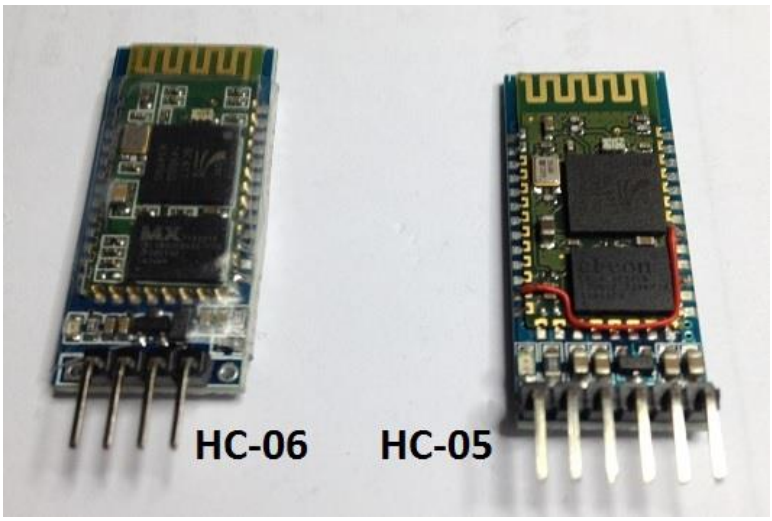


Figura 8. Bluetooth HC05 y HC06.
(Prometec, 2014)

Los bluetooth utilizan comandos AT para su configuración, la transmisión de datos la realizan por medio de los puertos TX y RX conectados de forma inversa al Arduino. (Prometec, 2014)

Se puede usar también en modo emulación, es decir que se utiliza otros pines de la placa Arduino para generar puertos TX y RX virtuales, así tener libres los puertos originales de la placa Arduino para otros componentes. (Prometec, 2014)

2.5.2 FUNCIONES

Entre los dos modelos HC05 y HC06 no existe diferencia de hardware sino más bien el firmware que controla al procesador es diferente. Las funciones que estos poseen los diferencian de forma muy fácil. (Prometec, 2014)

El módulo HC06 funciona solamente en modo esclavo, es decir que otro dispositivo se encargara de hacer el emparejamiento y búsqueda del módulo para la conexión y transmisión de datos. (Prometec, 2014)

El módulo HC05 por su parte tiene doble funcionalidad, este puede trabajar en función maestro o esclavo, así el mismo modulo con poca programación podrá encargarse de buscar, emparejar e iniciar la transmisión de datos con cualquier otro dispositivo. (Prometec, 2014)

Los dos módulos tienen las opciones para que por medio de comandos AT se cambie el nombre, clave de emparejamiento y en el dispositivo HC05 el método de trabajo es decir esclavo o maestro. (Prometec, 2014)

Los módulos bluetooth vienen preparados para ser conectados en un protoboard o directamente sobre una placa Arduino, por esto el modulo viene con pines de alimentación de energía, pines de transmisión, pines de recepción de datos y pines de configuración o también conocido como KEY

totalmente visibles e identificados como se observa en la figura 9. (Mcuoneclipse, 2015)

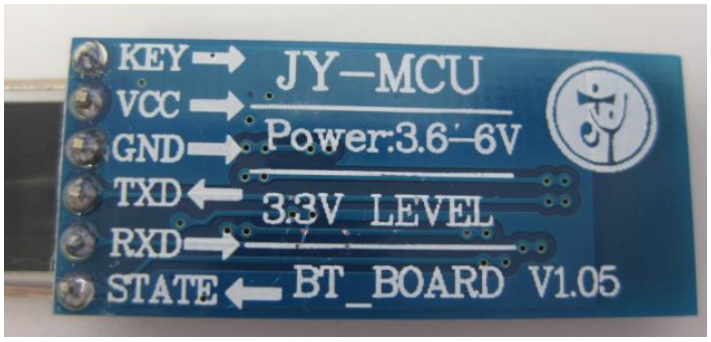


Figura 9. Pines de conexión modulo bluetooth HC05 y HC06. (Mcuoneclipse, 2015)

Los pines de conexión son:

- KEY: configuración comandos AT
- VCC: conexión a 3-6 voltios
- GND: conexión a tierra
- TXD: pin de salida de datos del módulo bluetooth
- RXD: pin de ingreso de datos del módulo bluetooth
- STATE: indica el estado del modulo (Mcuoneclipse, 2015)

2.5.3 COMANDOS AT

Los comandos AT son instrucciones en código que conforman un lenguaje de comunicación, se denominan comandos AT por la abreviatura de “attention”. (Mcuoneclipse, 2015)

Para la comunicación con comandos AT el dispositivo deberá estar conectado de forma serial (Tx/Rx) a un computador. La velocidad de conexión deberá ser la misma para el modulo bluetooth como para el hyperterminal del computador al cual se conecte. Y el pin KEY de nuestro modulo debe estar activado con 5 voltios. (Mcuoneclipse, 2015)

El comando de inicio para saber si un módulo acepta codificación o lenguaje de programación bajo comandos AT es “AT”, este se debe enviar por medio del hyperterminal, el dispositivo responderá con un “OK”, indicando que el modulo está listo para recibir instrucciones por parte del programador. (Mcuoneclipse, 2015)

Los comandos AT más utilizados en la configuración de un módulo bluetooth son:

- AT
- AT+VERSION
- AT+NAME
- AT+BAUD
- AT+PIN

Con estos comandos se podrá extraer la versión, el nombre, la velocidad de transmisión y la clave para el emparejamiento. Así también con estos comandos se podrá cambiar las configuraciones para personalizar el modulo.

2.6 ANDROID

2.6.1 DEFINICIÓN

El sistema operativo Android inicialmente fue creado para instalación en teléfonos móviles, como eran, BlackBerry OS, iOS y Symbian. La diferencia es que el sistema operativo Android está basado en Linux. (Android, 2015)

Las aplicaciones que trabajan en sistema operativo Android pueden ser desarrolladas bajo plataforma Dalvik que es una variación de Java. El desarrollo en Java es muy amplio y permite concatenar librerías que permitirían el acceso a todas las funciones de dispositivo como cámara, GPS, llamadas, etc. (Android, 2015)

El icono distintivo del sistema operativo Android se puede observar en la figura 10.

Figura 10. Símbolo de Android.
(Android, 2015)

2.6.2 ARQUITECTURA DE ANDROID

Android está internamente estructurada por cuatro componentes:

2.6.2.1 Aplicaciones

Todo dispositivo con sistema operativo Android contendrá aplicaciones de desarrollo bajo java, como SMS, GPS, contactos, notas, llamadas, correo electrónico, explorador de internet, etc. (Android, 2015)

2.6.2.2 Framework de aplicaciones

Para toda programación se tiene el libre acceso al código fuente de las aplicaciones base del sistema operativo Android para que de esta forma al desarrollar nuevas aplicaciones no se tenga que iniciar a programar desde cero. (Android, 2015)

2.6.2.3 Librerías

Android en su base de datos incluye un set de librerías C/C++, que están disponibles para todo desarrollador mediante el framework de las

aplicaciones Android System C library, librerías de gráficos, librerías de medios, SQlite, etc. (Android, 2015)

2.6.2.4 Runtime de Android

Son librerías desarrolladas en Dalvik que ayudan en la generación de nuevas aplicaciones, estas librerías pueden ser abiertas dentro del nuevo proyecto de desarrollo y simplifican la programación de funciones ya conocidas. (Android, 2015)

2.6.3 CARACTERÍSTICAS

- Máquina virtual Dalvik (basado en Java)
- Navegador basado en WebKit
- SQlite para almacenamiento de datos estructurados
- Soporta medios con formatos comunes de audio, vídeo e imágenes planas (MPEG4, H.264, MP3, OGG, AAC, AMR, JPG, PNG, GIF)
- Gráficos optimizados, con una biblioteca de gráficos 2D; gráficos 3D
- Telefonía GSM (depende del hardware)
- Bluetooth, EDGE, 3G, y Wifi (depende del hardware)
- Cámara, GPS, brújula, y acelerómetro (depende del hardware)
- Pantalla táctil
- Android Market para la comercialización gratuita o pagada para aplicaciones desarrolladas por Google o terceros
  (Android, 2015)

2.6.4 TABLET O DISPOSITIVO MÓVIL

Es una computadora portátil de mayor tamaño que un teléfono inteligente, esta viene integrada con una pantalla táctil y generalmente es de 7 a 12 pulgadas. El sistema operativo que ocupan estos dispositivos son generalmente Android o IOS desarrollado por Apple. (PC World, 2015)

La ventaja con respecto a otros dispositivos similares es su fácil uso en entornos pequeños, su peso ligero, la duración de su batería, entre otras. (PC World, 2015)

En estos dispositivos se puede instalar diversas aplicaciones que ayudarán en el día a día a sus usuarios, adicional cuentan con conexión WIFI y bluetooth que facilita la interacción y transmisión de datos, logrando con estas la conexión a internet. (PC World, 2015)

Actualmente Windows ya tiene en uso su sistema operativo en tablets lo que hace a estos dispositivos mucho más versátiles y útiles en las actividades cotidianas de la personas. (PC World, 2015)

Los fabricantes más conocidos de estos dispositivos son: Apple, Samsung y Hp, existen mucho más marcas que desarrollan estos dispositivos que cada día son más populares en el entorno. (PC World, 2015)

2.7 APP INVENTOR 2

2.7.1 HISTORIA

La plataforma App Inventor se puso a disposición del público el 12 de Julio del 2010 y está dirigida a usuarios que no cuenten con conocimiento extenso en programación, Google basó la creación del App Inventor en la Informática educativa, siendo así una principal y fácil herramienta para la creación de aplicaciones para el sistema operativo Android. (App Inventor, 2016)

2.7.2 DEFINICIÓN

Es una plataforma de acceso web gratuito creada por Google Labs y trabaja conjuntamente en la compilación y generación de aplicaciones con licenciamiento bajo la MIT (Massachusetts Institute of Technology), el portal

WEB está elaborado para la fácil creación de aplicaciones para el sistema operativo Android. Su uso es muy sencillo y de forma visual se pueden identificar un gran conjunto de herramientas básicas para que los usuarios puedan ir enlazándolas y así generando bloques de programación para la creación de aplicaciones. (App Inventor, 2016)

Google Labs junto con el MIT al realizar la creación de APP INVENTOR en su segunda versión esperan se incremente el número de aplicaciones desarrolladas por usuarios para Android ya que a más de ser gratuita los usuarios pueden colgar sus aplicaciones en Google Play para su distribución libre. (App Inventor, 2016)

2.7.3 CARACTERÍSTICAS

App Inventor utiliza la librería Open Blocks de Java, es decir basa su programación en Java pero la hace más amigable y la transforma en un lenguaje visual a partir de bloques. El compilador de las aplicaciones utiliza lenguaje Kawa que es distribuido por la Free Software Foundation todo esto bajo licencia libre (MIT License). (App Inventor, 2016)

App Inventor al ser una herramienta de fácil uso puede tener una aplicación sencilla en funcionamiento en una hora o menos y se puede programar aplicaciones más complejas en un tiempo mucho menor a que realizarlo con los lenguajes de programación tradicionales que se basan en texto. (App Inventor, 2016)

2.8 INTERRUPTOR DE ENCENDIDO DEL VEHÍCULO

2.8.1 DEFINICIÓN

El interruptor de encendido se encuentra en la columna de dirección del vehículo tiene un cable con alimentación directa de la batería, la función del

interruptor de arranque es comunicar esta corriente directa a las diferentes posiciones que posee un encendido o accionamiento de un vehículo. (Autobox, 2010)

El interruptor de encendido presentado en la figura 11 sirve en primer lugar para controlar que los accesorios no consuman la batería cuando el auto está estacionado por un largo periodo de tiempo, su segunda función es accionar la alimentación de todos sistemas que se involucran en el arranque del vehículo y como tercera función es accionar o alimentar el solenoide del motor de arranque y así el vehículo encenderá. (Autobox, 2010)

Figura 11. Interruptor de encendido del vehículo.
(Mecánica y Automoción, 2014)

2.8.2 POSICIONES DEL INTERRUPTOR DE ENCENDIDO

El interruptor de encendido generalmente consta de 4 posiciones como se puede observar en la figura 12, las cuales controlan los diferentes sistemas eléctricos del vehículo. (Tecnoficio, 2015)

- **Apagado:** en esta posición el interruptor corta el suministro eléctrico a todos sus controles principales de encendido y accesorios.
- **Accesorios:** en esta posición el interruptor envía energía al sistema de alimentación de accesorios, es decir se puede utilizar radio, luces, vidrios eléctricos, etc.

- **Contacto/Activado:** en esta posición el interruptor comunica la alimentación eléctrica hacia los sistemas de encendido del vehículo, es decir deja listo el automóvil para el arranque.
- **Arranque:** activa el solenoide del motor de arranque encendiendo así el vehículo, esta posición está conectada a un resorte que al finalizar el accionamiento o ignición, vuelve a su posición anterior en decir contacto.
  (Tecnoficio, 2015)

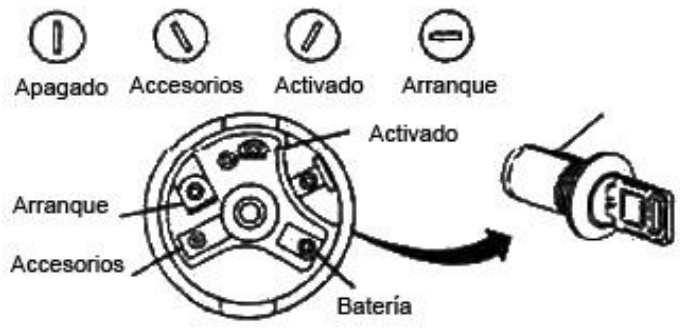


Figura 12. Posiciones del interruptor de encendido.
(Tecnoficio, 2015)

2.9 MANTENIMIENTO PREVENTIVO DEL VEHÍCULO

2.9.1 DEFINICIÓN

El mantenimiento preventivo del vehículo es una práctica periódica que todo conductor o dueño de un vehículo debe asumir. Siendo los vehículos máquinas, estas tienden a desgaste de sus partes, debido a esto debemos revisar de forma frecuente el funcionamiento y estado de las mismas. (Sura, 2015)

2.9.2 OBJETIVO

El objetivo del mantenimiento preventivo del vehículo es tener una conducción segura y confortable, así también para evitar daños futuros en

las partes que pudieron ser controladas con un buen mantenimiento periódico. (Sura, 2015)

2.9.3 PUNTOS DE REVISIÓN

Para el mantenimiento preventivo se debe revisar varios puntos alrededor de todo el vehículo los más comunes se muestran en la figura 13 y son:

- Sistema de refrigeración
- Motor (líquidos, bandas, mangueras, etc.)
- Transmisión
- Sistema eléctrico
- Sistema de Iluminación
- Sistema de frenos
- Llantas
- Sistema de suspensión
- Sistema de escape
- Dirección hidráulica
- Filtros

Figura 13. Puntos de control del auto.
(Sura, 2015)

METODOLOGÍA

3. METODOLOGÍA

Para la implementación del sistema interactivo en el vehículo se optó por el diseño de una placa de componentes en Proteus, junto con el uso de una placa Arduino Leonardo, un conector OBD-II de Arduino y una tablet con sistema operativo Android que soportará la aplicación a desarrollarse.

3.1 DISEÑO FUNCIONAL

3.1.1 PROTOCOLO OBD-II

Es una normativa que controla la ECU se puede decir que es el lenguaje de la ECU del vehículo, el sistema permite obtener un diagnostico electrónico del auto, también permite evaluar y supervisar el funcionamiento del motor para evitar la excesiva generación de emisiones, cuando algo está mal en el funcionamiento este sistema genera un código de falla y enciende la luz de “check engine”, toda la información que se recolecta para poder lograr este control viene de sensores instalados en el vehículo. (Agueda, 2009)

3.1.2 PLATAFORMA ARDUINO

Es una plataforma de software libre que permite programar secuencias o líneas de comandos a grabarse en un microprocesador que contiene una placa con entradas y salidas (analógicas/digitales), existen varios modelos de placas creadas por Arduino, estas varían en las capacidades que tienen con respecto a capacidad del procesador y periféricos. (Arduino, 2016)

3.1.3 CONECTOR OBD-II DE ARDUINO

Es un conector que sirve de interfaz entre la ECU del vehículo y la placa de Arduino, esta se comunica mediante comandos o PID’s designados en la normativa OBD-II. (Arduinodev, 2015)

3.1.4 COMUNICACIÓN BLUETOOTH

Es una tecnología de transmisión de información inalámbrica la cual permite comunicar varios dispositivos sin necesidad de cables, la transmisión depende de la capacidad o velocidad de procesamiento de la información. (Bluetooth, 2016)

3.1.5 SISTEMA OPERATIVO ANDROID

Es un sistema operativo de desarrollo gratuito, permite la fácil creación de aplicaciones que interactúen con las líneas periféricas que posea el dispositivo en el cual se encuentra instalado. Actualmente es un sistema operativo muy popular en celulares, tablet y computadores móviles. (Android, 2015)

3.1.6 PLAN DE MANTENIMIENTO DEL VEHÍCULO

Es un plan de revisión y mantenimiento que lo estipula el fabricante del automotor, se lo puede encontrar en el manual de usuario del vehículo y contiene información sobre los periodos recomendables para una revisión o cambio de partes. (Sura, 2015)

3.2 CARACTERÍSTICAS DEL PROYECTO

Para la realización del sistema se cuenta con el plan de mantenimiento del vehículo designado por el fabricante, los componentes electrónicos capaces de comunicarse y desencriptar la información extraída de la ECU del vehículo para plasmarla en la aplicación interactiva a desarrollarse en Android

El sistema interactivo permitirá al conductor del vehículo controlar los periodos correctos para el mantenimiento de las partes en funcionamiento.

3.3 PARÁMETROS PARA LA REALIZACIÓN DEL SISTEMA

Los parámetros a obtener de la ECU del vehículo son las RPM (revoluciones) y la distancia que recorre el vehículo en kilómetros, con estas variables se establecerán los intervalos de mantenimiento en la aplicación, la transmisión de datos es bidireccional ya que la tablet recibe datos de la ECU del vehículo y la tablet envía datos hacia el vehículo para controlar el encendido del mismo. El odómetro de la tablet aumentará conforme reciba los datos de la ECU y generará las alertas visuales y sonoras para el conductor. El aplicativo permitirá realizar la revisión de la alerta o posponer la misma para varios kilómetros después.

ANÁLISIS DE RESULTADOS

4. ANÁLISIS DE RESULTADOS

Para la demostración del funcionamiento del sistema se realizará la instalación en un Kia Cerato Koup del año 2010, como segundo paso se introducirá el kilometraje actual del auto en el aplicativo, se creará 3 alarmas con intervalos de 1,2 y 3 kilómetros, lo cual emitirá alarmas visuales y sonoras una vez pasadas las distancias antes señaladas y después de la última actualización realizada al odómetro, para verificar el funcionamiento de la opción posponer esta se configurará con 3 kilómetros de intervalo, por lo que se tendrá alarmas a los 4,5 y 6 kilómetros respectivamente. Posteriormente se encenderá el automóvil desde la aplicación presionando el botón “ENCENDER VEHÍCULO”, una vez verificadas las 6 alertas se apagará el vehículo y como paso final se realizará el mantenimiento en el sistema otorgando nuevos intervalos para su activación corroborando así el funcionamiento adecuado del sistema.

El plan de mantenimiento será cargado con los intervalos reales designados por el fabricante.

4.1 COMUNICACIÓN CON EL AUTOMÓVIL

4.1.1 PROTOCOLO OBD II KIA CERATO KOUP

El automóvil Kia Cerato Koup está dentro de los vehículos que utilizan protocolo OBD-II para la comunicación con la ECU, al ser protocolo estándar se puede usar los PID’S de comunicación ya desarrollados por los creadores del mismo, haciendo más viable la extracción de datos de la ECU.

La versión Kia Cerato Koup año 2010 utiliza el protocolo OBD-II ISO 9141-2, siendo sus pines de conexión el #7 con K-line y el #15 con L-line, como en todas las versiones de protocolo OBD-II se utiliza los pines 4 y 5 para GND o tierra y el pin 16 para la alimentación de 12V.

4.1.2 CONECTOR OBD-II KIA CERATO KOUP

El vehículo Kia Cerato Koup tiene el socket OBD-II de 16 pines ubicado en la parte inferior del panel frontal en el lado del conductor como se observa en la figura 14, este socket servirá para comunicar la ECU del vehículo con la placa Arduino Leonardo por medio del cable de conexión OBD-II de Arduino. Este socket también proporcionará alimentación eléctrica 12 voltios, que será transformada a 5 voltios por el conector OBD-II de Arduino para la alimentación de la placa Arduino Leonardo, esta alimentación no se interrumpe a pesar de que el vehículo esté apagado. Ya que la computadora del vehículo obtiene su voltaje directo de la batería.

Figura 14. Socket OBD II 16 pines Kia Koup.

4.2 CONECTOR OBD-II DE ARDUINO

Para la comunicación con la ECU del vehículo se utilizará un conector OBD-II como se observa en la figura 15 desarrollado por “Freematics by Arduino”. Este conector utiliza conexión UART con 5 voltios de alimentación y se encargará de comunicar la placa Arduino Leonardo con la ECU del vehículo.

Figura 15. Conector OBD-II Freematics by Arduino.

En el otro extremo del conector se puede encontrar los cables de alimentación y transmisión de datos, la configuración de estos se observa en la tabla 5.

Tabla 5. Conexión de puntos OBD-II con Arduino Leonardo

Tipo de Línea	Color	Nomenclatura	Descripción
Eléctrica	Rojo	VCC	Conexión a VCC de Arduino
Eléctrica	Negro	GND	Conexión a GND de Arduino
Datos	Blanco	RX	Cable a serie Tx de Arduino
Datos	Verde	TX	Cable a serie Rx de Arduino

El conector al estar siempre alimentado con 12 voltios, proporcionará el voltaje adecuado 5 voltios constantemente a la placa Arduino Leonardo utilizada para la extracción de datos de la ECU.

4.3 PID'S DE COMUNICACIÓN ARDUINO – OBD-II

El protocolo OBD-II cuenta con una serie de comandos para la comunicación con la ECU del vehículo, los PID's de comunicación se enviarán como instrucciones a la ECU la cual retornará el valor o dato solicitado por el

comando. Los PID’s de comunicación son estándar para todos los vehículos pero existen variantes ya que algunos vehículos no aceptan ciertos PID’s o no los interpretan como lo es esperado.

Un escáner automotriz ocupa estos PID’s para comunicarse con la ECU del vehículo los PID’s tienen varios modos de operación, ya sea solo para lectura o para sobrescribir datos de la ECU del vehículo.

El PID que utilizará la Arduino Leonardo para la programación es “PID_DISTANCE”, al enviar este PID a la ECU retornará la distancia que viaja el vehículo en cada evento, el vehículo toma como un evento desde el encendido hasta el apagado del vehículo.

Con el “PID_DISTANCE” la ECU retornará la distancia en kilómetros que viaja el vehículo en cada tramo, Arduino enviará la línea de comando a la ECU del vehículo cada cierto intervalo, el intervalo lo designa la programación, y en respuesta a esta solicitud se tendrá el dato en kilómetros a sumarse en el contador de la aplicación desarrollada.

4.4 BLUETOOTH HC05

El bluetooth HC05 es un interfaz de fácil uso y configuración 100% compatible con Arduino, la interfaz ofrece la comunicación de la placa Arduino Leonardo con un dispositivo móvil que tenga bluetooth.

4.4.1 CONFIGURACIÓN BLUETOOTH

Inicialmente se debe configurar el bluetooth HC05 que servirá de interfaz entre la placa Arduino Leonardo y el dispositivo táctil en este caso tablet Samsung Tab 4. Para la configuración se debe conectar el bluetooth HC05 a la placa Arduino Leonardo como lo indica la tabla 6, al conectar de forma inadecuada se provocará daños en la interfaz bluetooth.

Tabla 6. Configuración de conexión modulo bluetooth.

Tipo de Línea	Nomenclatura	Descripción
Eléctrica	VCC	Conexión a 3V de Arduino
Eléctrica	GND	Conexión a GND de Arduino
Datos	RX	Cable al pin 11 de Arduino
Datos	TX	Cable al pin 10 de Arduino

Las líneas de comunicación deben ir conectadas a los pines 10 y 11 respectivamente, y se debe alimentar la interfaz con 3 voltios y tierra del Arduino Leonardo como se observa en la figura 16.

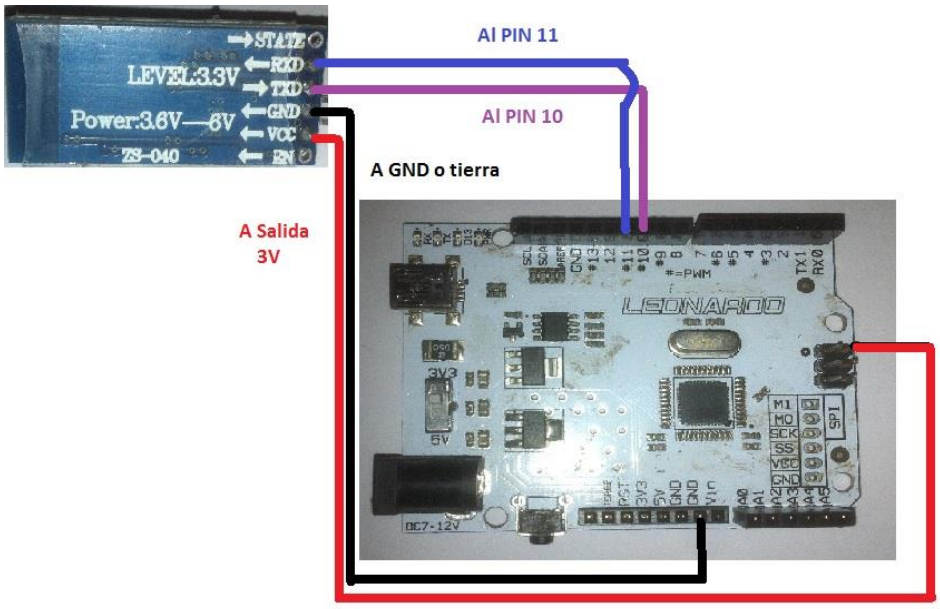


Figura 16. Conexión bluetooth HC05 – Arduino Leonardo.

Como segundo paso se debe encender la Arduino Leonardo e iniciar el entorno Arduino para establecer la secuencia de programación como se observa en la figura 17.

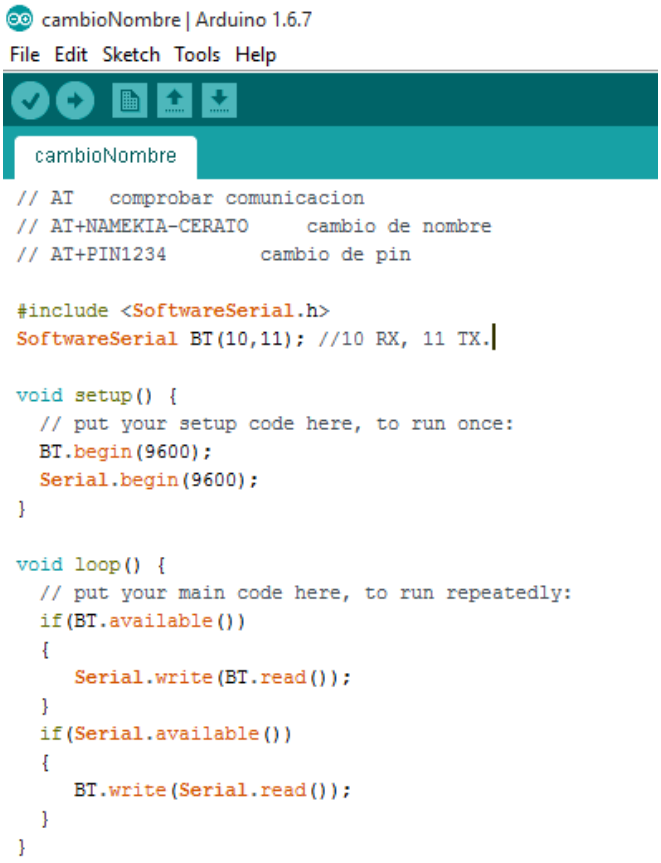


Figura 17. Programación en entorno Arduino para cambio de nombre HC05.

En donde:

- Incluye la librería para comunicación serial y designa los pines 10 para RX y 11 para TX en la placa Arduino Leonardo.
- Designa la velocidad de transmisión de la placa Arduino, en este caso 9600 bites.
- Establece la comunicación con el hyperterminal del entorno Arduino.

Una vez escrita la programación se debe compilar y enviar a la placa Arduino Leonardo por medio de cable USB.

Se abre el hyperterminal y se envía los códigos:

- **AT** para comprobar comunicación, la placa retornará un **OK**

- **AT+NAMEKIA-CERATO** para cambiar el nombre
- **AT+PIN1234** para establecer el PIN de seguridad los números 1234

Después de esto ya se tendrá el bluetooth HC05 configurado para la conexión y uso con la aplicación desarrollada. La programación podrá ser encontrada en el anexo 1.

4.5 ARDUINO LEONARDO

La placa Arduino Leonardo que se muestra en la figura 18 es una versión basada en un microcontrolador ATmega32u4, la cual tiene mayor facilidad de conexión ya que posee más entradas analógicas y digitales, 20 en total lo cual elimina los conflictos al conectar varios periféricos de entrada/salida a Leonardo, adicional su velocidad de procesamiento ayuda en la aplicación de proyectos de mayor complejidad.

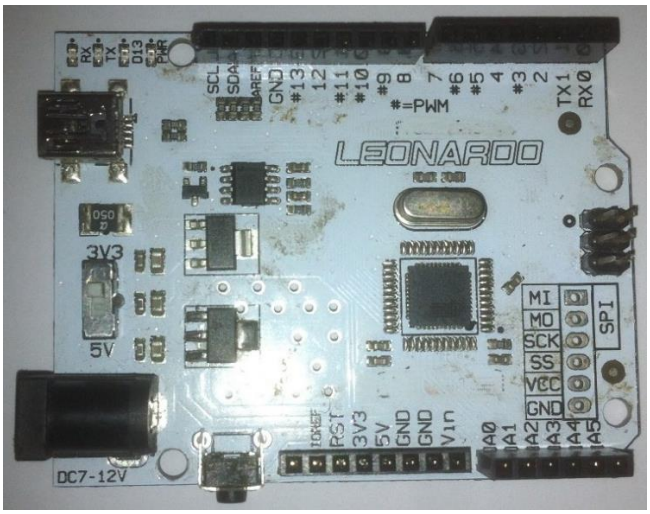


Figura 18. Placa Arduino Leonardo.

4.5.1 CONFIGURACIÓN ARDUINO LEONARDO

La Arduino Leonardo controlará la extracción de información de la ECU del vehículo Kia Cerato, por lo que se cargará en su memoria programación para este fin, una vez cargada la programación en la Arduino Leonardo esta

inicializará sola y extraerá la información de la ECU después de cada intervalo designado, en este caso se extraerá la información cada medio segundo.

La programación de la Arduino Leonardo se realizará en el entorno Arduino generando un nuevo proyecto y colocando la programación descrita en el anexo 2. En donde:

- Se incluye las librerías de Arduino y otros fabricantes para la programación del proyecto
- Se designa pines de comunicación en Arduino Leonardo
- Se designa las variables que tendrá el programa
- Se inicia las librerías y se establece velocidades de transmisión
- Se establece las acciones a realizar cuando el bluetooth esté conectado al dispositivo móvil.
- Se establece los comando que contendrá la acción calcular distancia.
- Se designa que variables se enviarán por serial y cuales por bluetooth.

Una vez que se termina la programación, se compila y envía por cable USB a la placa Arduino Leonardo, la cual almacenará esto en su memoria y empezará a correr cuando se conecte por medio del OBD-II Arduino al socket de 16 pines.

Para poder monitorear el programa se podrá conectar a un computador por medio de USB y abriendo el hyperterminal se logrará verificar los datos que son extraídos por Leonardo de la ECU del vehículo.

4.6 PROGRAMACIÓN DE APLICACIÓN EN ANDROID

Para la programación en Android se usará el portal APP Inventor 2, este portal gratuito de software libre ayudará a generar el aplicativo que recibirá la información recolectada por la Arduino Leonardo.

El App Inventor 2 generará un APK una vez compilada la programación y esta deberá ser instalada en la tablet Samsung.

4.6.1 GENERACIÓN DEL PROYECTO EN APP INVENTOR 2

Como primer paso se debe ingresar en el portal web ai2.appinventor.mit.edu, el único requisito será ingresar una cuenta de google para poder acceder a todas sus herramientas de forma gratuita.

Una vez ingresado al portal se debe generar nuevo proyecto y darle un nombre a este como lo muestra la figura 19.

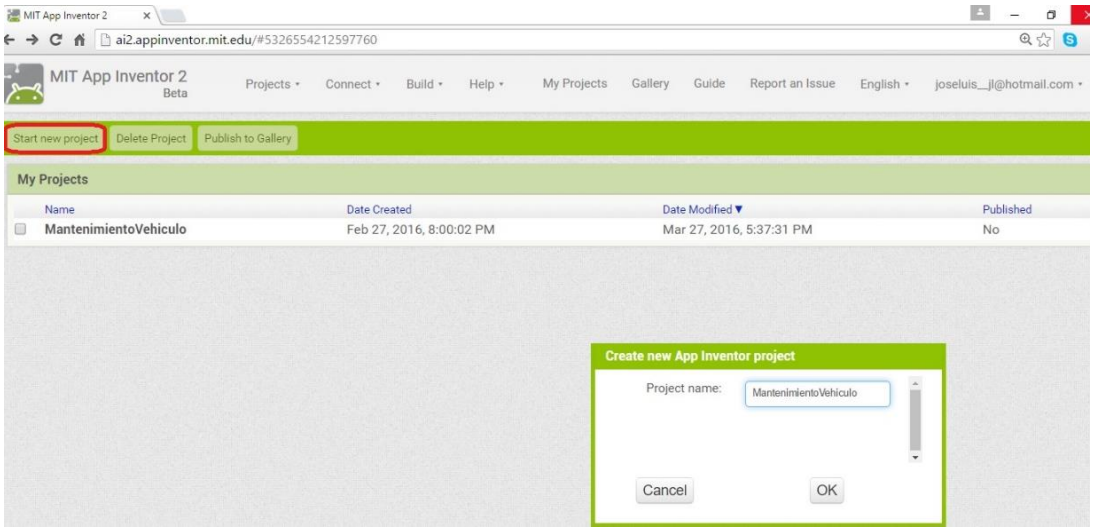


Figura 19. Generación del proyecto en App Inventor 2.

El proyecto contará con tres pantallas (screens) las cuales deben ser creadas y desarrolladas independientemente. Las pantallas se interconectarán y utilizarán la misma base de datos para funcionar. La programación de los screens se encontrará en el anexo 3

Como pantalla inicial se tendrá la selección del vehículo como lo muestra la figura 20, aquí existirá un hipervínculo que desplegará todos los dispositivos bluetooth, o vehículos con el sistema Arduino antes descrito que podrán conectarse a la aplicación.

Figura 20. Screen uno de la aplicación en Android.

La programación del screen uno se basará en la generación de la base de datos, nombre del programa de mantenimiento seleccionado y paso a la siguiente ventana o screen de la aplicación, esta ventana se inicializara al presionar el icono de la aplicación instalada en la tablet con sistema operativo Android.

Como primer paso se indica que inicialice la base AutosDB con el nombre y la MAC address del dispositivo bluetooth del vehículo conectado por Arduino, cuando se presione el botón **"Seleccionar Vehículo"** el aplicativo despliegue los bluetooth disponibles encontrados por la tablet. Cabe indicar que antes de realizar la conexión por medio de esta función el bluetooth HC05 del vehículo deberá ser emparejado a la tablet. Después de esto guardará esta información en la base de datos AutosDB y pasará al screen número dos.

El screen dos que se muestra en la figura 21 será el principal de la aplicación, aquí estarán trabajando de forma activa la transmisión bluetooth, almacenamiento en la base de datos y contadores para la activación de alarmas designadas al incremento del odómetro.

Una vez inicializada esta ventana no se podrá retornar o cambiar a otro screen hasta detenerla.

Figura 21. Screen dos aplicación en Android.

En esta ventana se muestra el nombre del vehículo así como en tiempo real las RPM junto con el odómetro, cuando esta ventana no esté inicializada permitirá configurar cada cuantos kilómetros se pospondrán las alarmas activas, en la opción tiempo permitirá configurar cada que intervalo de tiempo la tablet actualizará los datos, como parámetro recomendable es 0.5 segundos. En la sección inferior se tendrá la opción de guardar la información aunque esta ya cuenta con una opción de autoguardado, permite también configurar las alarmas cambiando al screen tres, el aplicativo cuenta con un botón para poder encender el vehículo con esto se asegura que el auto no sea encendido mientras no se conecte a el sistema ya que este botón se habilita únicamente cuando el screen dos está inicializado, esta acción la realiza presionando el botón iniciar.

El botón **“Encender Vehículo”** está concatenado a las RPM, es decir cuando las RPM son diferentes a cero este botón se deshabilita y cambia su nombre a **“Vehículo Encendido”**, esto para evitar que cuando el vehículo

este encendido se presione el botón y vuelva a activar el motor de arranque haciendo que este último pueda llegar a dañarse. La programación del screen dos se encontrará en el anexo 3.

En el Screen dos se iniciará todas las variables designadas para la aplicación, cada variable tendrá un espacio o sección en la base de datos AutosDB, así también se designará una regla, que si requieren salir de esta pantalla o regresar a la anterior cuando esta esté inicializada emita una alerta en forma de notificación, se debe recordar que el screen dos tendrá dos estados uno activo y uno inactivo. Adicional se programará las funciones de los botones que se tiene en el screen dos, cabe recordar que el botón “Encender Vehículo” está concatenado dentro del estado inicializado del screen dos y dependerá su actividad de las RPM recibidas del vehículo.

Al presionar “Guardar” se está indicando que guarde en la base de datos AutosDB el valor del odómetro, los tiempos de lectura de las alarmas, las alarmas que han sido pospuestas y los datos que han sido actualizados durante esa sesión.

Al presionar el botón relay “Encender Vehículo” lo que hará es enviar el número 1 a la placa Arduino, con esto la placa Arduino Leonardo enviará un pulso de 5V al transistor que activará el relay conectado al encendido del vehículo, como este botón está concatenado a las RPM hará las funciones antes mencionadas siempre y cuando el Screen dos esté inicializado y las RPM sean iguales a cero, caso contrario este botón se deshabilitara.

Al presionar el botón “Configurar Alarmas” lo que hará es enviar al screen número tres, el cual tendrá todas las opciones para crear, habilitar y configurar las alarmas de las cuales constará el aplicativo de mantenimiento preventivo del vehículo. Por último se configura el botón “Iniciar” el cual llevará el aplicativo al estado activo del screen dos y donde la aplicación operará la mayoría de su tiempo, al presionar el botón se programa que la

aplicación se conecte al bluetooth seleccionado en el screen uno, el cual está conectado a su vez por medio de la Arduino Leonardo a la ECU del vehículo, si la conexión se realizó sin problemas emitirá la notificación “Conexión Exitosa” caso contrario indicará “No se puede realizar la conexión con el dispositivo” y una vez conectado el botón “Iniciar” cambiará su nombre a “Detener” para poder retornar el screen dos a su estado inactivo, adicional se indica que al presionar “Detener” guarde los datos que se han modificado en la aplicación, así también que emita una notificación indicando “desconexión exitosa”.

También en el Screen dos se programará el comportamiento en estado activo, donde toda la información generada se guardará en la base de datos AutosDB y la información que se muestra en el screen será rescatada u obtenida de la base de datos es una comunicación bidireccional de la información de la base de datos. Dentro del estado inicializado del screen dos se debe programar la transmisión de información por medio de bluetooth, aunque la placa Arduino Leonardo recolecte la información de la ECU cada décima de segundo la transmisión hacia la tablet la controla la aplicación con esto se está indicando que la información del bluetooth que llega en bytes sea ingresada en las variables odómetro y RPM, la variable odómetro servirá para la activación y control de las alarmas programadas en el aplicativo de mantenimientos y las RPM servirán para controlar el botón “Encender Vehículo” aquí se indica que si las RPM son diferentes a cero el botón cambie a “Vehículo Encendido” como en todos los bloques anteriores se indica el llamado a la funciona guardar, que en este caso sería algo similar a un autoguardado.

El Screen dos también controlará el comportamiento de las alertas o alarmas creadas, lo primero que realiza la aplicación es una comprobación de las alarmas creadas, verifica si están activas o desactivas. Luego guarda temporalmente el próximo kilometraje de la alarma y de esta estar pospuesta lo guarda para su próxima notificación. Luego se programa la forma en la

cual se notifican las alarmas activas o que se activaron durante el evento y emite una alerta auditiva que está declarada como sound1, luego designa variables temporales de las alarmas activas las cuales se podrán modificar cuando el screen dos no este inicializado y se pueda acceder al screen tres por medio de el botón “Configurar Alarmas”.

Por último se creará el screen número tres ilustrado en la figura 22, el cual será la ventana de mantenimiento y creación de alarmas.

Figura 22. Screen número tres.

Este screen servirá para crear y dar mantenimiento a las alarmas de la aplicación Mantenimiento Vehículo UTE, los botones disponibles son para guardar cambios, crear una nueva alarma y eliminar la alarma seleccionada. Desde este screen se puede acceder a la lista de alarmas y se puede visualizar el estado, nombre y cada que kilometraje estas deben activarse. También el aplicativo tiene la opción de habilitar y deshabilitar las alarmas generadas.

La programación del screen número tres se encontrará en el anexo 3

En esta sección se inicia todas las variables de la base de datos para poder precargar las alarmas ya definidas y creadas por defecto o por el usuario, esta acción extrae la información de la base de datos AutosDB, adicional se programa que al presionar el retorno, cierre la ventana número 3 y retorne a la 2, también se indicará las acciones y botones disponibles en el screen, teniendo así disponibles los botones; “Guardar Cambios”, “Guardar Nueva”, “Eliminar” y “Realizar Revisión”.

El botón “Guardar Cambios” lo que hace es indicar la modificación de todos los parámetros ingresados por los que se encuentran en la base de datos incluso cambiando el estado y configuración de la alarma de haber realizado algún cambio.

Para el botón “Guardar Nueva” la aplicación va a generar un nuevo campo en la base de datos e ingresará los datos de kilometraje, nombre, estado y configuración de la alarma. No existe un límite de creación de alarmas para el dispositivo.

El botón “Realizar Revisión” será ocupado cuando una alarma haya sido activada por el aplicativo y ya con el vehículo en el taller una vez realizada la inspección validar digitalmente la misma y restablecer los contadores de la alarma.

El botón “Eliminar” lo que hará es conectar a la base de datos AutosDB y eliminar completamente la sección con el nombre designado para la alarma.

Deberá ser configurada la lista de las alarmas generadas en la aplicación, aquí se designará el comportamiento de las mismas y cómo van a presentarse ante el screen número 3. Al seleccionar la lista de alarmas que tiene el dispositivo se desplegará el estado de la misma si esta activa en rojo y si está inactiva en verde, adicional se habilitarán todos los campos de ingreso de datos y botones para poder modificar la alarma ya que despliega

y abre todas las variables guardadas en la base de datos con respecto a la alarma seleccionada

4.6.2 AGREGAR SONIDO DE LA ALERTA APP INVENTOR 2

Para agregar el sonido de la alerta que tendrá la aplicación se debe seleccionar en el App Inventor 2 el screen numero dos ya que en este en estado activo se usará el sonido ingresado para la alerta.

Se selecciona la variable Sound1 dentro de la ventana componentes y en las propiedades del componente se escoge el tiempo de reacción para la activación, luego se carga el archivo de audio con extensión MP3 para que sea agregado al aplicativo como lo muestra la figura 23.

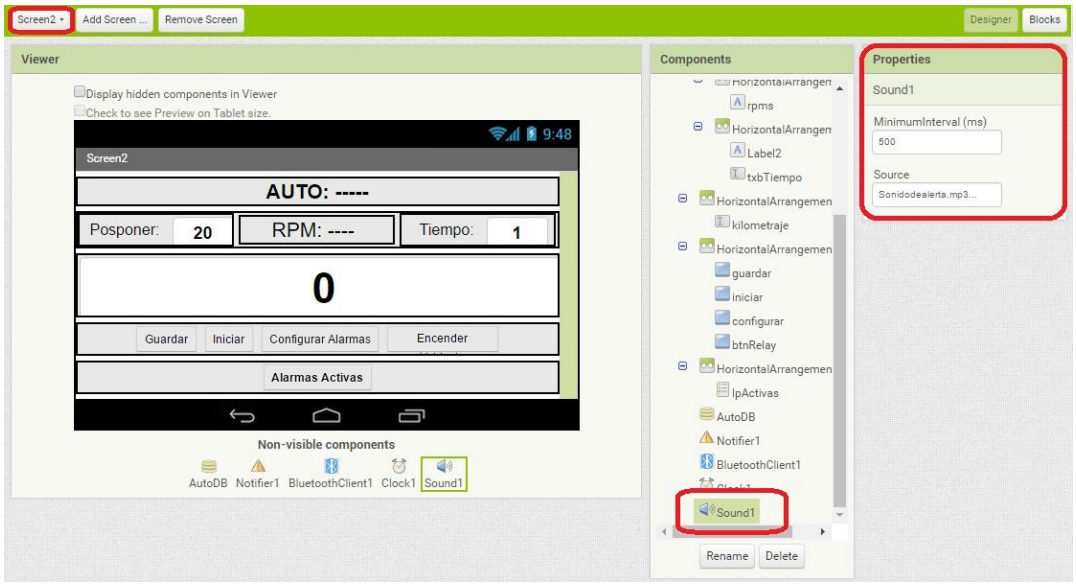


Figura 23. Carga del archivo de audio en App inventor 2.

4.6.3 GENERAR APK DE LA APLICACIÓN

Una vez realizada la programación de los tres screen se debe compilar y generar el APK del aplicativo para poder instalarlo en cualquier dispositivo Android. En la parte superior se selecciona la pestaña “Build” y se escoge la opción “App (save .apk to my computer)” como lo muestra la figura 24.

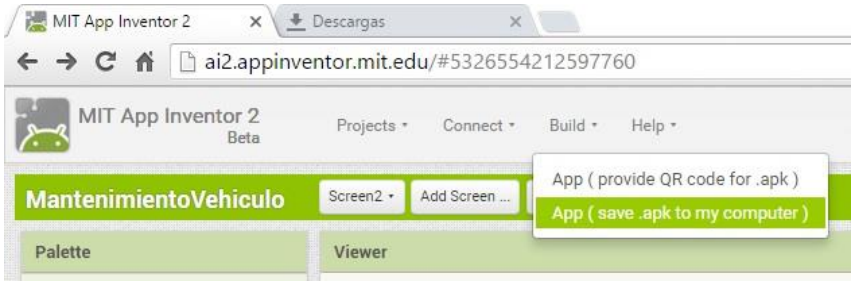


Figura 24. Creación de .APK del aplicativo.

Una vez seleccionada esta opción aparecerá una barra de carga como lo muestra la figura 25, cuando esta llegue al 100% el .APK de la aplicación se habrá guardado en la carpeta de descargas del computador.

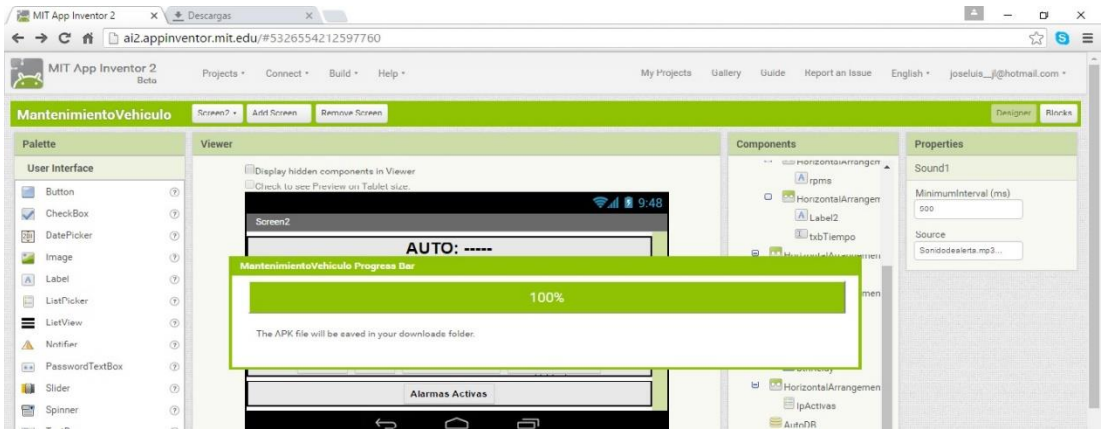


Figura 25. Carga total del archivo .APK.

En la carpeta de descargas de Windows se encontrará el archivo “MantenimientoVehiculo.apk” como lo muestra la figura 26, el mismo que se copiará en el dispositivo Android y se ejecutará para su instalación.

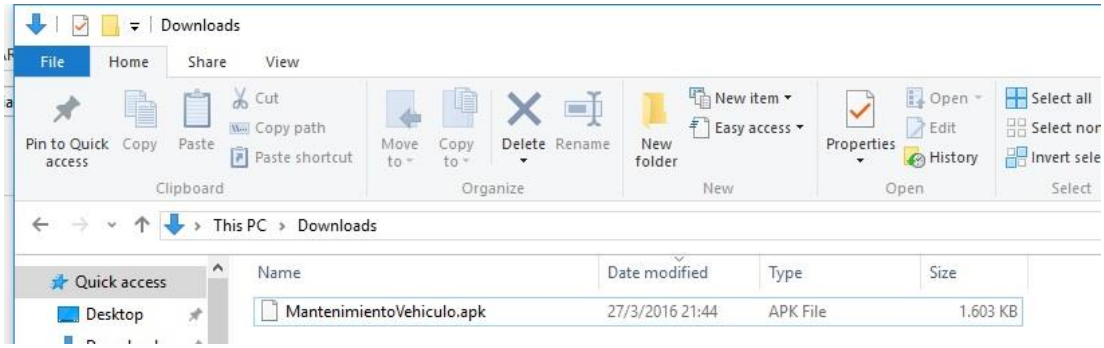


Figura 26. Archivo APK.

4.7 CREACIÓN DE PLACA DE TESIS

Para la instalación de los componentes en el vehículo se diseñará y realizará una placa independiente que será conectada a la ya implementada Arduino Leonardo la misma que tendrá los componentes para transmitir los datos de la ECU del Vehículo a la Arduino Leonardo y de la Arduino Leonardo a la aplicación en Android, esta placa también alojará los componentes electrónicos para poder encender el vehículo por medio de un pulso de 5V.

Esta placa será diseñada en Proteus, el cual es un programa que permite diseñar los circuitos que llevará la placa a sus diferentes componentes, este programa también contiene en sus librerías la mayoría de componentes electrónicos utilizados en la actualidad, y los que no tenga en sus librerías permite crearlos.

Existen dos versiones de Proteus, estas dependen del grado de complejidad de la placa a crear, la versión gratuita permite crear placas con componentes básicos como resistencias, relés, transistores, etc.

La versión profesional que debe ser adquirida bajo costo permite la creación de placas mucho más complejas con el uso de componentes más avanzados en su conexión como microprocesadores, sensores, actuadores, etc.

4.7.1 AGREGAR COMPONENTES ELECTRÓNICOS EN PROTEUS

Como paso inicial se debe agregar todos los componentes electrónicos, o por lo menos uno de cada tipo que se vaya a utilizar ya que una vez agregados se podrá copiar los mismos y cambiarlos de valor, esto para el caso de las resistencias a utilizar. Se selecciona la opción “Pick Device”, y se escoge el grupo electrónico en el cual se encuentra el componente, en esta opción se agrega:

- **Arduino base:** que será usado para emular la Arduino Leonardo, como seleccionarlo lo muestra la figura 27.

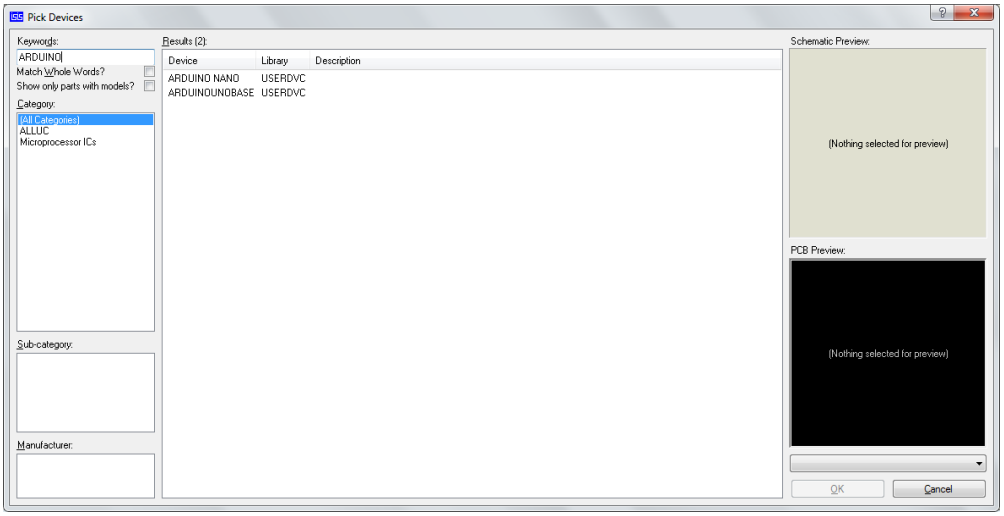


Figura 27. Selección de componente Arduino en Proteus.

- **Transistor IRFZ44N:** será el encargado de activar el relé con 5 voltios para que el vehículo pueda ser encendido, su selección se muestra en la figura 28.

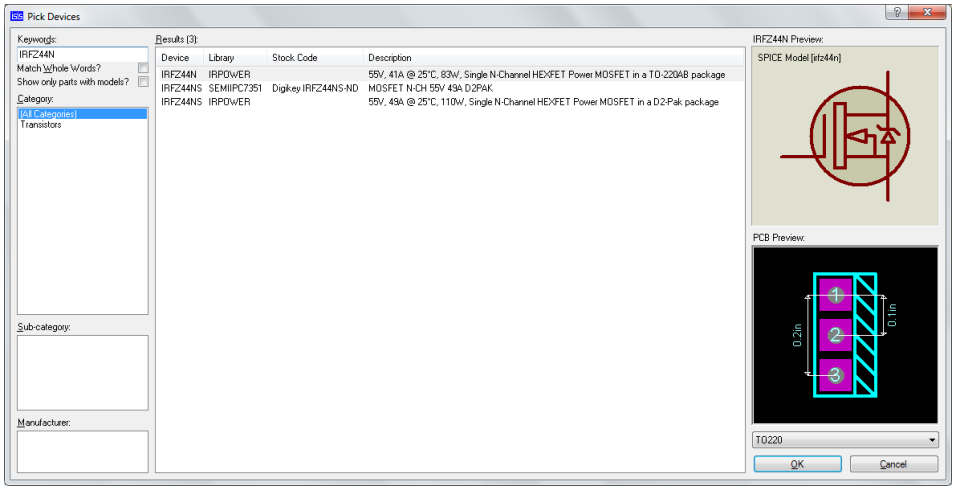


Figura 28. Selección de componente transistor IRFZ44N en Proteus.

- **Bluetooth HC05:** se usará para la transmisión de datos desde la placa Arduino Leonardo hacia la aplicación en comportamiento bidireccional, como seleccionarlo lo muestra la figura 29.

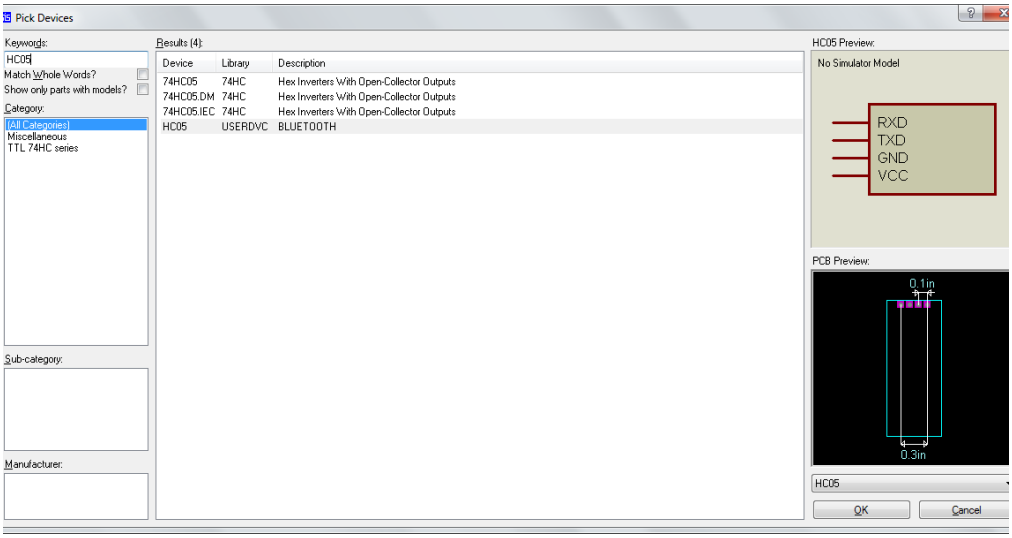


Figura 29. Selección de componente bluetooth HC05 en Proteus.

- **Relé:** Este componente se activará con 5 voltios que recibe del transistor y conectará el sistema de encendido, se debe tener en cuenta la configuración del relé ya que un relé tiene 2 pines de salida de voltaje uno normalmente abierto y uno cerrado, como seleccionarlo en Proteus lo muestra la figura 30.

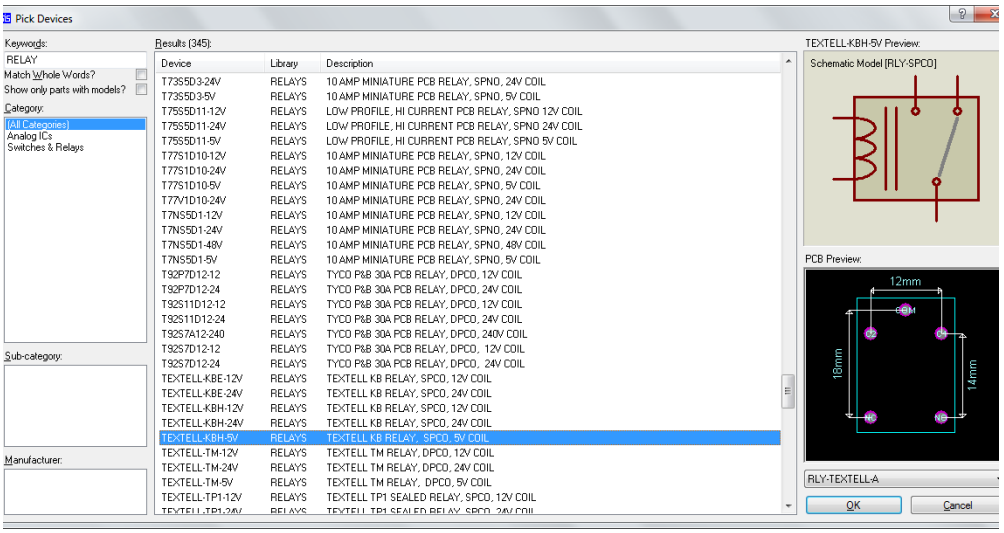


Figura 30. Selección de componente relé en Proteus.

Resistencia: sirve para controlar el voltaje y retorno que pasa por la placa Arduino Leonardo, como seleccionarlo en Proteus lo muestra la figura 31.

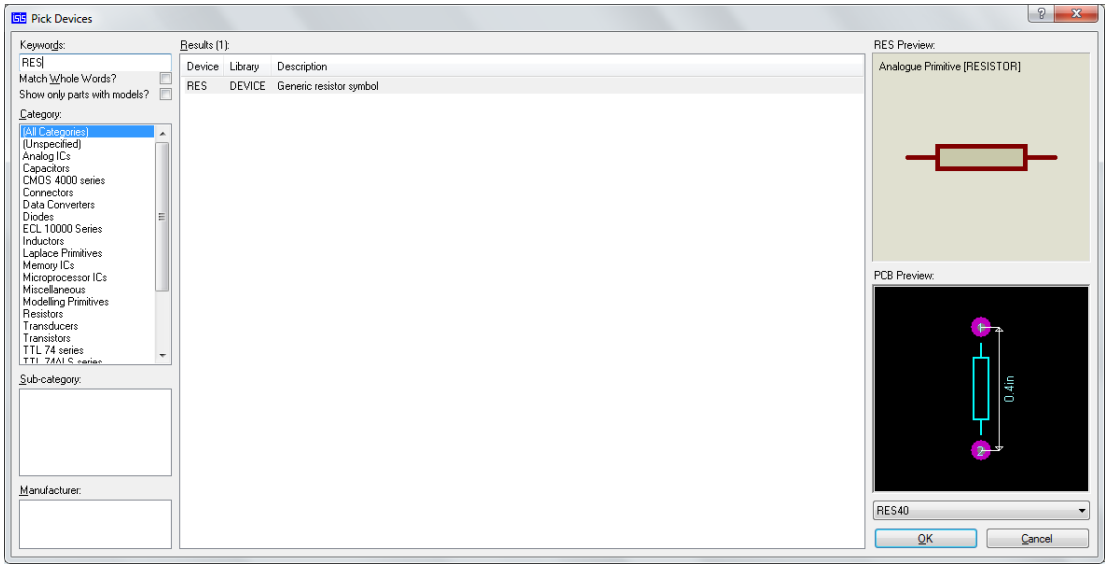


Figura 31. Selección de componente resistencia en Proteus.

- **Bornera 3 pines**: se encargará de la conexión del encendido del vehículo, aquí se recibirá 12 voltios de la batería del vehículo para poder enviarla hacia el motor de arranque, como seleccionarlo lo muestra la figura 32.

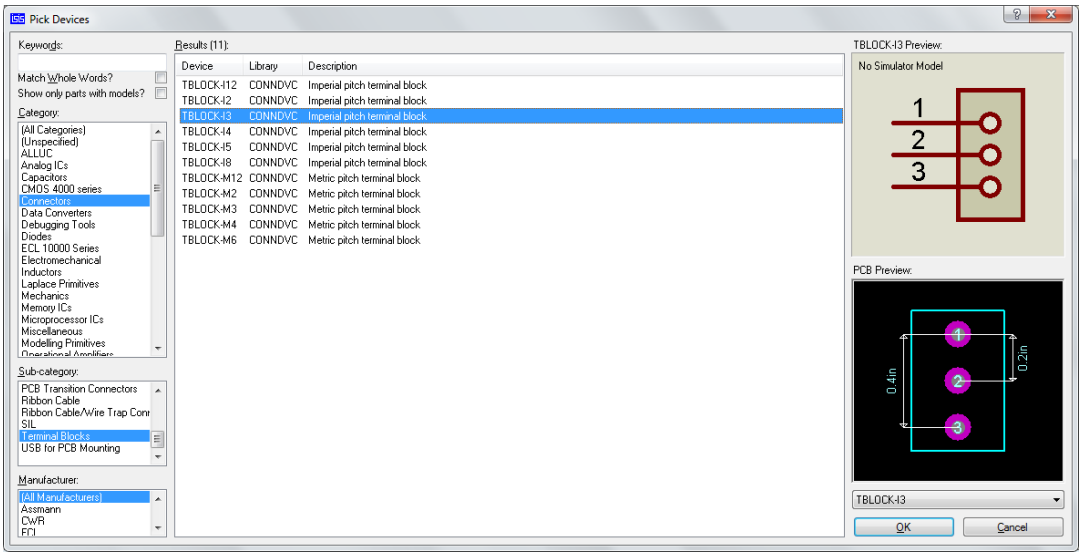


Figura 32. Selección de componente bornera 3 pines en Proteus.

- **Bornera 4 pines**: servirá para la conexión del conector ODB-II que viene desde la ECU del vehículo a la placa Arduino Leonardo, como seleccionarlo lo muestra la figura 33.

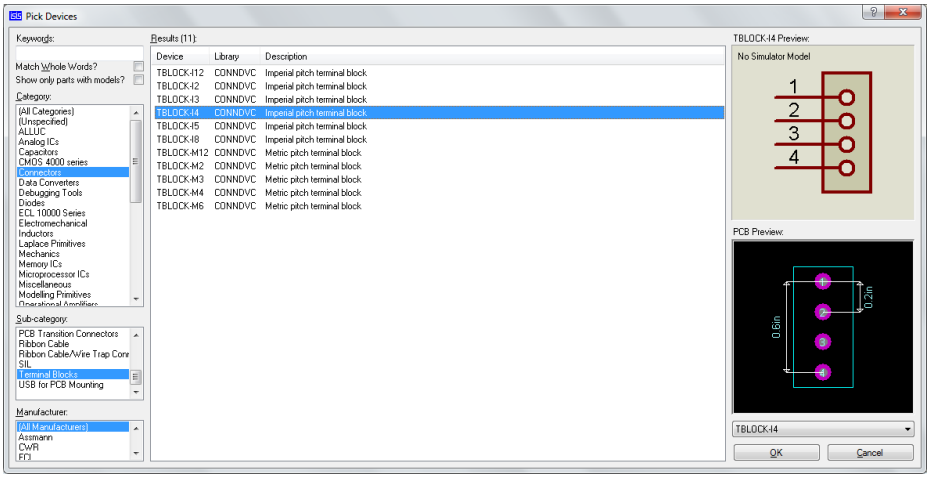


Figura 33. Selección de componente bornera 4 pines en Proteus.

4.7.2 DESIGNAR CONEXIONES Y GENERAR TRAZOS

Como siguiente paso se designa las conexiones de los pines de cada uno de los componentes como lo muestra la figura 34, es decir pines de entrada, salida e interconexiones, para que el programa Proteus reconozca la mejor ruta para la habilitación de las pistas en la placa de cobre.

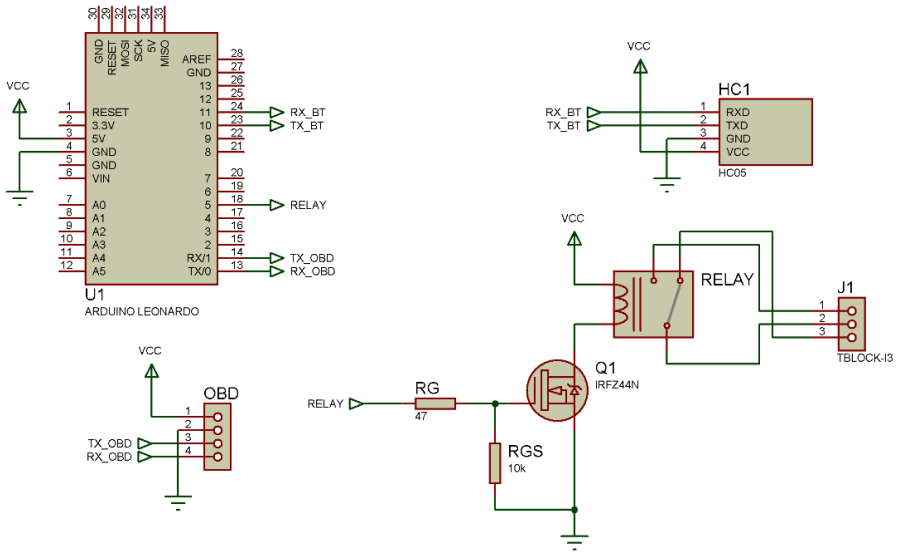


Figura 34. Designación de pines en Proteus para la interconexión de pistas.

Una vez designadas las pistas se coloca en posición los componentes, el diseño se basará en el tamaño de la placa Arduino Leonardo, el

posicionamiento de los pines deben ser exactos. El posicionamiento de los componentes es simple, se debe arrastrar con el mouse hasta la posición deseada como lo muestra la figura 35.

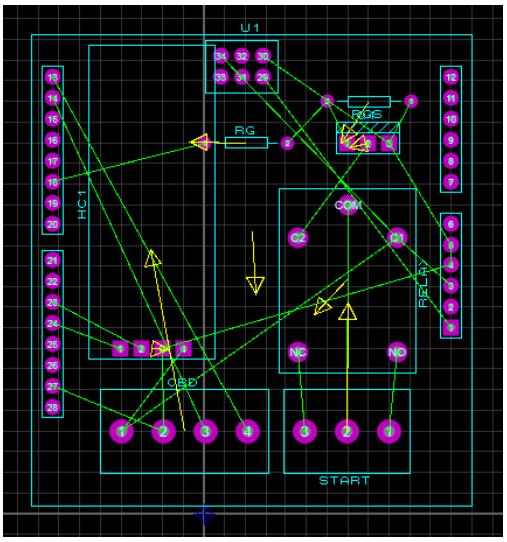


Figura 35. Diseño de la placa de componentes electrónicos en Proteus.

Luego de colocar en posición los componentes electrónicos se debe seleccionar en Proteus la opción “Design Rule Manager” para configurar el grosor de las pistas a trazar en la placa de cobre. Para una implementación más sencilla se selecciona como trazo estándar 25 puntos como lo muestra la figura 36, para las pistas que se encargarán de encender el vehículo posteriormente serán ensanchadas de forma individual.

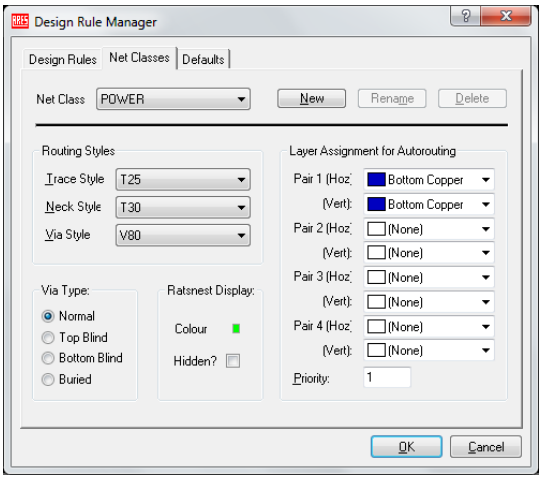


Figura 36. Configuración de las opciones de trazado en Proteus.

Una vez configurado el tipo de trazado se selecciona la herramienta “Shape Based Auto Router” la cual asistirá en el ruteo de las pistas que interconectaran los componentes en la placa. En esta ventana se debe asegurar que este la opción “Run Basic Schedule Automatically” habilitada y presionamos “Begin Routing” como lo muestra la figura 37.

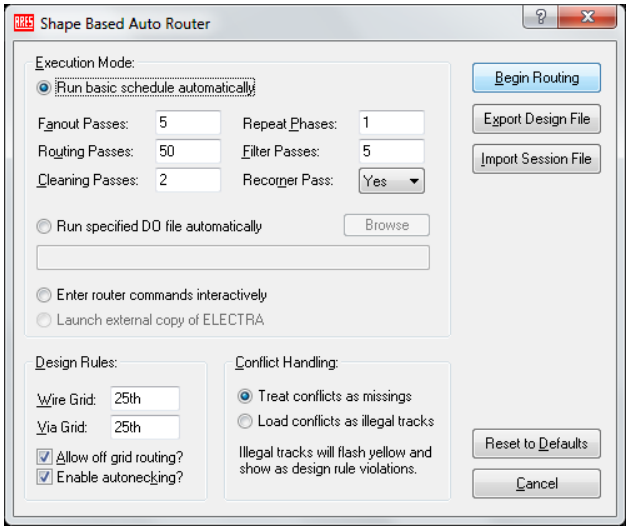


Figura 37. Inicio de trazado automático en Proteus.

Cuando ha finalizado el ruteo automático, se debe verificar que todos los componentes se hayan conectado y que no exista un error en su configuración como lo muestra la figura 38.

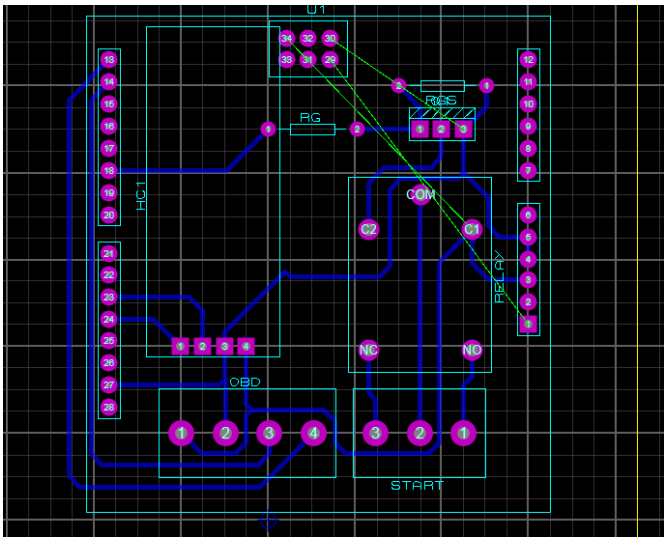


Figura 38. Placa ruteada en Proteus.

Finalizado el ruteo se debe modificar el grosor de las pistas por la cuales se conducirá el voltaje para poder encender el vehículo, 12 voltios provenientes de la batería del vehículo.

Se selecciona con click derecho la pista a modificar y se escoge la opción “Change Trace Style”, se selecciona el trazo a 60 puntos T60 como lo muestra la figura 39.

Con esto se ensanchará lo suficiente las pistas para el paso de 12 voltios que trabajarán en el encendido del vehículo.

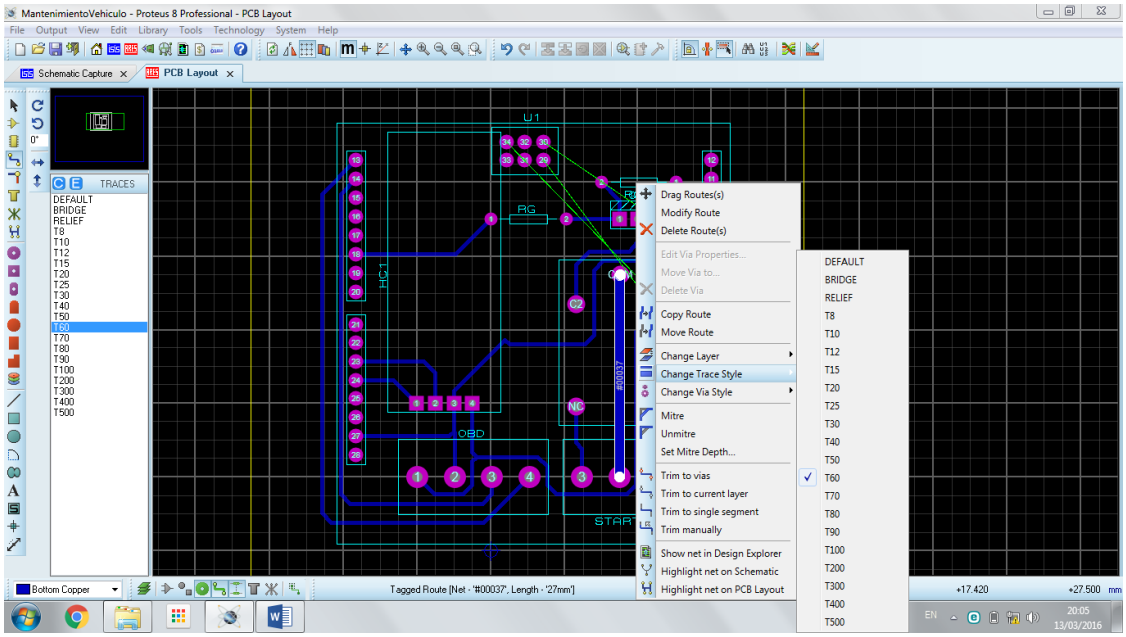


Figura 39. Modificación de trazos en Proteus.

Cuando se haya finalizado la modificación como lo muestra la figura 40 se tendrá la placa de componentes lista para ser impresa y transmitida a la placa de cobre o baquelita para su posterior quemado y preparado para la instalación final, se debe revisar minuciosamente que ninguna pista llegue a tomar contacto con otra, que las pistas no se sobrepongan haciendo así contacto, para que cuando ya se esté en funcionamiento la placa no pueda generar falsas lecturas ó en el caso de pistas con voltaje llegue a provocarse un cortocircuito.

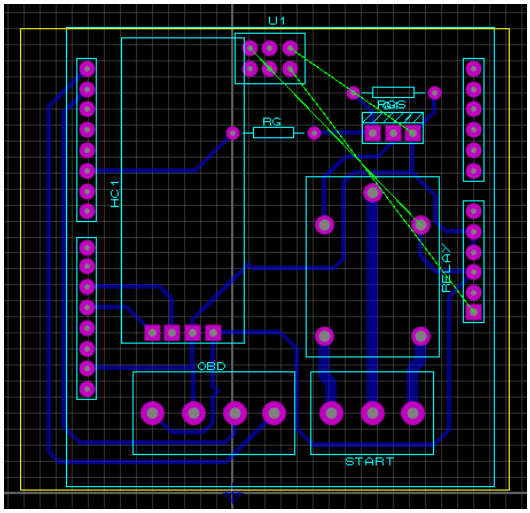


Figura 40. Diseño final de la placa.

4.7.3 IMPRESIÓN DEL DISEÑO EN LA BAQUELITA

Para la impresión de la placa se necesitará papel couche y una impresora a laser. En Proteus se selecciona la opción “Print Layaout”, la impresora y que capa se va a imprimir como lo muestra la figura 41, la capa superior e inferior deben ser impresas independientemente.

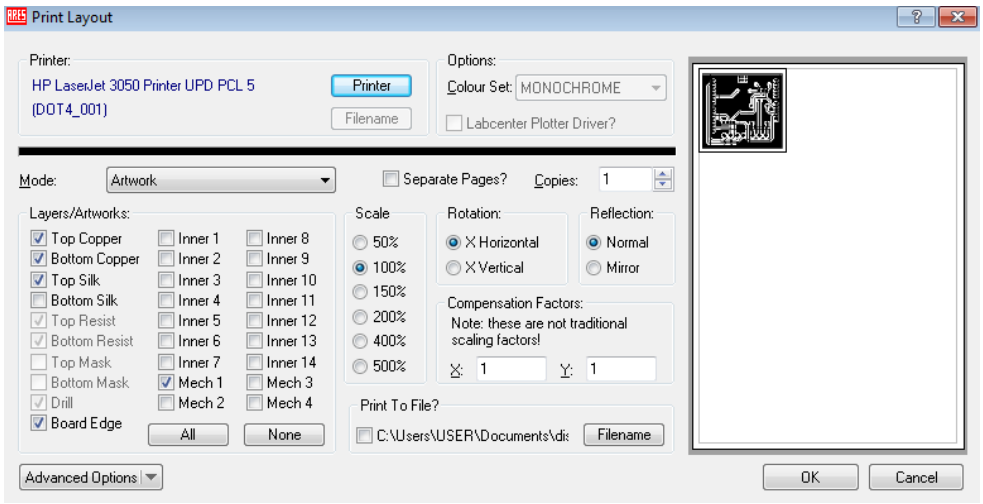


Figura 41. Ventana de impresión de capas Proteus.

Después de haber impreso individualmente el frente y el reverso se tendrán en el papel couche la cara superior y la cara inferior de la placa diseñada en Proteus como lo muestra la figura 42.

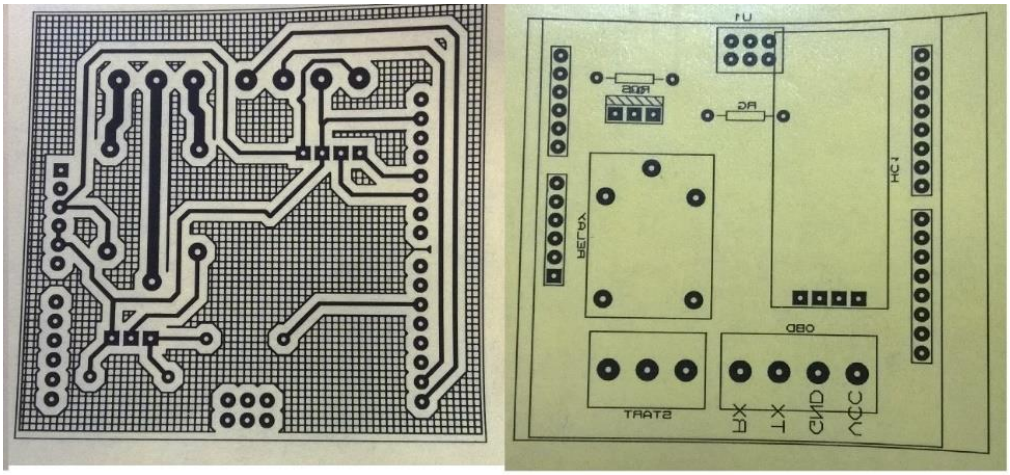


Figura 42. Impresión en papel couche del frente y reverso de la placa.

El siguiente paso es cortar la baquelita a las dimensiones del diseño, para esto se recortará el diseño impreso del papel couche y deberá ser colocado sobre la baquelita para trazar el tamaño y proceder a cortar la misma como lo muestra la figura 43.

Figura 43. Trazo y corte de la baquelita.

Una vez cortada la baquelita se debe lijar con lustre fino, extrayendo cualquier suciedad o partícula de la superficie hasta dejarla de color cobre brilloso como lo muestra la figura 44.

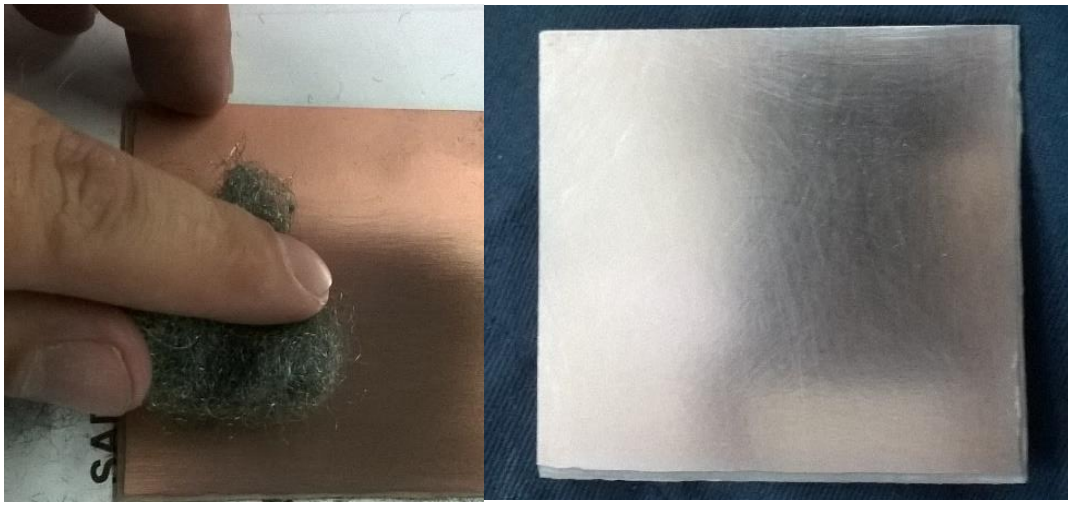


Figura 44. Limpieza de la baquelita.

Cuando la placa ya se haya tornado de color brilloso lo que se deberá hacer es colocar el diseño del frente sobre el lado de cobre y con una plancha de manera uniforme se deberá calentar de borde a borde el papel sobre la placa para que el material de la impresión a laser sea transferido del papel couche a la baquelita como lo muestra en la figura 45.

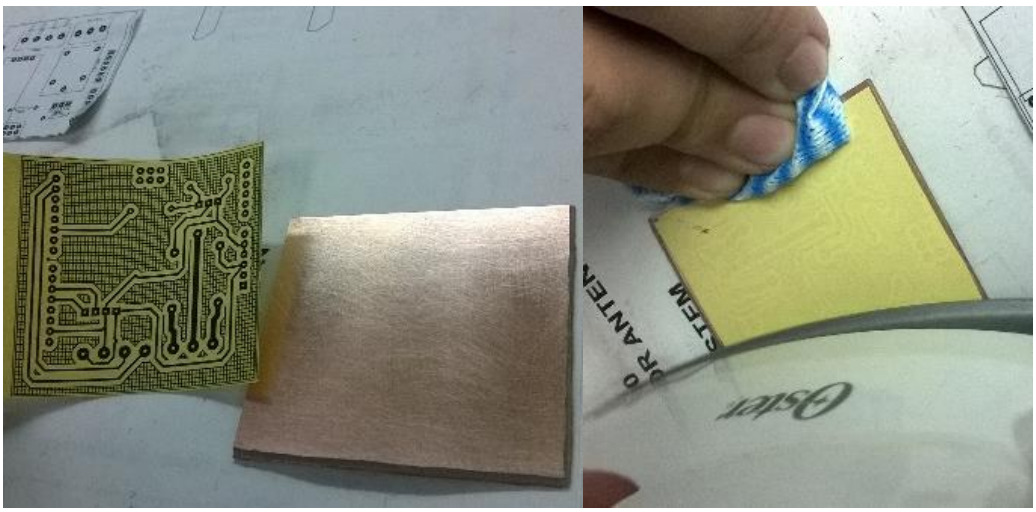


Figura 45. Transferencia del diseño a la baquelita.

Se deja enfriar por unos segundos y se procede a retirar el papel couche con cuidado, verificando que el material se haya transferido completamente como lo muestra la figura 46, caso contrario se debe volver a calentar hasta lograr la transferencia completa del material.

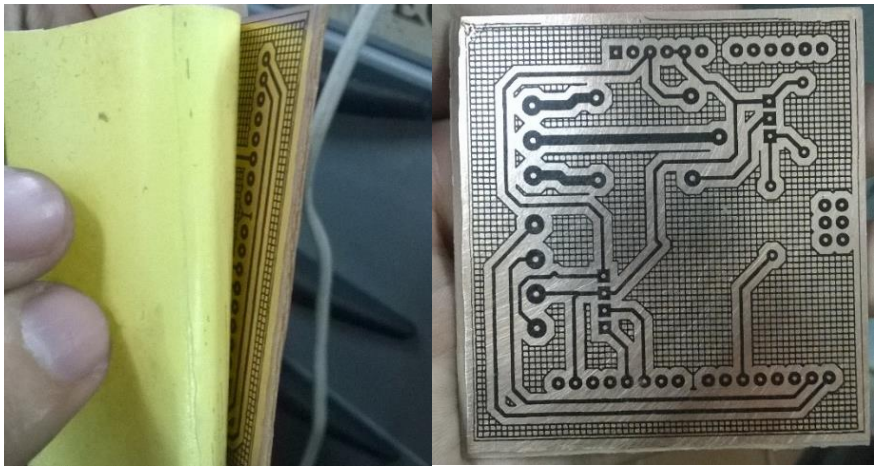


Figura 46. Diseño transferido completamente a la baquelita.

4.7.4 QUEMADO Y ARMADO DE LA PLACA DE TESIS

Como siguiente paso se quemará con ácido la placa para generar los trazos y pistas diseñadas en Proteus y así tener el diseño final. Para quemar la placa se necesita un recipiente con agua caliente y un sobre de ácido para baquelitas como lo muestra la figura 47, tener pendiente que este material debe ser manipulado con sumo cuidado.

Figura 47. Preparación del ácido para quemar la placa diseñada.

Se deberá introducir la placa dentro del ácido durante unos minutos sin dejar esta estática como lo muestra la figura 48, la placa debe estar en

movimiento constantemente para evitar que el ácido se concentre y queme pistas importantes en el diseño.

Figura 48. Placa quemada con ácido.

Cuando el ácido haya quemado completamente el diseño de la placa esta deberá ser extraída y limpiada con un paño su superficie como lo muestra la figura 49, posterior a esto deberá ser lijada con lustre la superficie obteniendo así el diseño terminado.

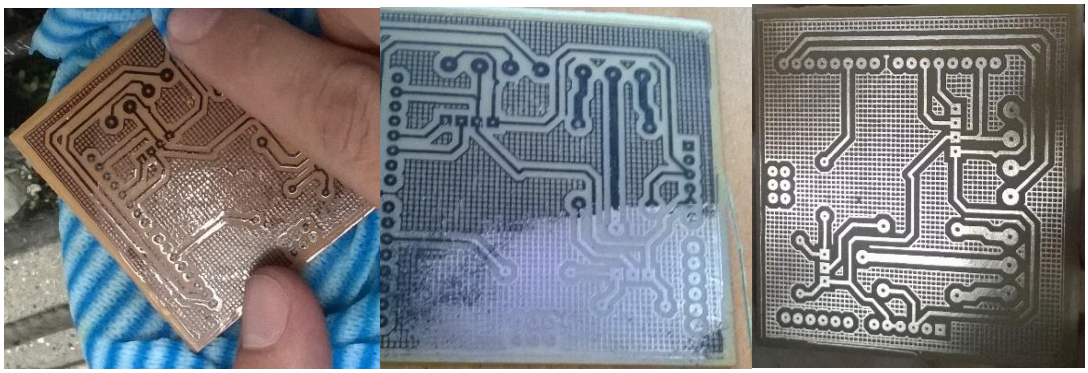


Figura 49. Placa con el diseño y quemado terminado.

Para la conexión de la placa se utilizará espadines, para poder colocar estos se debe hacer agujeros de medio milímetro en los puntos de contacto de la placa diseñada como lo muestra la figura 50. Los agujeros se los realiza con una broca de medio milímetro y un rotomatic básico.

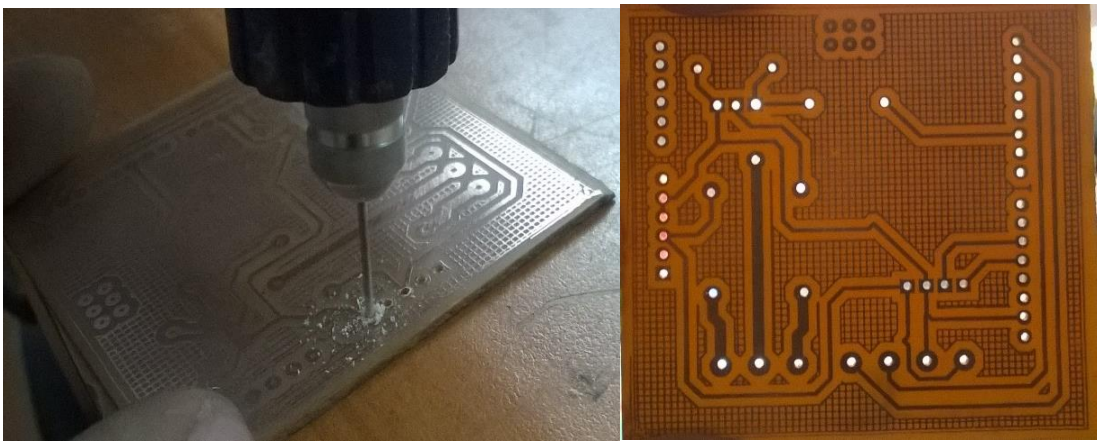


Figura 50. Agujeros en la placa terminada.

Por temas de diseño y presentación se colocará la capa del reverso impresa en papel couche sobre la baquelita, para esto se deberá repetir el procedimiento realizado anteriormente para el frente de la baquelita, la placa terminada se muestra en la figura 51.

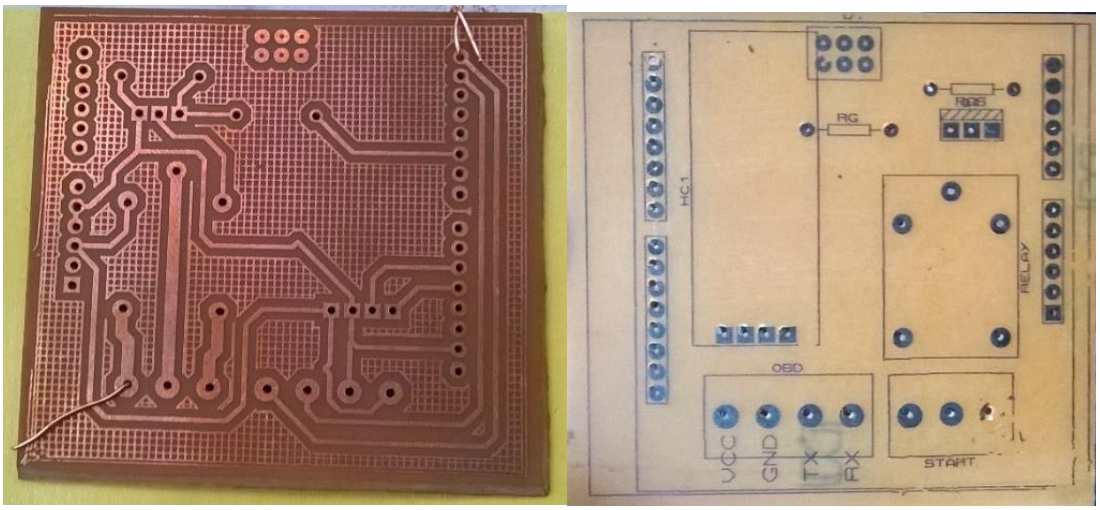


Figura 51. Transferencia de la impresión del reverso en la baquelita.

Como siguiente paso se deberá colocar todos los espadines que conectarán la placa de componentes con la placa Arduino Leonardo. Para esto se utilizará los espadines macho, estaño, pasta y cautín.

Se deberá colocar los espadines en todos los pines que tiene por defecto la placa Arduino Leonardo como lo muestra la figura 52.

Figura 52. Colocación de espadines y conexión a Leonardo.

Después de que se haya verificado que la placa de componentes ingresa sin esfuerzo en los pines de la Arduino Leonardo se procederá a colocar todos los componentes en la placa diseñada como lo muestra la figura 53.

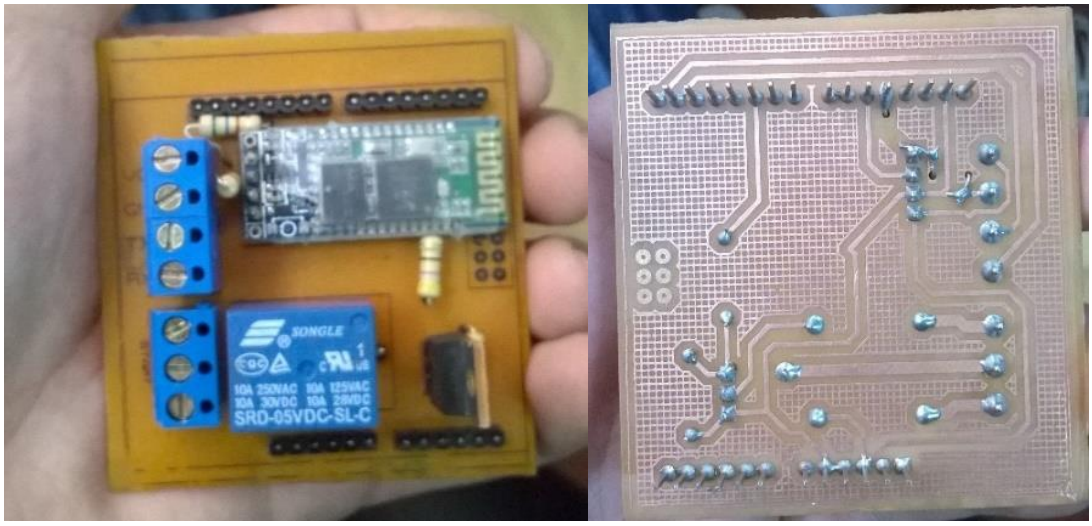


Figura 53. Colocación de componentes en la placa diseñada.

Una vez realizada la instalación de todos los componentes se procederá a enlazar el adaptador OBD-II que está conectado a la ECU del vehículo con la bornera de 4 pines en la placa de tesis como lo muestra la figura 54, se debe seguir la nomenclatura impresa en la baquelita para la conexión correcta de los 4 cables provenientes del conector OBD-II de Arduino.

La conexión a la bornera de 3 pines se realizará en la instalación del vehículo, aquí irán conectadas la entrada de voltaje y la línea de salida al arranque del vehículo.

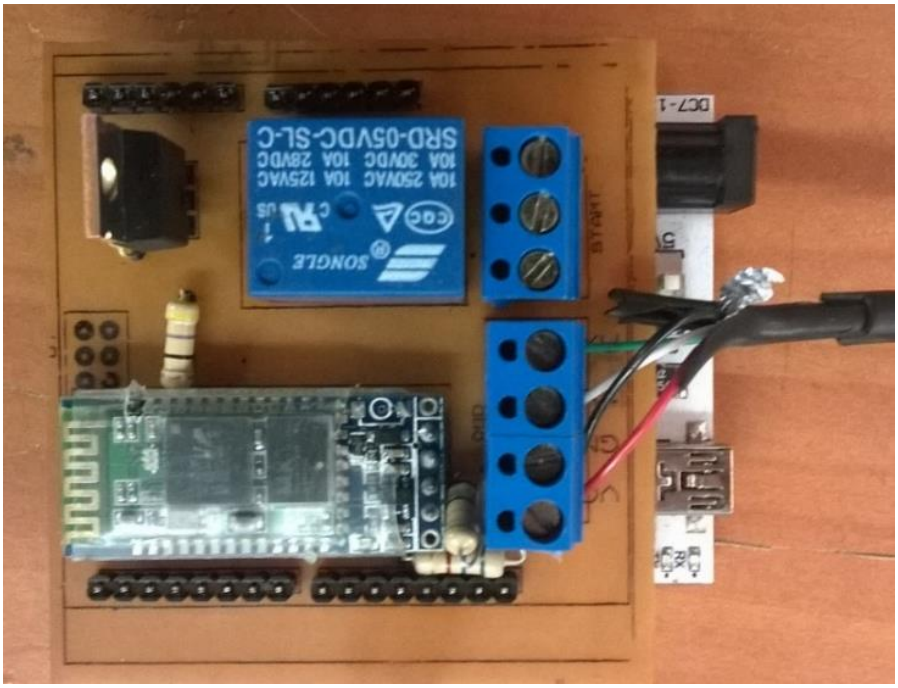


Figura 54. Conexión de adaptador OBD-II Arduino.

Con esto el diseño final y creación de la placa queda terminado para la instalación en el vehículo. El siguiente paso es colocar la placa diseñada en una caja de proyectos e instalarla en un lugar seguro en el compartimento cerca del conector OBD-II de 16 pines del vehículo.

4.8 INSTALACIÓN EN EL VEHÍCULO

4.8.1 IMPLEMENTACIÓN FÍSICA

Para la instalación en el vehículo se necesita identificar el socket de conexión del sistema de encendido y el lugar en donde se encuentre el conector OBD-II del vehículo.

El automóvil Kia Cerato Koup del año 2010 tiene el conector OBD-II de 16 pines con protocolo ISO 9141-2 en la parte inferior izquierda del panel frontal, el socket de conexión del sistema de encendido del vehículo se encuentra atrás del volante como se muestra en la figura 55, para poder encontrarlo se deberá desmontar la cubierta de este compartimento.

Figura 55. Posición de conector OBD-II y socket de encendido.

Como siguiente paso se extrae la cubierta plástica que cubre los conectores requeridos para la instalación como lo muestra la figura 56.

Figura 56. Desmontaje de cubierta plástica.

Una vez retirada la cubierta plástica se localiza el socket de conexión del sistema de encendido como lo muestra la figura 57 y se verifica la configuración de los cables, se debe conectar la placa de tesis al cable que tenga alimentación de 12 voltios una vez colocado el auto en contacto y el otro cable al retorno de arranque del socket de alimentación.

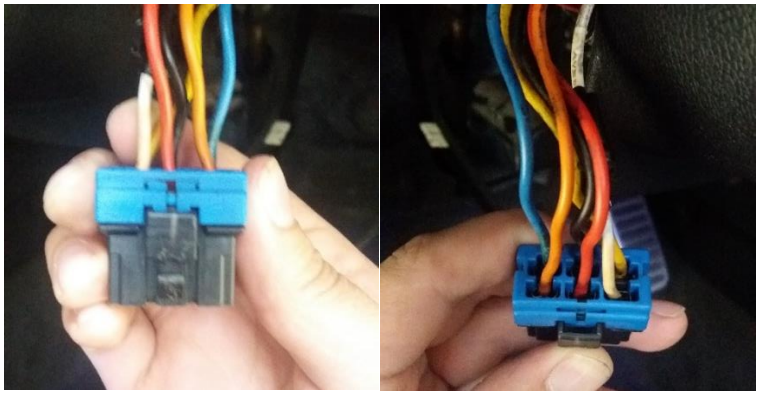


Figura 57. Socket del sistema de encendido del vehículo.

Para este caso en el vehículo Kia Cerato Koup se tiene la configuración descrita en la tabla 7.

Tabla 7. Configuración del socket de encendido del vehículo Kia Koup.

Tipo	Color	Descripción
Electricidad	Azul	Alimentación 12voltios cuando el auto está en contacto.
Electricidad	Blanco	Conducción de energía hacia el arranque del vehículo.

Se debe cortar el cable blanco como lo muestra la figura 58 para que al accionar el arranque con el switch el auto no realice acción alguna.

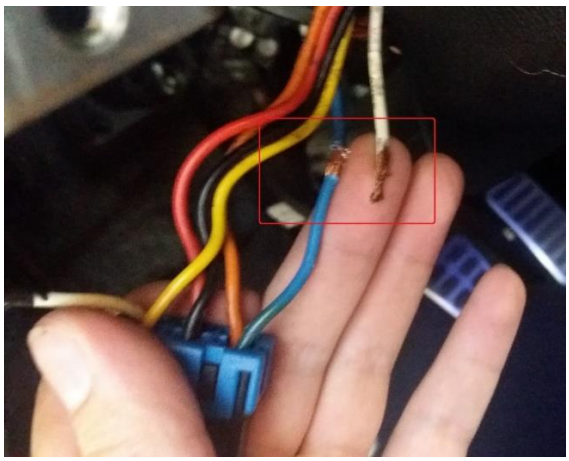


Figura 58. Socket del sistema de encendido preparación para el puenteo.

Este cable deberá ser conectado a la bornera de la placa de tesis que está creada con conexión de pistas hacía el relé, el otro cable de la bornera que va a la alimentación del relé deberá ser conectado al cable azul que es de alimentación 12 voltios cuando el vehículo esté en contacto como lo muestra la figura 59.

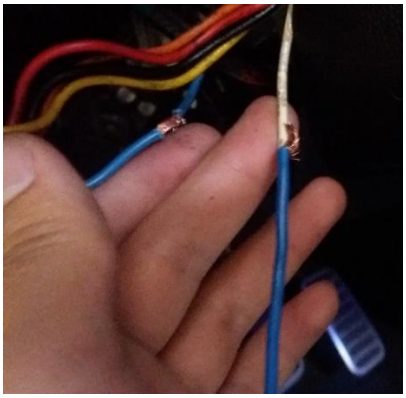


Figura 59. Conexión entre placa de tesis y socket del sistema de encendido.

El otro extremo del cable deberá ser conectado en la placa de tesis a la bornera del relé, y en paralelo deberá ser conectado un pulsador de emergencia como lo muestra la figura 60 para poder encender el vehículo sin necesidad de la aplicación en Android.

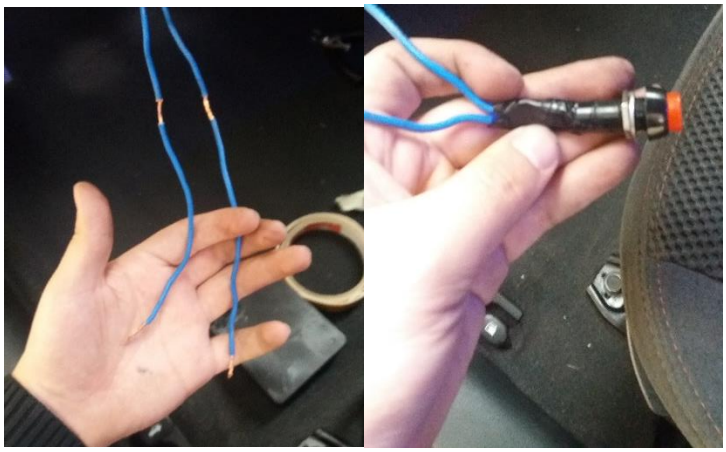


Figura 60. Conexión del botón de encendido de emergencia.

Este botón deberá ser usado solo en caso de emergencia ya que el procedimiento correcto es la utilización del encendido de la aplicación.

Una vez puenteado el socket del encendido debe ser conectado otra vez al vehículo, como siguiente paso se conectará el adaptador OBD-II en el socket que viene en el automóvil para la comunicación con la ECU como lo muestra la figura 61.

Figura 61. Conexión del Adaptador OBD-II Arduino al socket del vehículo.

Una vez realizadas las conexiones se deberá cerrar la caja de proyectos como lo muestra la figura 62 y se tendrá que empotrar en el vehículo.

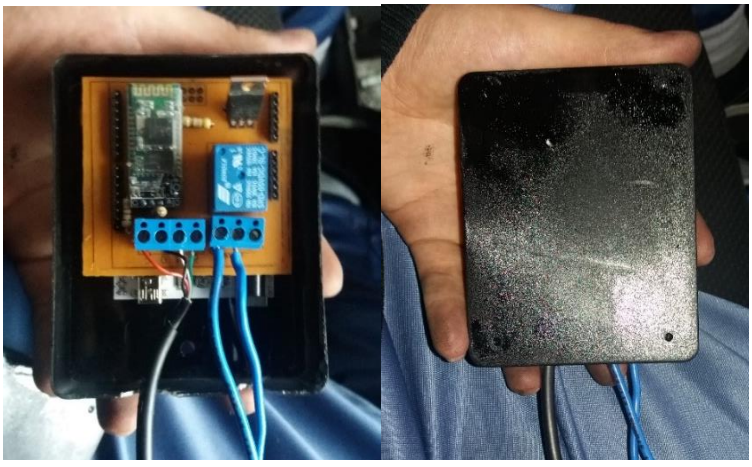


Figura 62. Caja de proyectos conectada.

Se debe verificar que los cables no se interpongan en el paso de vinchas o tornillos de fijación de la cubierta de plástico como lo muestra la figura 63.

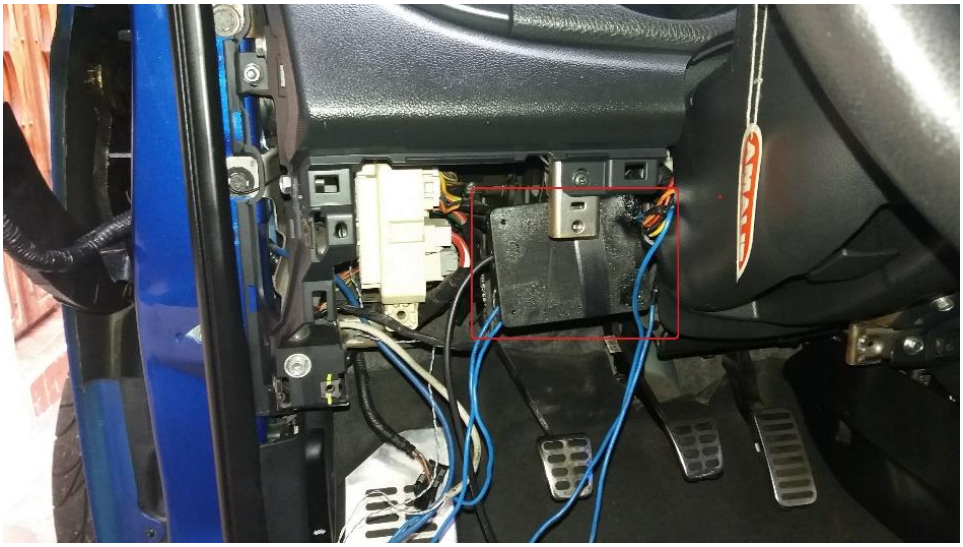


Figura 63. Fijación de la caja de proyectos en el vehículo.

Después de haber fijado la caja de proyectos y verificado la colocación de los cables de conexión se debe posicionar el pulsador en un lugar seguro y se debe volver a colocar la cubierta de plástico, como lo muestra la figura 64.

Figura 64. Armado de la sección de instalación.

Se debe realizar pruebas de encendido colocando el auto en contacto y presionando el botón de emergencia. El siguiente paso será la configuración del aplicativo para el vehículo en el cual fue instalado.

4.8.2 INICIO Y CONFIGURACIÓN DE LA APLICACIÓN.

Para poder iniciar el uso de la aplicación esta debe ser instalada en la tablet, se debe copiar y ejecutar el APK generado en la programación desde la tablet para así instalarla como lo muestra la figura 65.

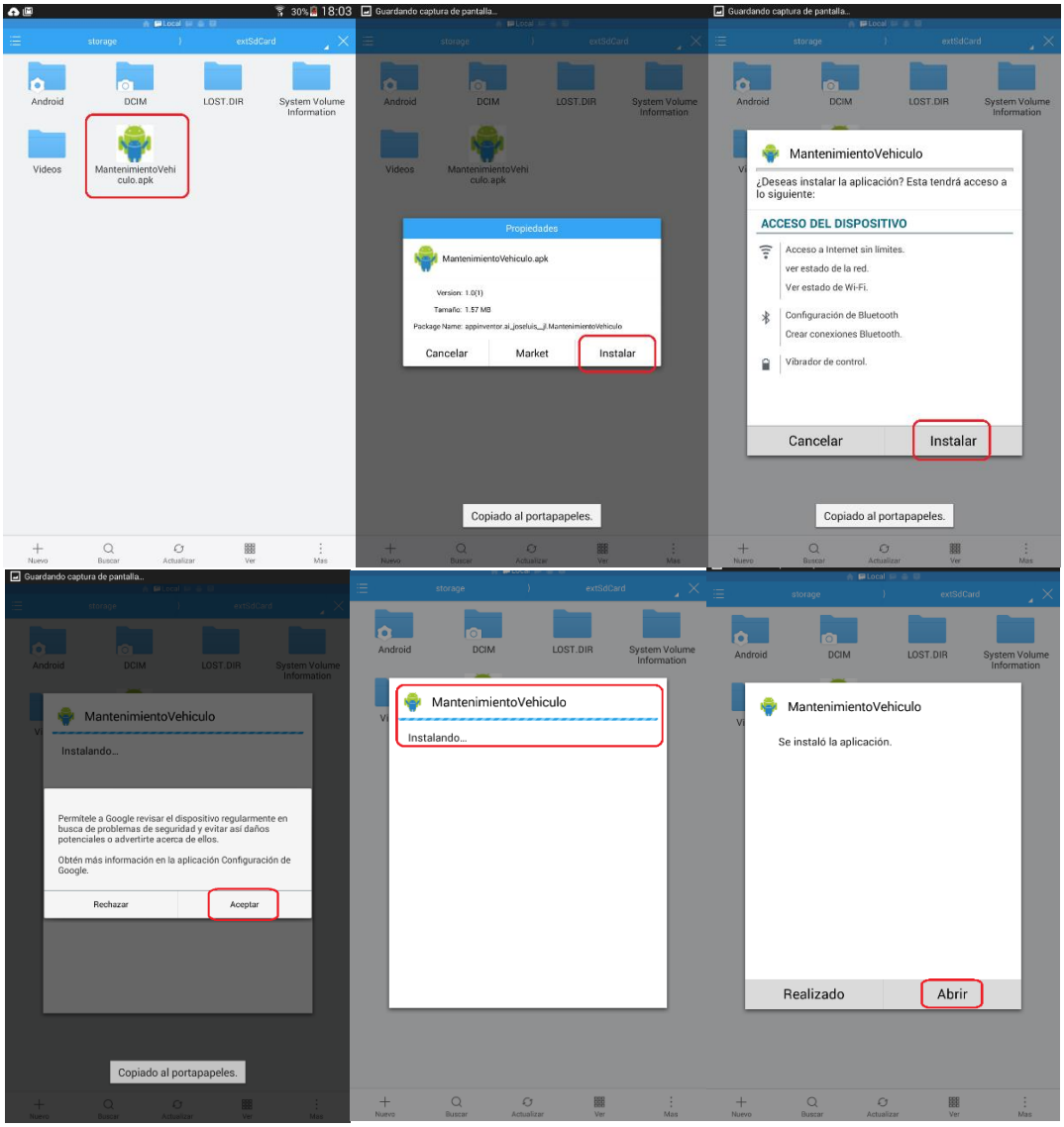


Figura 65. Instalación del APK en la tablet.

Al finalizar el proceso de instalación se debe abrir el aplicativo y se debe presionar la opción “seleccionar vehículo”, lo que llevará a una lista de autos o dispositivos disponibles para conectarse mediante bluetooth, de esta lista se debe seleccionar el auto designado para las pruebas, en este caso un Kia

Cerato, el aplicativo pasará a la siguiente ventana en donde trabajará principalmente, cuando el vehículo seleccionado es nuevo en el entorno de la aplicación se generará un cuadro de aviso indicando que se ha agregado a la base de datos un nuevo vehículo con éxito como lo muestra la figura 66.

Figura 66. Selección del vehículo en el aplicativo.

Como siguiente paso se debe ingresar las alarmas que se estipulen en el plan de mantenimiento del manual de usuario del vehículo, inicialmente solo se tendrá la opción “Nueva” como lo muestra la figura 67.

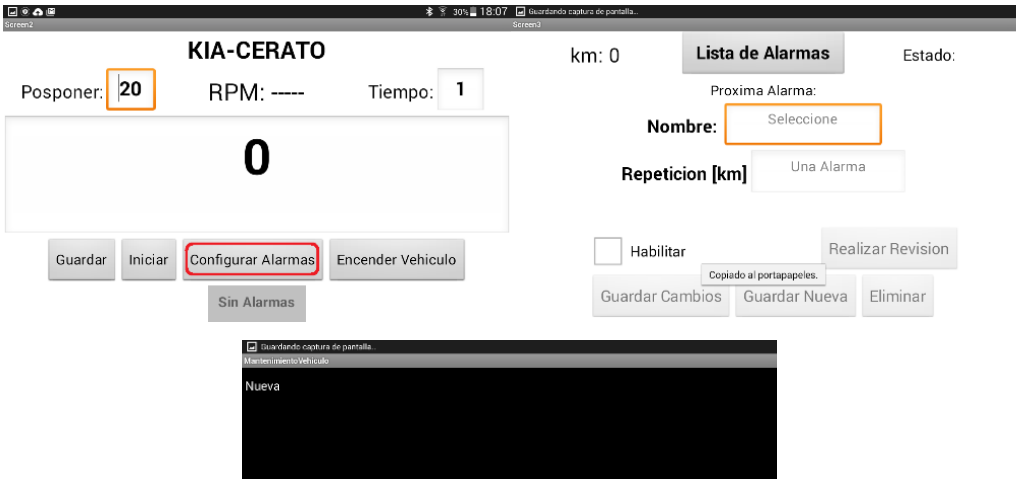


Figura 67. Configurar alarmas en aplicativo.

Para las pruebas se registrarán 3 alarmas de mantenimiento, las mismas que estarán configuradas con intervalos de 1,2 y 3 kilómetros respectivamente. Se las nombrará como “Prueba 01”, “Prueba 02” y “Prueba 03”.

Con esto al pasar 1,2 y 3 kilómetros las alarmas deberán activarse una a una. Seleccionamos del menú la opción nueva e ingresamos la alarma “Prueba 01” con intervalo de 1 kilómetro.

Repetimos el proceso para las alarmas “Prueba 02” y “Prueba 03” como lo muestra la figura 68.

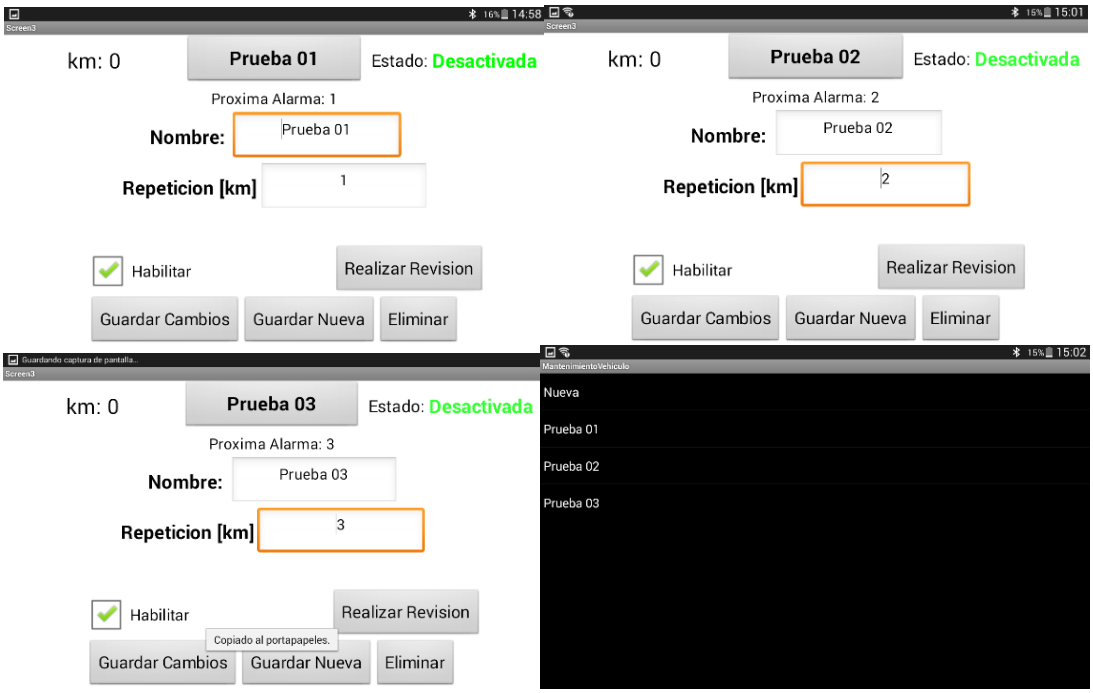


Figura 68. Creación de las alarmas de prueba.

Al finalizar la creación de las alarmas se necesitará configurar el intervalo para posponer una alarma y cada que tiempo se va a actualizar la información de la aplicación.

Esto deberá ser configurado en la pantalla principal de la aplicación en la sección superior como lo muestra la figura 69.

Figura 69. Configuración de tiempo y posponer.

Para vehículos nuevos no habrá necesidad de realizar este paso, para vehículos que ya han recorrido una distancia sin la aplicación conectada se deberá actualizar el dato del odómetro como lo muestra la figura 70, en este caso el Kia Koup en el cual se está implementando el sistema tiene 102505 kilómetros.

Figura 70. Configuración del odómetro de la aplicación.

Al iniciar la aplicación se está indicando el inicio de la transmisión de datos, como se acaba de actualizar el odómetro las tres alertas de mantenimiento; “Prueba 01”, “Prueba 02” y “Prueba 03” deberán activarse automáticamente como lo muestra la figura 71. Y se deberá desplegar las alertas visuales y sonoras en la pantalla principal.

Figura 71. Screen de alarmas activas.

Luego de esto se debe detener la transmisión de datos y realizar la revisión de las alarmas activas de mantenimiento en el aplicativo como lo muestra la figura 72.

Para verificar cuales son las alarmas activas se debe seleccionar el botón resaltado con rojo de “Alarmas Activas” el cual dirigirá el sistema a una pantalla con la lista de alarmas activas a las cuales se deberá realizar el proceso de mantenimiento.

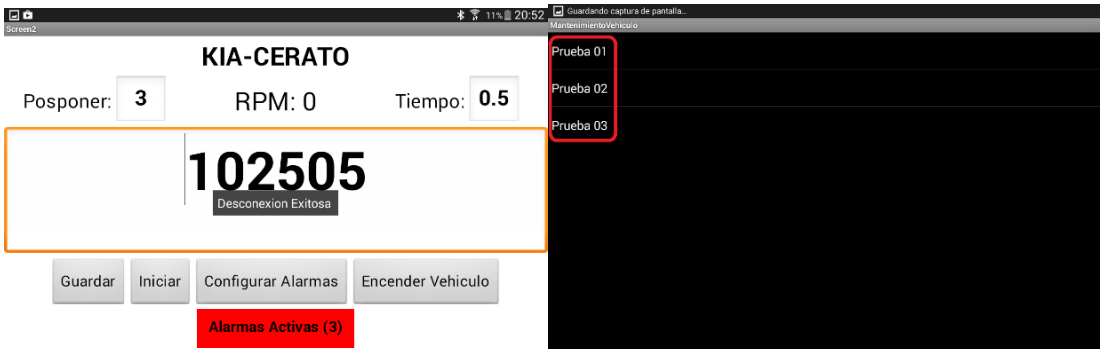


Figura 72. Verificación de alarmas activas.

Una vez verificada la lista de alarmas activas se debe presionar el botón “Realiza Revisión”, con esto el intervalo de las alarmas de mantenimiento se cambiará 1,2 y 3 kilómetros respectivamente como lo muestra la figura 73.

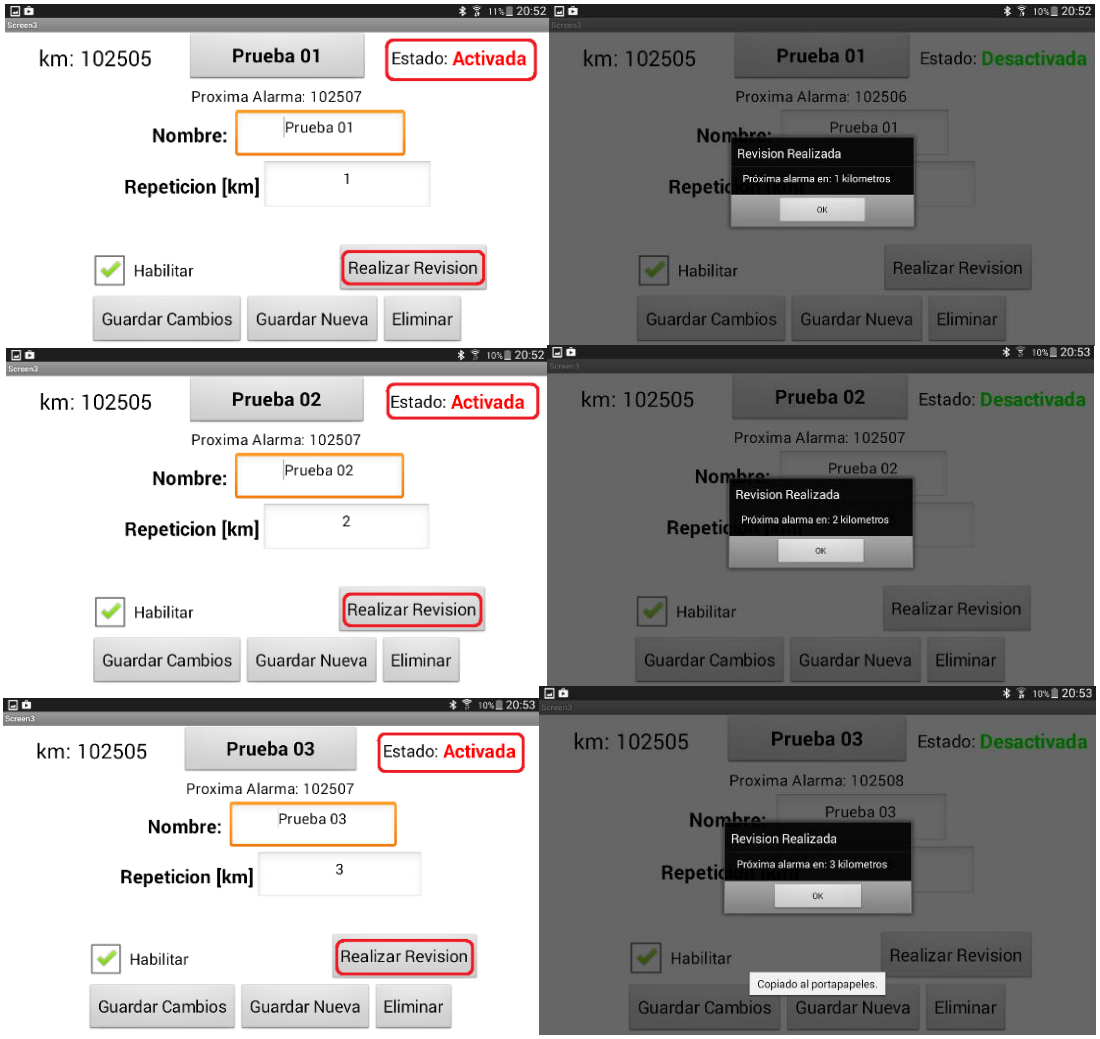


Figura 73. Restablecimiento de alarmas.

Con esto termina la configuración del aplicativo, ya tiene 3 alarmas listas para ser activadas conforme el kilometraje del vehículo aumenta. Al activarse una alarma se visualizará en la pantalla principal y se generará una alarma sonora.

4.9 PRUEBAS Y RESULTADOS

Para las pruebas se configuró la aplicación con 3 alarmas denominadas “Prueba 01”, “Prueba 02” y “Prueba 03”. Los intervalos de activación de estas alarmas son 1, 2 y 3 kilómetros respectivamente. Como resultado de las pruebas realizadas se obtuvo:

4.9.1 PRUEBA A. CONFIGURACIÓN DEL ODÓMETRO DE LA APLICACIÓN.

Las pruebas fueron satisfactorias se logró configurar el odómetro de la aplicación sin problemas como lo muestra la figura 74.

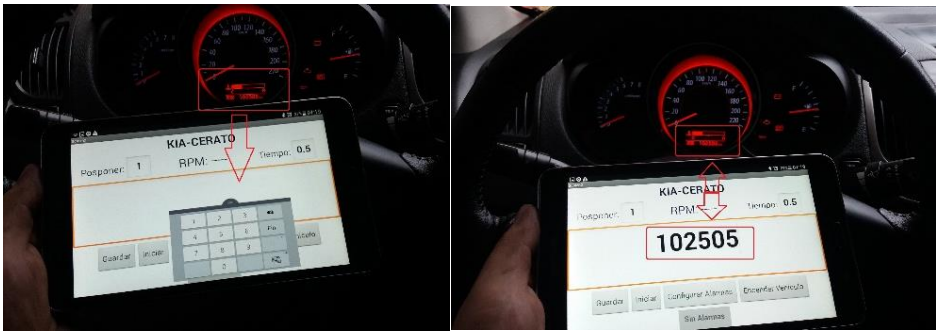


Figura 74. Prueba A configuración del odómetro.

4.9.2 PRUEBA B. CONEXIÓN Y DESCONEXIÓN CON EL VEHÍCULO

Las pruebas fueron satisfactorias se logró realizar la conexión y desconexión del aplicativo con el vehículo sin errores como lo muestra la figura 75

Figura 75. Conexión y desconexión del aplicativo.

4.9.3 PRUEBA C. VERIFICACIÓN DEL BOTÓN DE ENCENDIDO.

Las pruebas fueron satisfactorias se logró encender el vehículo desde la aplicación y se verificó la habilitación y deshabilitación del mismo por medio del valor de las RPM como lo muestra la figura 76.

Figura 76. Prueba C botón de encendido.

4.9.4 PRUEBA D. VERIFICACIÓN DE ACTIVACIÓN DE ALERTAS

Las pruebas fueron satisfactorias se configuró 3 alarmas las cuales se debían activar en intervalos de 1, 2 y 3 kilómetros, el kilometraje inicial de las pruebas fue 102505 kilómetros. Por lo que las alarmas debían activarse a: 102506; 102507; 102508 respectivamente, todas las alarmas se activaron como lo muestra la figura 77.

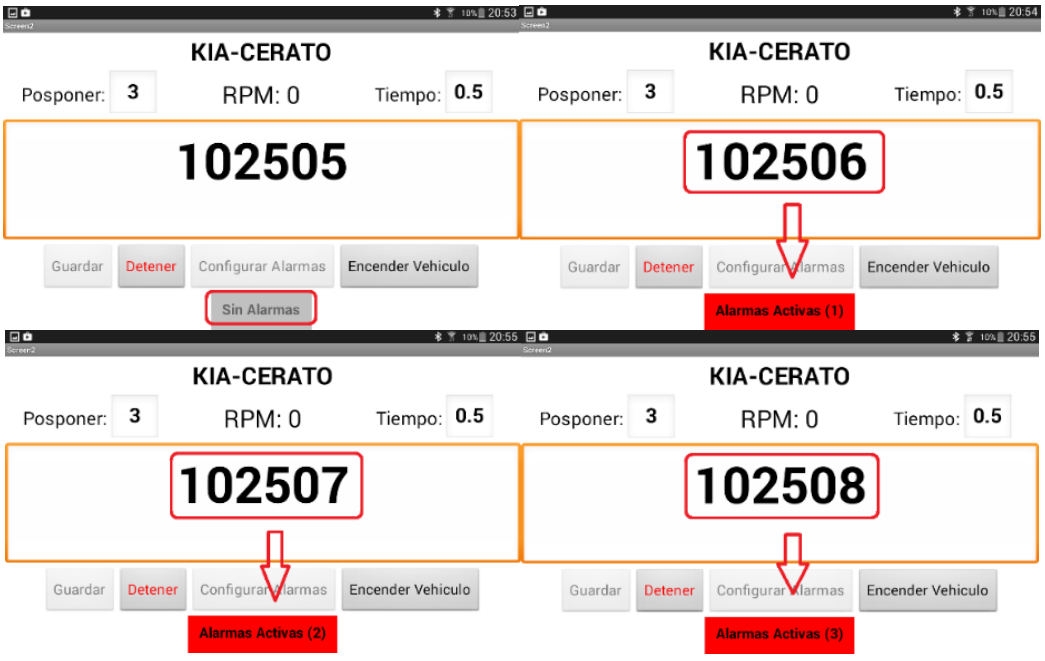


Figura 77. Activación de alarmas.

Con las pruebas realizadas se puede corroborar el correcto funcionamiento de la aplicación y con esto se cargará el plan de mantenimiento del vehículo con los intervalos reales de revisión.

CONCLUSIONES Y RECOMENDACIONES

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- El sistema interactivo propuesto logra obtener cada medio segundo información de la ECU del vehículo para así mantener alertado al conductor sobre los periodos correctos para el mantenimiento y revisión de partes del vehículo.

- La transmisión de datos por medio de protocolo OBD-II permite obtener información de la ECU en tiempo real mediante PID’s de comunicación designados por la normativa.

- Cada vehículo tiene su plan de mantenimiento estipulado en el manual de usuario, los intervalos de mantenimiento cambian dependiendo el material y calidad de sus componentes.

- El diseño de la aplicación es amigable con el conductor del vehículo, permitiendo así la puesta en funcionamiento del aplicativo teniendo como beneficio un vehículo con mantenimientos adecuados y mayor confiabilidad para su uso diario.

- El proceso de instalación en el vehículo se debe realizar tomando en cuenta las conexiones eléctricas ya que una conexión incorrecta puede provocar el daño en la placa de tesis y demás componentes del Arduino.

- Con las pruebas realizadas se logró determinar el correcto funcionamiento del aplicativo, teniendo así gran impacto positivo en la vida útil de los vehículos al dar un mantenimiento periódico adecuado.

5.2 RECOMENDACIONES

- El uso del aplicativo desarrollado para el aviso de los intervalos correctos de mantenimiento es indispensable para un conductor inexperto en el área de la mecánica automotriz

- Se debe tener en cuenta el protocolo OBD-II que usa el vehículo ya que existen varios, y para una correcta transmisión de datos se debe asegurar que el tipo de protocolo sea compatible con los componentes usados.

- Para la instalación en el vehículo se deben seguir las recomendaciones de conexión ya que realizar mal este procedimiento podría llegar a dañas la placa de tesis y demás componentes de Arduino.

- Es necesario realizar mantenimiento preventivo en el vehículo para así reducir el desgaste prematuro de sus partes, aumentar la seguridad y confiabilidad del vehículo.

- El aplicativo puede ser instalado en cualquier dispositivo con sistema operativo Android se recomienda dar todos los permisos de transmisión bluetooth para así no tener interferencia en la transmisión de datos.

- Para las pruebas se recomienda calcular intervalos de creación y distancia a posponer para que no generen confusión al realizar las verificaciones visuales y auditivas de una alarma logrando así identificar adecuadamente cada alarma configurada.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Aficionados a la Mecánica. (12 de 12 de 2015). Recuperado el 20 de Diciembre de 2015, de http://www.aficionadosalamecanica.com/obd2.htm

Águeda, E., García , J., Navarro, J., & Gómez, T. (2009). *Fundamentos Tecnológicos del Automóvil (1° Ed.).* Madrid, España: THOMSON Editores Paraninfo S.A.

Android. (11 de 11 de 2015). Recuperado el 13 de Febrero del 2016, de Android: http://www.xatakandroid.com/sistema-operativo/que-es-android

Android. (09 de 09 de 2015). Recuperado el 13 de Febrero del 2016, de Android: http://www.mundomanuales.com/telefonia/telefonos-moviles/que-es-android-caracteristicas-y-aplicaciones-4110.html

App Inventor. (12 de 11 de 2015). Recuperado el 13 de Febrero del 2016, de App Inventor: https://googleblog.blogspot.com/app-inventor-for-android.html

App Inventor. (12 de 11 de 2015). Recuperado el 13 de Febrero del 2016, de App Inventor: /http://appinventor.googlelabs.com/about/

Arduino. (28 de 01 de 2016). Recuperado el 29 de Enero del 2016, de Arduino: http://www.Arduino.cc/

Arduinodev. (04 de 02 de 2016). Recuperado el 28 de Febrero de 2016, de Arduino: http://arduinodev.com/hardware/obd-kit/

ArduinoLeonardo. (20 de 12 de 2015). Recuperado el 07 de Enero de 2016, de Arduino: http://www.electronicaembajadores.com/Productos/modulo-arduino-leonardo

Astudillo, M. (2010). *Tecnología del Automóvil (1° Ed.).* Madrid, España: Ediciones Paraninfo, SA.

Autobox. (10 de 10 de 2015). Recuperado el 12 de Diciembre de 2015, de Autobox: http://www.autobox.biz/page_id14

Bluetooth. (11 de 12 de 2015). Recuperado el 24 de Enero de 2016, de Bluetooth: http://es.ccm.net/faq/10973-la-tecnologia-bluetooth

Bluetooth. (10 de 09 de 2015). Recuperado el 24 de Enero de 2016, de Bluetooth: http://mcuoneclipse.com/2013/06/19/using-the-hc-06-bluetooth-module/

Bluetooth. (04 de 10 de 2015). Recuperado el 24 de Enero de 2016, de Bluetooth: http://www.neoteo.com/comandos-at-modulo-bluetooth-hc-06/

Bosch, R. (2005). *Manual de la técnica del automóvil.* Alemania.

Bosch GmbH, R. (2005). Manual de la Técnica del Automóvil (4° Ed.). Alemania: Bosch.

Boxall, J. (2013). *Arduino Workshop.* San Francisco, USA: Library of congress cataloging.

Lucas, B. (2007). *Seguridad en el mantenimiento de vehículos.* España: Paraninfo.

Mantenimiento del vehículo. (02 de 09 de 2015). Recuperado el 11 de Noviembre de 2015, de Mantenimiento del vehículo: http://www.sura.com/blogs/autos/mantenimiento-preventivo-vehiculo.aspx

MecánicaMotor. (2012). *Mecánica y Motores*. Recuperado el 23 de Noviembre de 2015, de www.mecanicaymotores.com/mantenimientoprev.html

NAPA Institute of Automotive Technology*. Introduction to OBDII*. Estados Unidos de Norteamérica, Recuperado 25 de Noviembre de 2015. Disponible en: http:lbcc.edu/attcdocumentsBD.pdf

OBDII Experts. *OBDII Connector & Protocol by manufacturer*.

Estados Unidos de Norteamérica. Recuperado 10 de Noviembre de 2015 Disponible en: http://www.obdexperts.co.uk/faq.html

OBDII. (03 de 08 de 2015). Recuperado el 15 de Noviembre de 2015, de OBDII: http://www.teseomotor.com/ubicacion-conector-obd/

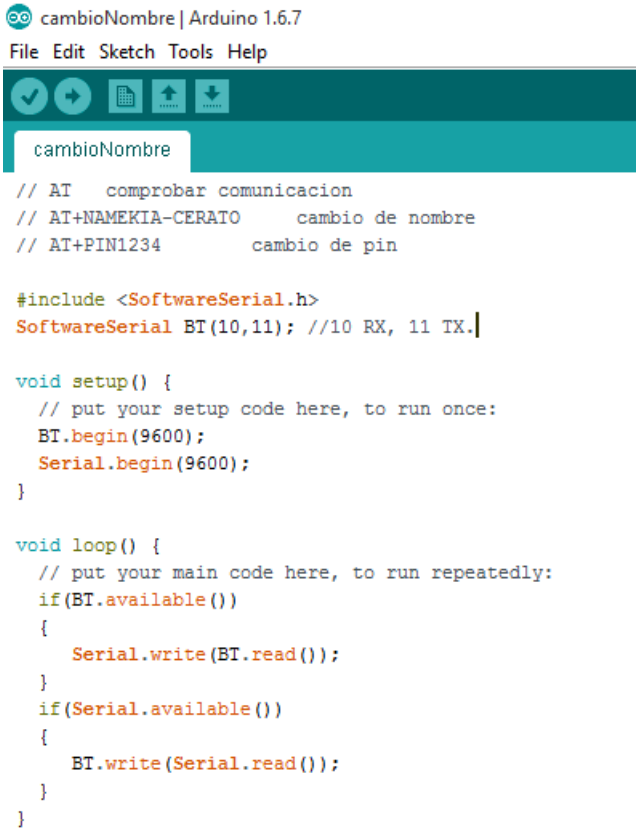
OBDII. (15 de 10 de 2015). Recuperado el 15 de Noviembre de 2015, de OBDII: http://www.multiscanners.cl/atecnicos/conector2.htm

OBDII. (01 de 09 de 2015). Recuperado el 15 de Noviembre de 2015, de OBDII: http://www.obdexperts.co.uk/

ANEXO

ANEXO 1.

PROGRAMACIÓN EN ENTORNO ARDUINO PARA CAMBIO DE NOMBRE HC05.



En donde:

- Incluye la librería para comunicación serial y designa los pines 10 para RX y 11 para TX en la placa Arduino Leonardo.

```
#include <SoftwareSerial.h> // Incluye la librería serial
SoftwareSerial BT(10,11); //10 RX, 11 TX. Designa pines de Arduino
```

- Designa la velocidad de transmisión de la placa Arduino, en este caso 9600 bites.

```
void setup() {
  BT.begin(9600); // Configura velocidades de transmisión
  Serial.begin(9600);} // Configura velocidades de transmisión
```

- Establece la comunicación con el hyperterminal del entorno Arduino.

```
void loop() {
  if(BT.available()) // Indica que si el bluetooth está activo realice las acciones
  {Serial.write(BT.read()); } // Escriba serial y lea bluetooth
  if(Serial.available())  // Indica si serial esta active realice las acciones
  {BT.write(Serial.read());}} // Escriba bluetooth y lea serial
```

Una vez escrita la programación se debe compilar y enviar a la placa Arduino Leonardo por medio de cable USB.
Se abre el hyperterminal y se envía los códigos:

- **AT** para comprobar comunicación, la placa retornará un **OK**
- **AT+NAMEKIA-CERATO** para cambiar el nombre
- **AT+PIN1234** para establecer el PIN de seguridad los números 1234

Después de esto ya se tendrá el bluetooth HC05 configurado para la conexión y uso con la aplicación desarrollada.

ANEXO 2.

PROGRAMACIÓN DE LA PLACA ARDUINO LEONARDO

Programacion_Leonardo | Arduino 1.6.7

File Edit Sketch Tools Help

Programacion_Leonardo §

```
#include <Wire.h>
#include <OBD.h>
#include <SoftwareSerial.h> //Librería que permite establecer comunicación serie en otros pins

//Aquí conectamos los pins RXD,TDX del módulo Bluetooth.
SoftwareSerial BT(10,11); //10 RX, 11 TX.
COBD obd;

const float Ts = 0.1; // Tiempo de muestreo en segundos

int y,z;
int valueRPM, valueODO, value;
float x = 0,xk = 0;
float distancia = 0;

void setup()
{ obd.begin();
  while (!obd.init());
  BT.begin(9600); //Velocidad del puerto del módulo Bluetooth
  Serial.begin(9600); //Abrimos la comunicación serie con el PC y establecemos velocidad
  pinMode(13,OUTPUT);}

void loop()
{if(BT.available())
  {z = BT.read();

    Serial.println(z);
    if (z == 1){digitalWrite(13,1);}
    else{digitalWrite(13,0);}
    calcularDistancia();}

  if(Serial.available())
  {BT.write(Serial.read());}}

void calcularDistancia()
{if (obd.read(PID_RPM, valueRPM)) {
      y = valueRPM;}
  if (obd.read(PID_DISTANCE, valueODO)) {
      x = valueODO;}
  distancia = x - xk;
  xk = x;
  if (distancia>100 || distancia < 0){
    distancia  = 0;}
  BT.print("MV ");
  BT.print(distancia);
  BT.print(" ");
  BT.println(y);
  Serial.print("Distance: ");
  Serial.print(x);
  Serial.print(" RPM: ");
  Serial.print(y);}
```

Done Saving.

En donde:

- Incluir librerías de Arduino y otros fabricantes para la programación del proyecto.

```
#include <Wire.h> // Para la creación de nuevas variables
#include<OBD.h>// Incluye la librería para la utilización de los PID OBD-II
#include <SoftwareSerial.h>  // Incluye la librería para uso de puerto serie
```

- Se desgina los pines de comunicación en Arduino Leonardo.

```
SoftwareSerial BT(10,11); //10 RX, 11 TX
```

- Se desgina las variables que tendrá el programa.

```
const float Ts = 0.1;      // Tiempo de muestreo en segundos
int y,z;   // Se desgina la creación de variables “y” y “z”
int valueRPM, valueODO, value;  // Se desgina la creación de variables
float x = 0,xk = 0; // Se desgina la creación de variables
float distancia = 0; // Creación de variables siempre reseteando a cero
```

- Se inicia las librerías y se establece velocidades de transmisión.

```
void setup()
{ obd.begin(); // Se inicia librería OBD
while (!obd.init()); // Se indica mientras OBD este iniciado, realice la acción
BT.begin(9600);   // Velocidad del puerto del módulo bluetooth
Serial.begin(9600); // Se establece velocidad del puerto serie
pinMode(13,OUTPUT);}  // Se establece el pin 13 de Leonardo como salida
```

- Se establece las acciones a realizar cuando el bluetooth esté conectado al dispositivo móvil.

```
void loop()
 {if(BT.available())  // Indica si el bluetooth está disponible realice la acción
{z = BT.read();  // Indica que la variable “z” será la lectura de bluetooth
Serial.println(z);  // Indica que muestre la variable “z” en la pantalla
if (z == 1){digitalWrite(13,1);} // Indica si “z” es igual a 1 envíe 5V al pin 13
```

```
else{digitalWrite(13,0);} // Indica si “z” no es igual a 1 envíe 0V al pin 13
calcularDistancia();}   // Inicia la acción calcular distancia
if(Serial.available())    // Indica si la comunicación en serie está activa pueda escribir en el bluetooth
{BT.write(Serial.read());}}// Indica que escriba en el bluetooth lo que indique la comunicación serial TX RX
```

- Se establece los comando que contendrá la acción calcular distancia.

```
void calcularDistancia()  // Se abre acción calcular distancia
{if  (obd.read(PID_RPM,  valueRPM))  {  // Se establece el comando para extraer las RPM del vehículo, esto servirá para poder encender y apagar el vehículo desde la aplicación, el valor obtenido será designada como valueRPM
y = valueRPM;} //Se indica que “y” será igual al valor de valueRPM
if (obd.read(PID_DISTANCE, valueODO)) { // Se establece el comando para extraer la distancia que recorre el vehículo en cada evento, esto servirá para poder ingresar el kilometraje recorrido por el auto y así tener el control de los mantenimientos del vehículo
x = valueODO;} // Se indica que “x” será igual a valueODO
distancia = x - xk;  // Se establece que para calcular distancia reste de “x” la variable “x” anterior
xk = x; // Se indica que después de cada resta “x” pasara a ser “xk” o “x” anterior
if (distancia>100 || distancia < 0){distancia  = 0;} // Se establece que si el cálculo realizado anteriormente es mayor a 100 o menor a 0 guarde la distancia recorrida como 0, esto se coloca en afán de no tener desfases en el kilometraje por errores de transmisión de datos
```

- Se designa que variables se enviarán por serial y cuales por bluetooth.

```
BT.print("MV "); // Se transmite por bluetooth encabezado “MV" y espacio
BT.print(distancia); // Se transmite por bluetooth la distancia
```

```
BT.print(" ");// Se transmite por bluetooth un espacio
BT.println(y); // Se transmite por bluetooth un enter y la variable “y”
Serial.print("Distance: "); // Se transmite por serie Distance: y un espacio
Serial.print(x); // Se transmite por serie la variable “x”
Serial.print(" RPM: "); // Se transmite por serie RPM: y un espacio
Serial.print(y);} // Se transmite por serie la variable y
```

Una vez que se termina esta programación, se compila y envía por cable USB a la placa Arduino Leonardo, la cual almacenará esto en su memoria y empezará a correr cuando se conecte por medio del OBD-II Arduino al conector de 16 pines.

Para poder monitorear el programa se podrá conectar a un computador por medio de USB y abriendo el hyperterminal se logrará verificar los datos que son extraídos por Leonardo de la ECU del vehículo.

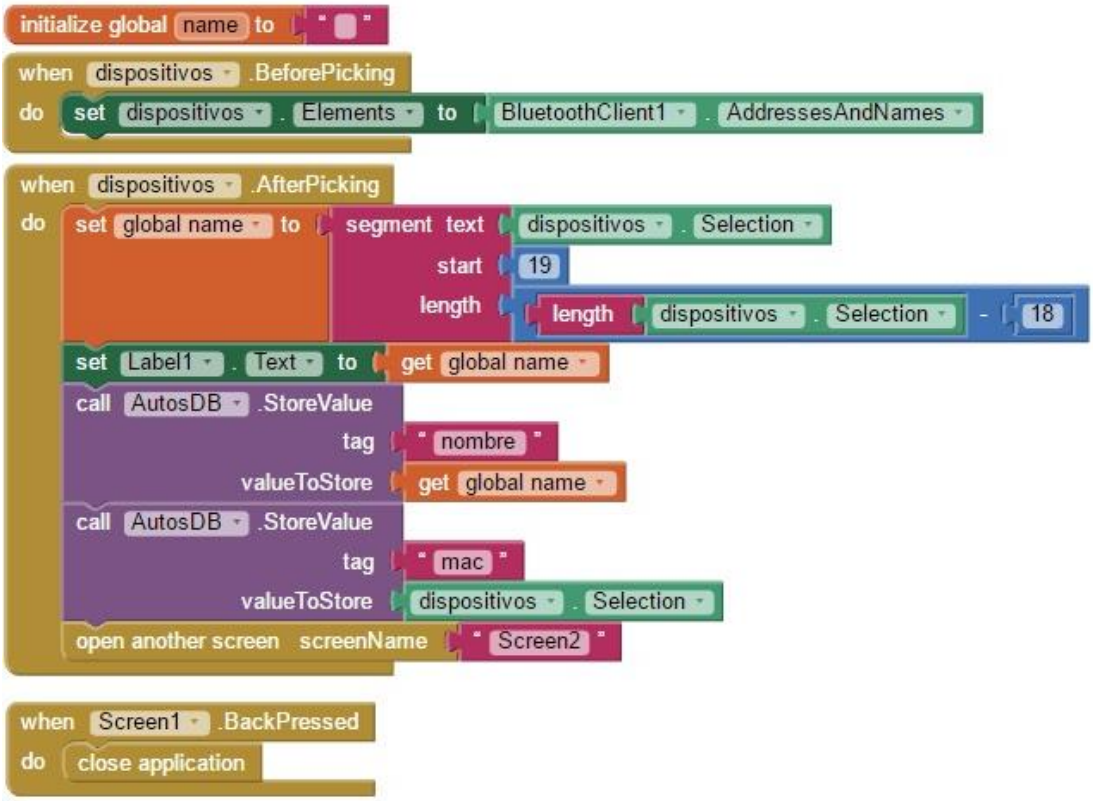
ANEXO 3.

PROGRAMACIÓN DE LA APLICACIÓN EN APP INVENTOR 2

- **PROGRAMACIÓN SCREEN UNO**

Se indica que inicialice la base AutosDB con el nombre y la MAC address del dispositivo bluetooth del vehículo conectado por Arduino, cuando se presione el botón **"Seleccionar Vehículo"** el aplicativo despliegue los bluetooth disponibles encontrados por la tablet.

Después de esto guardará esta información en la base de datos AutosDB y pasará al screen número dos.

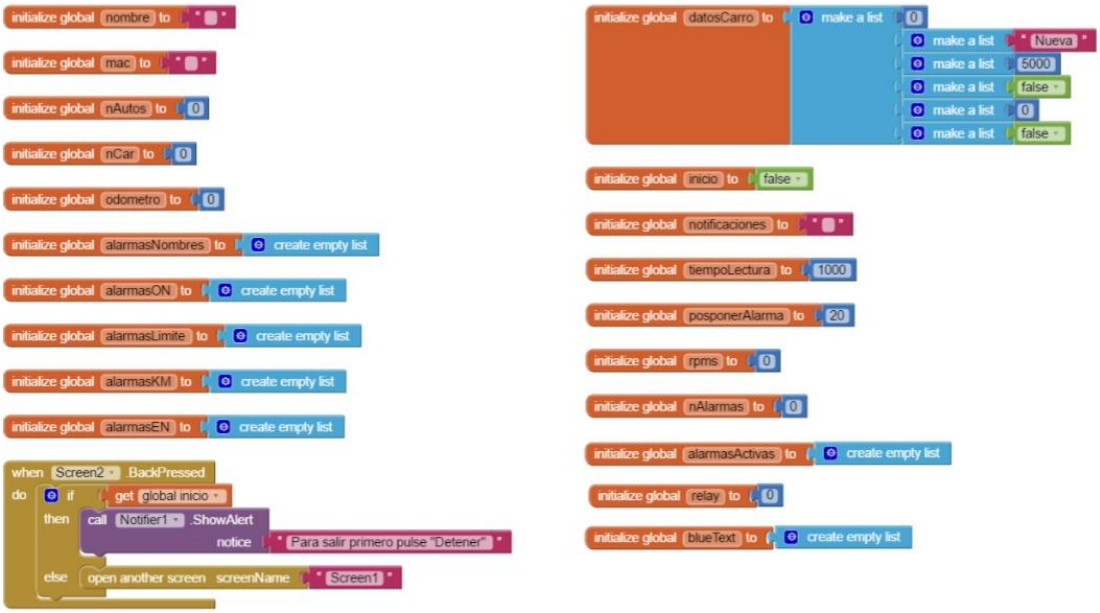


- **PROGRAMACIÓN DEL SCREEN DOS**

Esta programación será dividida en 5 partes de las cuales:

Programación screen dos parte uno.

En esta parte se iniciará todas las variables designadas para la aplicación, cada variable tendrá un espacio o sección en la base de datos AutosDB, así también se designará una regla, que si requieren salir de esta pantalla o regresar a la anterior cuando esta esté inicializada emita una alerta en forma de notificación, se debe recordar que el screen dos tendrá dos estados uno activo y uno inactivo.

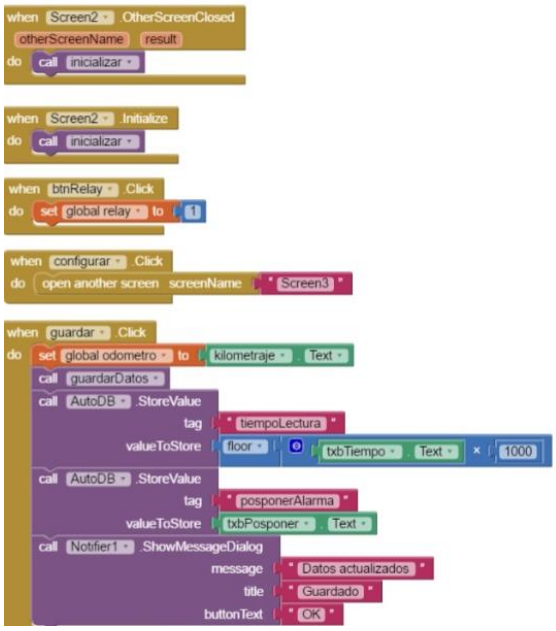


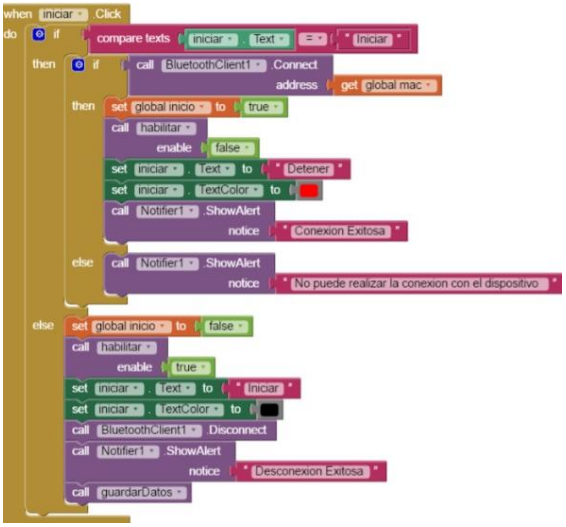
Programación screen dos parte dos

En esta sección se programará las funciones de los botones que se tiene en el screen dos, cabe recordar que el botón “Encender Vehículo” está concatenado dentro del estado inicializado del screen dos y dependerá su actividad de las RPM recibidas del vehículo.

Al presionar “Guardar” se está indicando que guarde en la base de datos AutosDB el valor del odómetro, los tiempos de lectura de las alarmas, las alarmas que han sido pospuestas y los datos que han sido actualizados durante esa sesión.

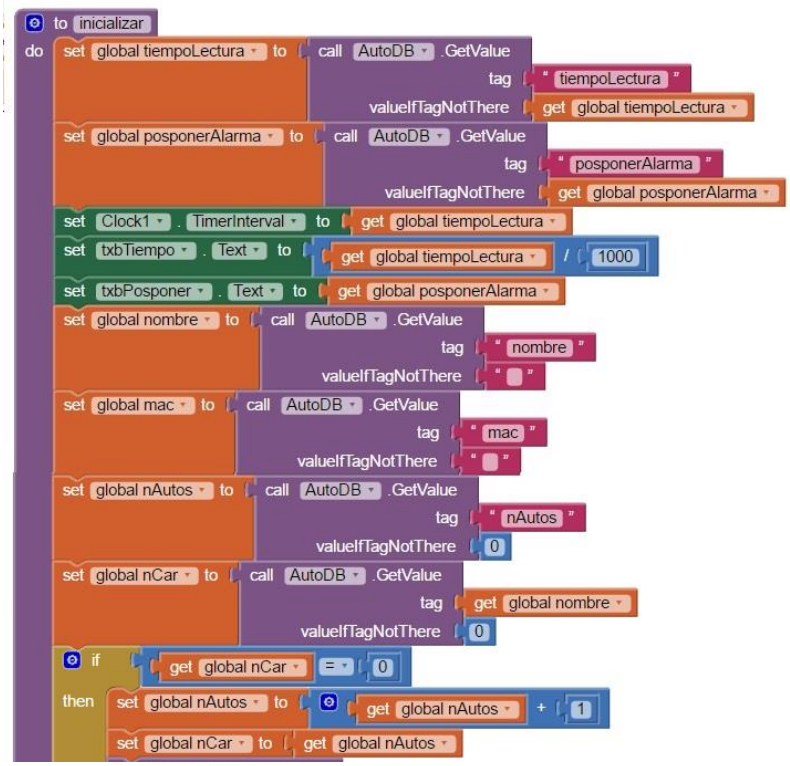
Al presionar el botón relay “Encender Vehículo” lo que hará es enviar el número 1 a la placa Arduino, con esto la placa Arduino Leonardo enviará un pulso de 5 voltios al transistor que activará el relay conectado al encendido del vehículo, como este botón está concatenado a las RPM hará las funciones antes mencionadas siempre y cuando el Screen dos esté inicializado y las RPM sean iguales a cero, caso contrario este botón se deshabilitara. Al presionar el botón “Configurar Alarmas” lo que hará es enviar al screen número tres, el cual tendrá todas las opciones para crear, habilitar y configurar las alarmas de las cuales constará el aplicativo de mantenimiento preventivo del vehículo. Por último se configura el botón “Iniciar” el cual llevará el aplicativo al estado activo del screen dos y donde la aplicación operará la mayoría de su tiempo. Al presionar el botón se programa que la aplicación se conecte al bluetooth seleccionado en el screen uno, si la conexión se realizó sin problemas emitirá la notificación “Conexión Exitosa” caso contrario indicará “No se puede realizar la conexión con el dispositivo” y una vez conectado el botón “Iniciar” cambiará su nombre a “Detener” para poder retornar el screen dos a su estado inactivo, adicional se indica que al presionar “Detener” guarde los datos que se han modificado en la aplicación, así también que emita una notificación indicando “desconexión exitosa”.

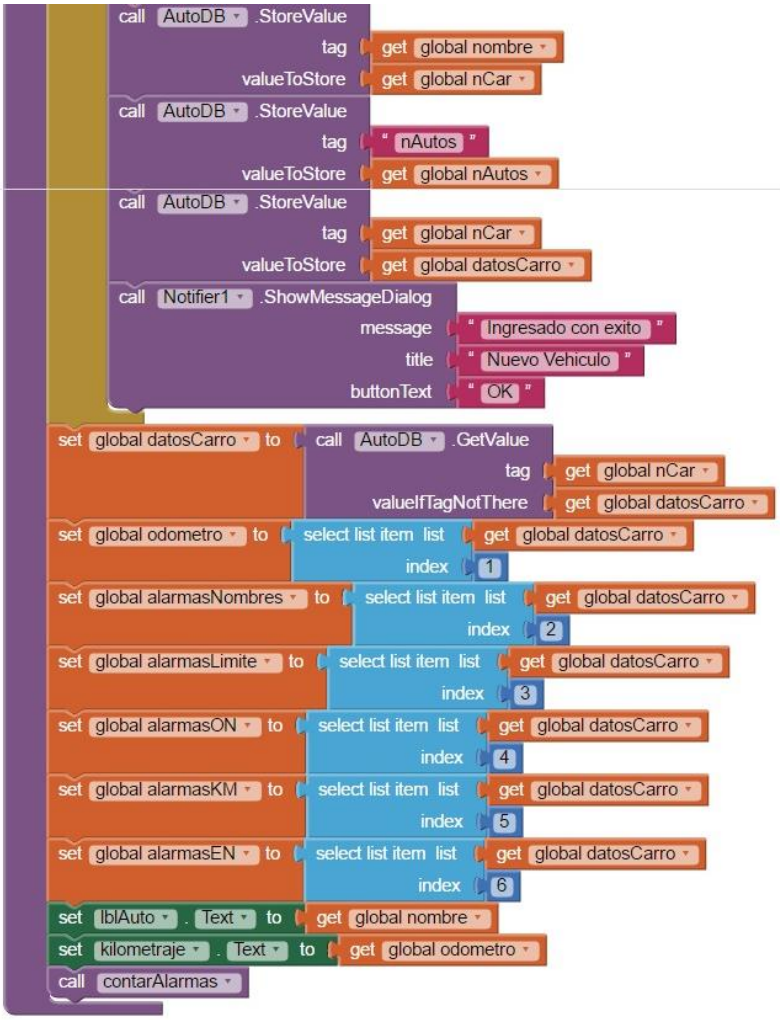




Programación screen dos parte tres.

En esta sección se programará el comportamiento del screen dos en estado activo, donde toda la información generada se guardará en la base de datos AutosDB y la información que se muestra en el screen será rescatada u obtenida de la base de datos es una comunicación bidireccional de la información de la base de datos.



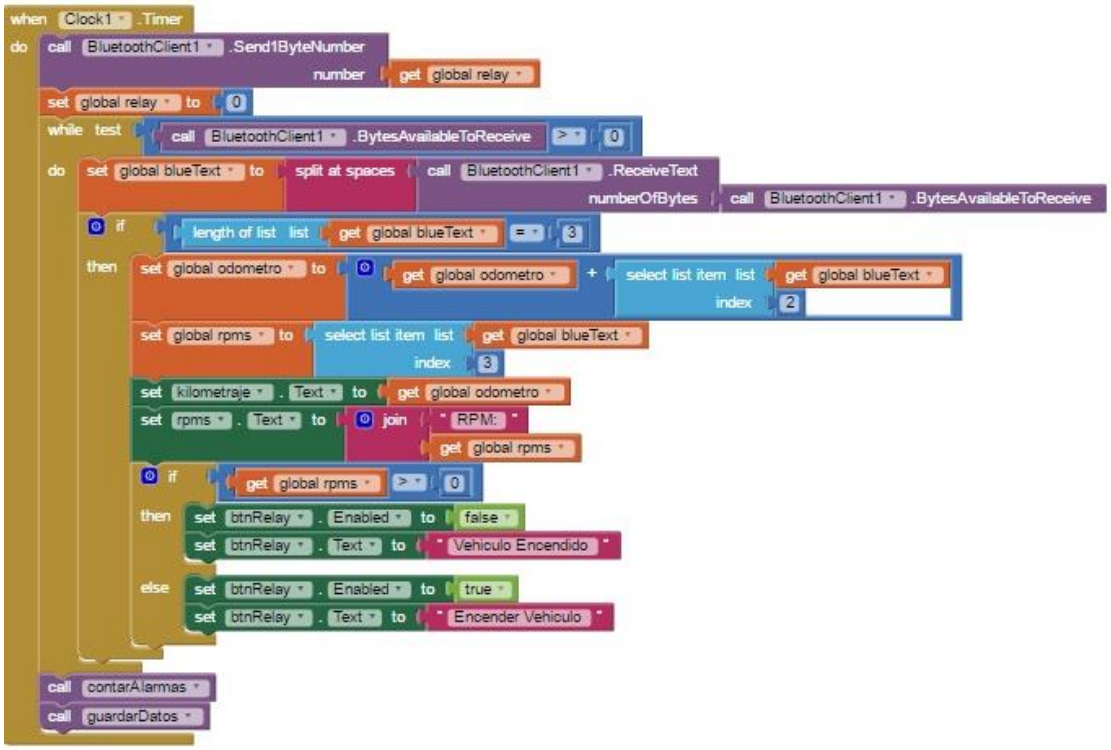


Programación screen dos parte cuatro.

Dentro del estado inicializado del screen dos se debe programar la transmisión de información por medio de bluetooth, aunque la placa Arduino Leonardo recolecte la información de la ECU cada décima de segundo, la transmisión hacia la tablet la controla la aplicación con esto se está indicando que la información del bluetooth que llega en bytes sea ingresada en las variables odómetro y RPM.

La variable odómetro servirá para la activación y control de las alarmas programadas en el aplicativo de mantenimientos y las RPM servirán para controlar el botón “Encender Vehículo” aquí se indica que si las RPM son diferentes a cero el botón cambie a “Vehículo Encendido” como en todos los

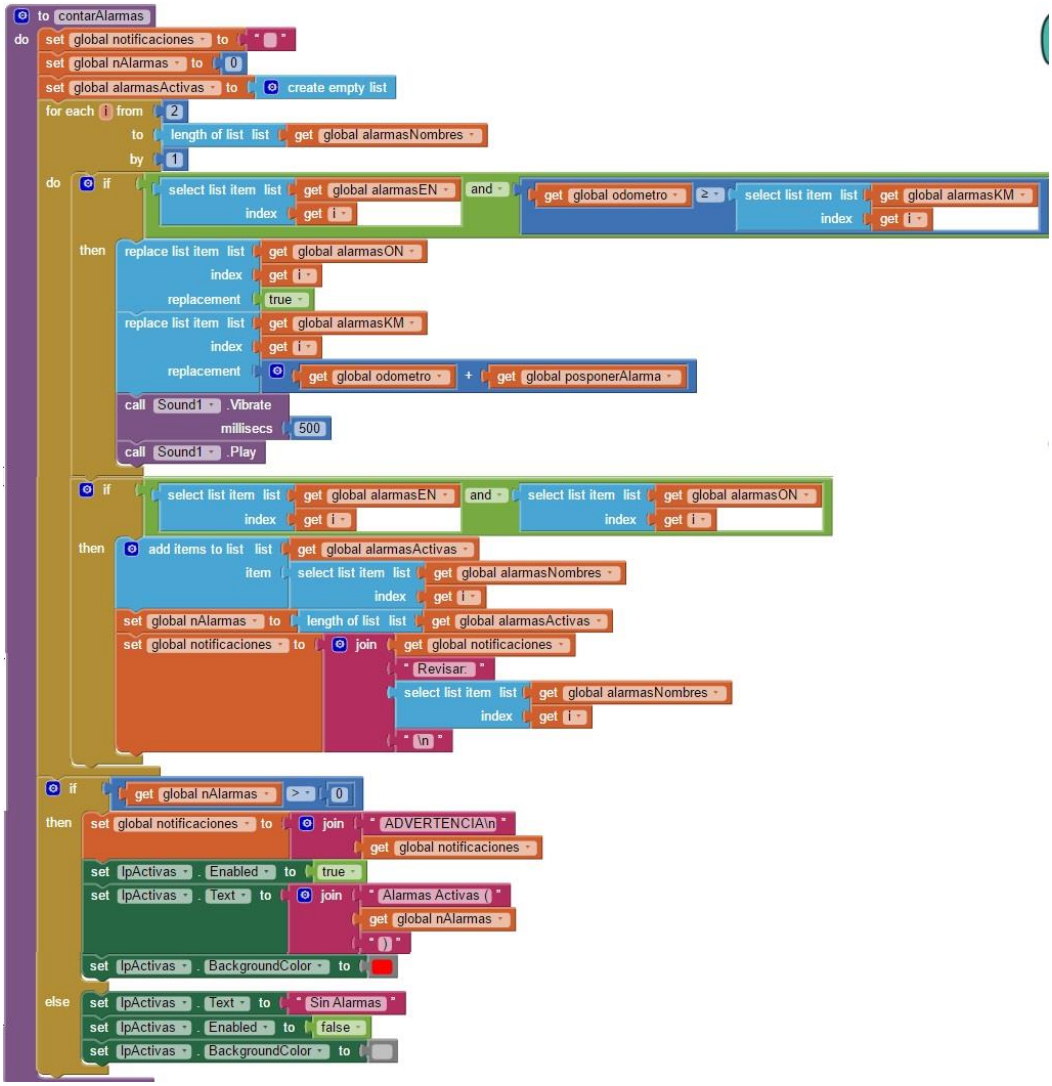
bloques anteriores se indica el llamado a la funciona guardar, que en este caso sería algo similar a un autoguardado.



Programación screen dos parte cinco.

En esta sección se programará el comportamiento de las alertas o alarmas creadas, lo primero que realiza la aplicación es una comprobación de las alarmas creadas, verifica si están activas o desactivas. Luego guarda temporalmente el próximo kilometraje de la alarma y de esta estar pospuesta lo guarda para su próxima notificación.

Como siguiente paso se programa la forma en la cual se notifican las alarmas activas o que se activaron durante el evento y emite una alerta auditiva que está declarada como sound1, luego designa variables temporales de las alarmas activas las cuales se podrán modificar cuando el screen dos no este inicializado y se pueda acceder al screen tres por medio de el botón “Configurar Alarmas”.

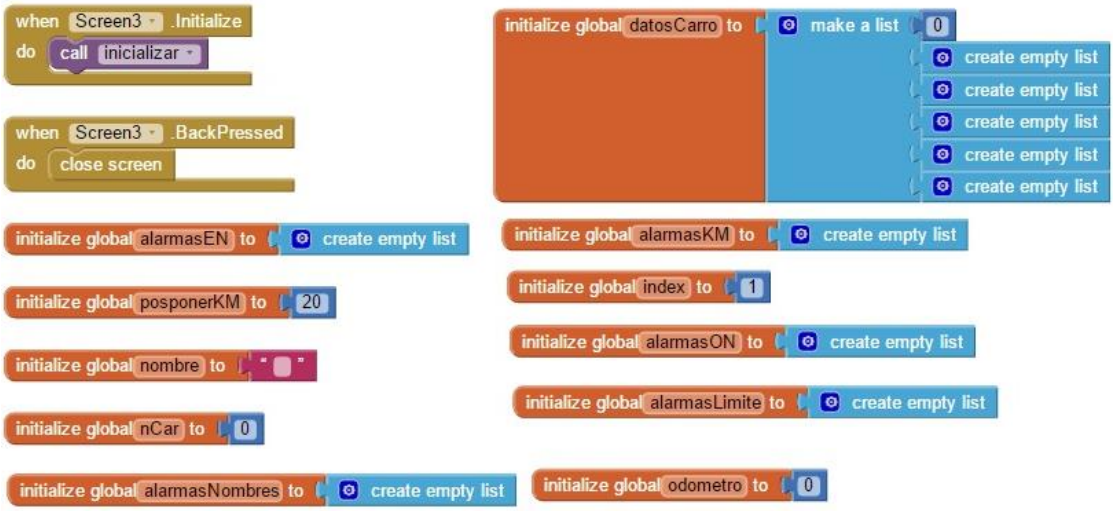


- **PROGRAMACIÓN DEL SCREEN TRES**

La programación del Screen 3 será dividida en tres partes:

Programación screen tres parte uno

En esta sección se inicia todas las variables de la base de datos para poder precargar las alarmas ya definidas y creadas por defecto o por el usuario, esta acción extrae la información de la base de datos AutosDB, adicional se programa que al presionar el retorno, cierre la ventana número 3 y retorne a la 2.



Programación screen tres parte dos

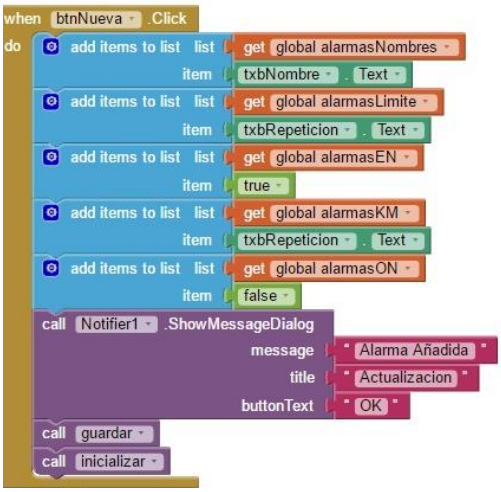
En la segunda sección de programación del screen tres se indicará las acciones y botones disponibles en el screen, teniendo así disponibles los botones; “Guardar Cambios”, “Guardar Nueva”, “Eliminar” y “Realizar Revisión”.

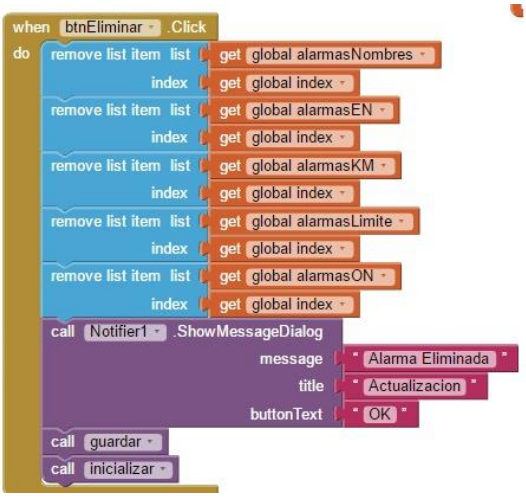
El botón “Guardar Cambios” lo que hace es indicar la modificación de todos los parámetros ingresados por los que se encuentran en la base de datos incluso cambiando el estado y configuración de la alarma de haber realizado algún cambio.

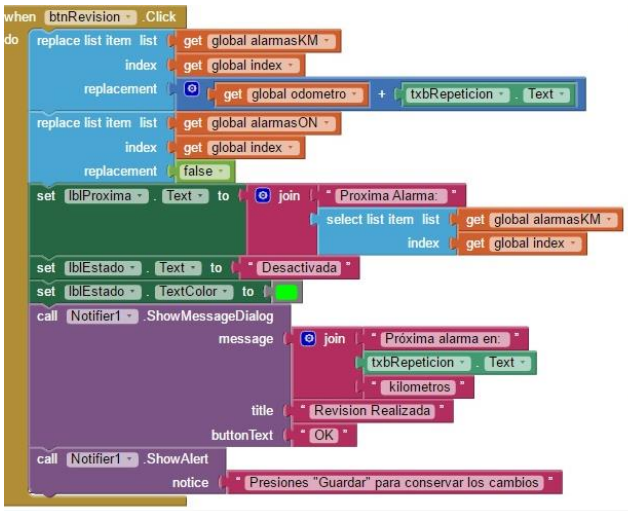
Para el botón “Guardar Nueva” la aplicación va a generar un nuevo campo en la base de datos e ingresará los datos de kilometraje, nombre, estado y configuración de la alarma. No existe un límite de creación de alarmas para el dispositivo.

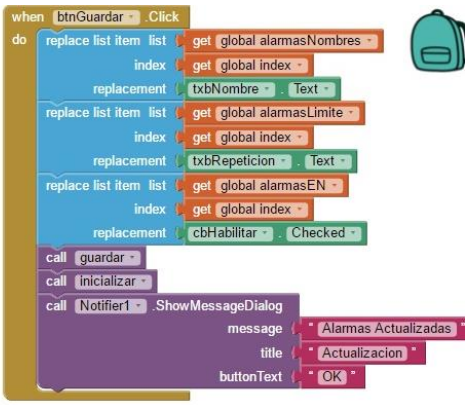
El botón “Realizar Revisión” será ocupado cuando una alarma haya sido activada por el aplicativo y ya con el vehículo en el taller una vez realizada la inspección validar digitalmente la misma y restablecer los contadores de la alarma.

El botón “Eliminar” lo que hará es conectar a la base de datos AutosDB y eliminar completamente la sección con el nombre designado para la alarma, con esto todos los datos registrados en la base de datos AutosDB serán desechados permanentemente.



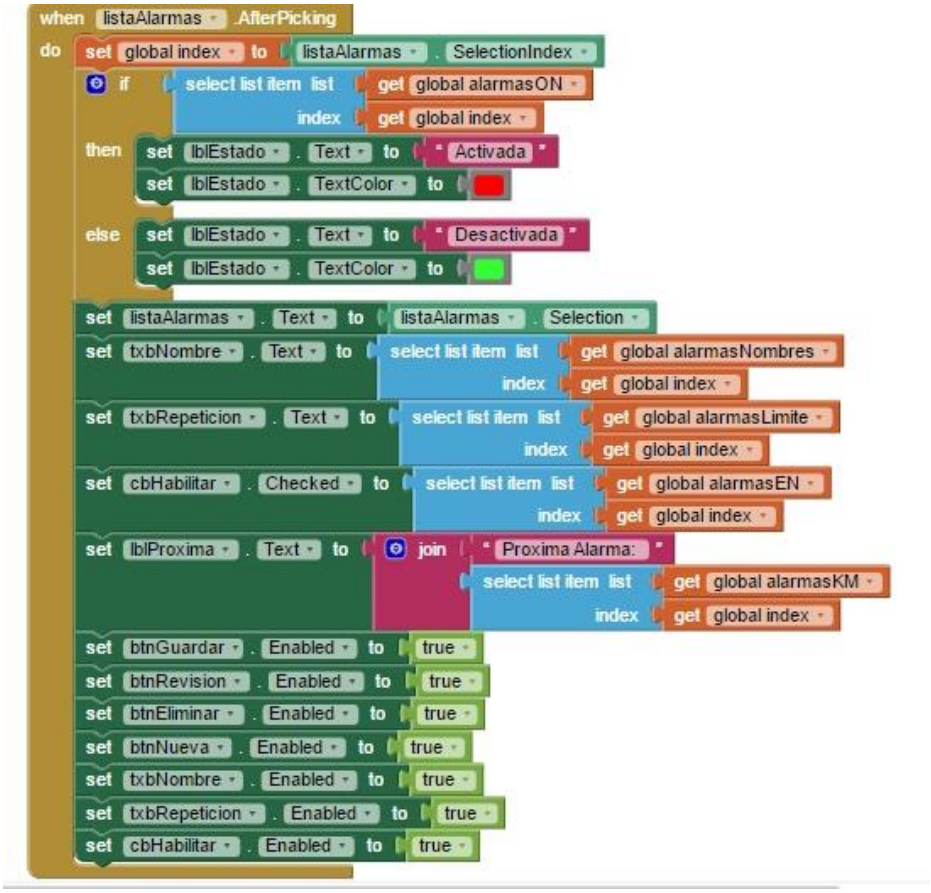






Programación screen tres parte tres.

En la sección número tres de la programación del screen 3 se configurarán la lista de las alarmas generadas en la aplicación, aquí se designará el comportamiento de las mismas y cómo van a presentarse ante el screen número 3.



Al seleccionar la lista de alarmas que tiene el dispositivo se desplegará el estado de la misma si esta activa en rojo y si está inactiva en verde, adicional se habilitarán todos los campos de ingreso de datos y botones para poder modificar la alarma ya que despliega y abre todas las variables guardadas en la base de datos con respecto a la alarma seleccionada.

ANEXO 4.

FICHA TÉCNICA ARDUINO LEONARDO

LCA1008

ARDUINO LEONARDO

Características:
Leonardo es el nuevo modelo de Arduino.
Utiliza un microcontrolador ATmega32U4 que permite un diseño mucho más sencillo y económico.
Una de las ventajas de este nuevo microcontrolador es que dispone de USB nativo por hardware y por lo tanto no necesita de ningún conversión serie-USB.
También permite a la placa ser utilizada y programada como un dispositivo de entrada para emular un teclado, ratón, etc.

Tiene 12 entradas analógicas y dado que el puerto de comunicación USB es emulado, deja el puerto serial hardware libre para la programación!
De esta forma ya no ocurren conflictos de programación mientras tenemos periféricos serial conectados a la placa.

Especificaciones:
Microcontrolador: ATmega32u4
Tensión de funcionamiento: 5V
Alimentación recomendada: 7-12V
Pines I/O Digitales: 20
Canales PWM: 7
Entradas analógicas: 12
Corriente Maxima de los pines I/O: 40 mA
Corriente Maxima de los pines 3.3V: 50 mA
Memoria Flash: 32 KB (4 KB usados para el bootloader)
SRAM: 2.5 KB
EEPROM interna: 1 KB
Velocidad: 16 MHz

ANEXO 5.

FICHA TÉCNICA BLUETOOTH HC05

HC-05

-Bluetooth to Serial Port Module

Overview

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.
Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector