SISTEMA DE MONITOREO PARA MOTOCICLETAS CON TECNOLOGÍA ARDUINO Y ANDROID

Fabio Fernando Martínez Orozco

John Fredy Callejas Piñeros

Universidad Nacional Abierta y a Distancia UNAD

Escuela de Ciencias Básicas Tecnológicas e Ingeniería

Ingeniería Electrónica

Ibagué

2016

SISTEMA DE MONITOREO PARA MOTOCICLETAS CON TECNOLOGÍA ARDUINO Y ANDROID

Fabio Fernando Martínez Orozco

John Fredy Callejas Piñeros

Asesor: Noel Jair Zambrano Sánchez

Universidad Nacional Abierta y a Distancia UNAD

Escuela de Ciencias Básicas Tecnológicas e Ingeniería

Ingeniería Electrónica

Ibagué

2016

Nota de aceptación

_____

_____

_____

_____

_____

Director del proyecto

_____

Firma del Jurado

Ibagué, Julio de 2016

Tabla de Contenidos

Lista de figuras

Lista de tablas

Resumen

Es posible realizar el control del sistema eléctrico que compone un motor de combustión interna de una motocicleta de forma fácil y segura, utilizando dispositivos compatibles con el sistema operativo Android, (celulares y tablas con servicio GPRS); utilizando una aplicación que comunique por medio de una conexión Bluetooth y/o GPRS el sistema electrónico desarrollado con nuestro dispositivo Android, con el fin de tener un medio virtual para que las motocicletas puedan ser apagadas y encendidas. Este circuito es complementando con un sistema de alarma de robo y forcejeo, teniendo la ventaja de saber por medio de un mensaje de texto, si la alarma fue activada.

Este sistema cuenta con un sistema de control de activación o desactivación de forma manual, gracias a un emisor de RF portátil tipo llavero, con el fin de que el usuario no posea el celular a mano o este se encuentre descargado. Por último, el sistema también contara con un sistema GPS, indicando al usuario donde se encuentra el dispositivo en caso de robo.

Palabras Clave: Control, motocicletas, sistema de combustión, GPS, GPRS, Bluetooth, Arduino.

Abstract

It is possible to control the electrical system that composes an internal combustion engine of a motorcycle safely and easily, using compatible devices with the Android operating system (cellular and tables with GPRS); using an application, which communicates through a Bluetooth connection and / or GPRS electronic system developed with our Android device, in order to have a virtual means for motorcycles can be turned off and on. This circuit is complemented by an alarm system theft and floundering, having the advantage of knowing through a text message, if the alarm was activated.

This system has a control system on or off manually, thanks to an RF transmitter key portable type, so that the user does not have the phone handy or this be downloaded. Finally, the system will also have a GPS system, telling the user where the device in case of theft.

Keywords: Control, motorcycles, combustion system, GPS, GPRS, Bluetooth, Arduino.

Introducción

Colombia posee un nivel de delincuencia muy elevada en las capitales de cada uno de los departamentos, con un total del 27.260 de motocicletas robadas en el 2015. Teniendo en cuenta esta cifra, el hurto de motocicletas ha aumentado en este primer trimestre del año en un 2,9%. (Fernández de Soto, Moreno, Restrepo, & Villegas, 2016). Estas Cifras son un claro ejemplo del problema social que se enfrentan todos los días los propietarios de motocicletas.

De tal forma se plantea la implementación de un dispositivo de seguridad electrónico con sistema de geoposicionamiento, el cual contara con varios servicios tales como Bluetooth, GSM, GPS, y control manual remoto.

Existen en el mercado dispositivos de alarma GSM/GPS con interfaz para Smartphone que ayudan en la minimización de robos en las motocicletas, pero estos son caros y poseen ciertas falencias en sus diseños sin tener en cuenta muchos de los problemas cotidianos que presentan los celulares.

En el proceso de diseño, se seleccionara Módulos y Shields para Arduino, ya que estos cuentan con excelentes microcontroladores y vienen ya prediseñado en una placa, con su propio sistema de regulación e interconexión a diferencia de los PICs.

Objetivos

Objetivo general

Diseñar e implementar un sistema electrónico de geoposicionamiento y control sobre el sistema de ignición de una motocicleta utilizando un APK para celulares con sistema operativo Android.

Objetivos específicos

Plantear el control de encendido y apagado de la motocicleta desde un celular o tabla que posea sistema operativo Android.

Visualizar en la pantalla del dispositivo Android por medio de una aplicación APK, la ubicación de la motocicleta implementando un sistema GPS.

Desarrollar una aplicación Android, que contenga un tablero de control de fácil manejo para el usuario.

Establecer la comunicación del sistema de control electrónico y la aplicación para dispositivos Android, por medio de tecnologías de comunicación GPRS y Bluetooth.

Comprobar el funcionamiento de la aplicación APK y el sistema de control electrónico.

Considerar el uso de un pequeño control de radio frecuencia para el modo manual del sistema de seguridad.

Definición del problema

El hurto de motocicletas es un delito común tanto en las ciudades como en los pueblos. Según el periodista del diario El Tiempo, quien publico el 27 de Mayo del 2015 un reporte titulado "Todos los días se roban 64 motos en ciudades capitales de Colombia", en el 2014, 23.398 motocicletas fueron robadas. (Avila Jimenez, 2015). Entre los métodos de hurto más comunes se encuentra el modo de atraco, donde se efectúa cuando los ladrones siguen a su víctima hasta ver el momento justo para abordarlos y amenazarlos con algún tipo de arma despojándolo de su vehículo. El modo de halado, es en el cual los ladrones aprovecha el descuido de los propietarios cuando estos dejan los vehículos en vías públicas sin ningún sistema de seguridad donde los ladrones rompen el traba cabrillas y por medio de ganzúas encienden la moto y huyen o simplemente siendo empujadas por otro delincuente. Las motos robadas son llevadas a bodegas donde las desarman por completo para vender sus partes por separado o son carnada para la extorsión hacia el propietario del vehículo, pidiendo grandes cantidades de dinero por el rescate de la motocicleta. Por lo tanto, ¿de qué forma se puede controlar el hurto de motocicletas en Colombia?

Justificación

Como resultado al proceso educativo, es necesario demostrar las fortalezas en cada una de las asignaturas vista en todo el curso, que aportan para el desarrollo profesional de cada uno de los estudiantes. Por lo tanto, es necesario reflejar tales destrezas en un proyecto que beneficie a las personas y que a su vez, en un futuro profesional irradien las capacidades y aptitudes que se desarrollaron a lo largo del plan de estudios establecido por la universidad.

Este proyecto tiende a disuadir o evitar posibles casos de hurto de motocicletas por medio de tecnología de fácil acceso, económico y con la confianza donde no se intervenga con el funcionamiento normal de la motocicleta. De esta forma los usuarios estarán siendo comunicados si la motocicleta fue encendida, trasladada sin autorización o si fue forzada. De esta forma se aplicara los conocimientos generados a lo largo del proceso académico, con fundamentos de un Ingeniero Electrónico.

Estado del arte

A la hora de hablar de sistema de seguridad para vehículos, nos encontramos con una gran variedad de productos que ayudan a minimizar los casos de hurtos en motocicletas y automóviles. Muchos de estos dispositivos son sencillos y solo manejan un circuito de On/Off, dependientes directamente de la batería principal del vehículo. Por consecuente, son muy vulnerables y de poca confianza.

A medida que la tecnología avanza, se crean productos cada vez más confiables los cuales poseen: comandos a distancia, posicionamiento global, control de voz, llaveros magnéticos, sistema biométrico entre otras.

En el mercado actual, existen alarmas para vehículos con sistemas de geoposicionamiento los cuales tienen funciones de GPS y GSM. De las más destacables encontramos la LCF Bike, cuyas características son; sistema GPS y GSM, batería de respaldo, sensores de golpe e inclinación, y desactivación del motor por mensaje de texto (ICF Bike, 2012). Otra marca de alarma es la Bretto w200, cuyas características son muy similares a la LCF Bike, con la diferencia de que esta posee un sistema de alarma de velocidad programable (Bretto, 2015). Si el usuario necesita un monitoreo más preciso del vehículo, marcas como Sherlog, ofrecen servicios GPS pero requiere de un pago mensual para su uso. En el caso si se requiere servicios de control del motor a distancia, la Pelacrash Scorpion, ofrece un sistema de comunicación directa por medio de antenas FM,

ofreciendo cobertura de hasta 1 Kilometro, siendo útil para situaciones de robo directo del vehículo. (Pelacrash, 2014).

Se han desarrollado proyectos similares de alarmas GPS/GSM enfocadas a otras áreas como por ejemplo el sistema de localización y bloqueo de maquinaria agrícola vía GSM/GPS, desarrollado por Mario Bellacetin (2013), el cual implemento un sistema de seguridad donde utilizo inicialmente un Arduino UNO, un Shield GPS EM406A y un Shield GSM SM-5100B. En las pruebas llevadas en campo abierto, No especifica en el documento la precisión del GPS, pero si, que variables interfieren en la exactitud de la misma. (Bellacetin, 2013).

Proyectos como: Prototipo de seguridad y vigilancia aplicable a medios de transportes público, desarrollados por los estudiantes de la universidad Católica de Colombia, diseñaron un sistema GPS y GSM con tecnología Raspberry Pi, con el objetivo de implementar en un vehículo un sistema embebido que posea cámara, GPS y Modem inalámbrico para hacer un monitoreo visual y constante al vehículo. (Barahona & Quitian, 2014).

Abilio Marques (2008), implementa un circuito que vincula GPS, GSM y sistema de control en una sola tarjeta. Este dispositivo cuenta con sistema de control por mensaje de texto y de geoposicionamiento pero no interviene con sistemas convencionales en el mercado. La idea principal es de hacer una placa exclusiva y propia que al final resulta ser un dispositivo con una funcionalidad básica en sus servicios. (Marques, 2008).

Las ventajas que se posee al construir este tipo de proyecto frente a los productos que se ofrecen en el mercado, es la utilización de diversidad de módulos para Arduino, ya sea de comunicación o de monitoreo de variables físicas. Por tal razón el sistema de

alarmas y de geoposicionamiento que se puede desarrollar con componentes Arduino, puede adaptarse al cliente teniendo en cuenta lo que quiere el propietario del vehículo, y no estar ligado a las características limitadoras de los productos convencionales.

Desarrollo teórico

Arduino

Son pequeños microcontroladores de plataforma de hardware libre, en una placa diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. Este hardware consiste en una placa con un microcontrolador Atmel AVR, con puertos de entrada y salida.

Su software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing-Wiring y un cargador de arranque. En este dispositivo se puede para utilizar el desarrollo de objetos interactivos autónomos, o bien, ser conectados a un software del ordenador. (Doutel, 2015)

Figura 1. Módulo de Arduino UNO.

Fuente página web:

http://www.iberobotics.com/shop/index.php?cPath=64_83&osCsid=97b61d757159ad6ab44e43e293319a19

El entorno de desarrollo libre puede ser descargado gratuitamente, siendo de una distribución libre, con el fin de ser utilizado para el desarrollo de cualquier proyecto sin alguna licencia. Por ser sistemas de desarrollo libre se utiliza en gran variedad de proyectos electrónicos incluyendo sistemas de seguridad, ya sea para casas o para vehículos, ya que este dispositivo electrónico cuenta con una variedad de Shields, como: sistemas de comunicación, sensores, controles, manejo de señales análogas y digitales, entre otras., complementando la aplicación del circuito de una forma más compacta y precisa al proyecto. (Doutel, 2015).

Modulo GPS Arduino

El sistema GPS se basa en una serie de 24 satélites que determinan la posición en tres dimensiones; Latitud, Longitud y altura; de algún dispositivo remitente en cualquier parte del mundo a cualquier hora del día y de forma gratuita. Estos sistemas contienen relojes atómicos para un mejor sincronismo donde se activan todos al mismo tiempo. Estos satélites se posicionan a 26 kilómetros de la tierra, dando un periodo de la órbita de la tierra en 12 horas. (Rodriguez, 2010).

Figura 2. Shield con dispositivo GPS para Arduino.

Fuente, página web: http://blog.bricogeek.com/noticias/tutoriales/tutorial-arduino-gps-logger-con-em406a-gps-shield-y-microsd-shield/

El modulo Neo 6m Ublox es utilizado comúnmente para proyectos de vehículos autónomos como aeromodelismos Drones, Robots móviles o controles de vuelo puesto que es muy potente, con una antena integrada donde se consiguen datos cada segundo. Los datos son entregados por comunicación serial, por lo que sirve para cualquier dispositivo microcontrolados que posea este tipo de comunicación. (Tutorial Arduino: GPS logger con EM406A, 2016).

Shield Bluetooth Arduino

Este Shield, puede ser utilizado de forma rápida y fácil con dispositivos Arduino, siendo transparente para la comunicación inalámbrica, por medio de una transmisión de puerto serial. Estos dispositivos son compatibles con PIC´s, siendo muy económicos y populares para proyectos que definen transmisiones de corto alcance. El modulo Bluetooth, se vende de forma independiente del Shield, reduciendo más el costo y con pines que pueden ser utilizados para desarrollo en proyectos con Protoboard o cablearlo directamente al microcontrolador (Prometec.Net, 2014).

Figura 3. Shield Bluetooth para Arduino.

Fuente página web: http://www.prometec.net/tag/bluetooth/

El modulo más común y es el HC-06, el cual permite configurar el dispositivo por medio de comandos AT, donde se pueden cambiar parámetros tales como: Nombre del dispositivo, Contraseña, velocidad de comunicación, Tipo de función (esclavo y/o maestro), entre otras., por lo que puede recibir conexiones de una tabla y se capaz de generar conexiones hacia otros dispositivos Bluetooth.

Para la conexión para Arduino se requiere solo la conexión de alimentación y las de recepción serial RX a la transmisión serial del Arduino TX y viceversa con la transmisión de Bluetooth con la recepción del Arduino. (Jesus, 2014). (Prometec.Net) (Todo Motos, 2013)

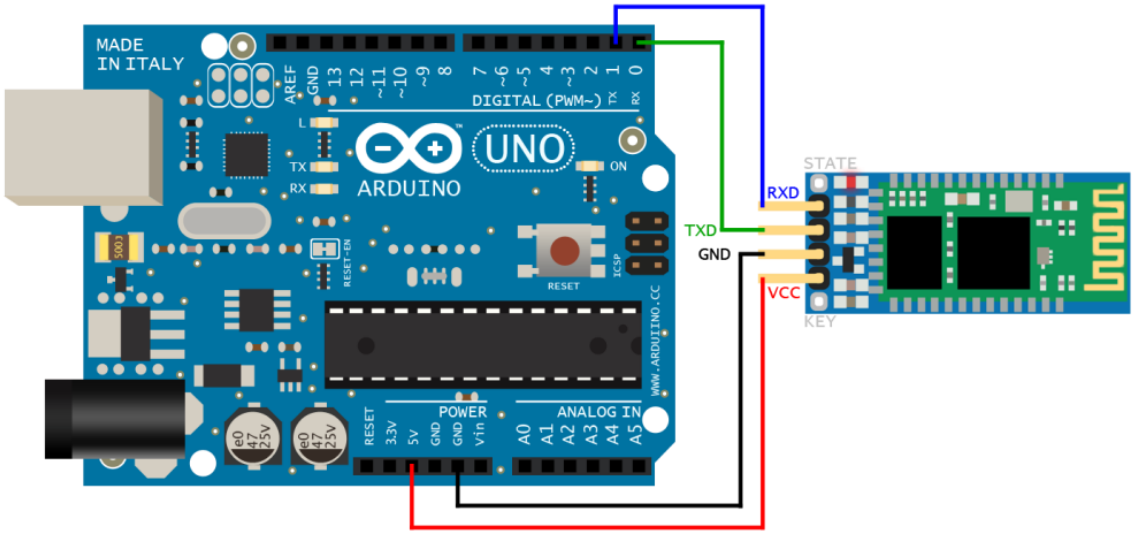

Figura 4. Conexión de un módulo Bluetooth a un Arduino UNO.

Fuente página web: http://www.geekfactory.mx/radio/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/

Shield GPRS/GSM Arduino

Este dispositivo permite a un dispositivo Arduino opere como un teléfono GSM programando de forma correcta las funciones del mismo. Dentro del hardware del Shield, se encuentra: Modem GSM/GPRS, Basado en el chip SIM900, conectores de entrada y salida de audio, reloj RTC, con batería, pines de GPIO libres y controlables mediante comandos AT y conexión RS232 hacia el Arduino. Se utiliza para construir Alarmas de seguridad de hogar por medio de mensaje de texto, hasta proyectos para contestar llamadas.

Posee un amplio set de funciones, entre las cuales están GPRS, TCP, UDP, PPP, FTP, HTTP, SMS, Voz y FAX. Por esto puede ser usado en diversidad de aplicaciones. La tarjeta incluye todos los componentes necesarios para operar el Modem, tales como regulador e interfaces de SIM-Card, UART, antena, audio, botones y LEDs de control. (Electan, 2016).

Figura 5. Shield GPRS para Arduino.

Fuente página web: http://www.electan.com/arduino-shield-gsmgprs-sparkfun-con-sm5100b-p-3151.html

Sistema operativo Android

Es una plataforma de software para módulos móviles que incluye un sistema operativo y aplicaciones de base. Android es un ligado de herramientas y aplicaciones sujetadas a una distribución Linux para dispositivos móviles. No es un Sistema Operativo Android como tal, es de código abierto, gratuito y no requiere pago de licencias. Android es una plataforma de código abierto para terminales móviles que se basada en Linux y es desarrollada por Open Handset Alliance, donde se dice que los primeros teléfonos con Android aparecieron en el segundo semestre de 2008 y grandes compañías como LG, Motorola y HTC ya han diseñado alguno de los modelos que incorporarán el Sistema Android. (Lipa Chouqe).

Es una stack de software para mecanismos móviles que incluye un sistema operativo, Middleware y aplicaciones de base. Los desarrolladores pueden crear aplicaciones en la plataforma usando el SDK de Android. Las atenciones se han escrito manejando el lenguaje de programación Java y se ejecutan en Dalvik, una máquina virtual personalizada que se ejecuta en la parte superior de un núcleo de Linux. (Rodriguez, 2010).

Motor De Combustión Interna

Es un tipo de motor que quema la mezcla de combustible y aire dentro de una cámara cerrada o cilindro, incrementando la presión para generar potencia en un movimiento lineal por medio de un pistón. Este movimiento se transmite por medio de la biela al cigüeñal donde se convierte en movimiento rotativo siendo transmitido por piñones hasta llegar a las ruedas. (E-auto).

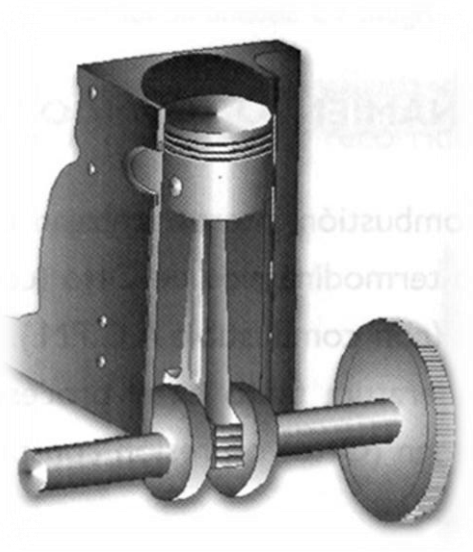

Figura 6. Conjunto móvil.

Fuente página web: http://admin.banrepcultural.org/node/92121

Mediante el proceso de la combustión, la energía química se transforma en energía calórica y en energía cinética. Esta última es utilizada en el trabajo útil del motor, con otros tipos de componentes que mejoran el rendimiento de la combustión. Este tipo de motor necesita de una mezcla precisa de aire y combustible, la cual es generada por medio del carburador o inyectores, dependiendo del tipo de motor. Una vez inducida la mezcla en el cilindro, se genera una chispa eléctrica para provocar la combustión dentro de la cámara del cilindro.

La energía liberada en el proceso de combustión se transmite por una biela al cigüeñal, donde se encuentra unos descentramientos, que apoya la biela generando un movimiento circular en vez de uno lineal. Este movimiento es sincronizado por un sistema valvular, donde se componen de dos tipos de válvula; Admisión y de escape. Todo esto se sincroniza por medio del ciclo de combustión constituido por la admisión, compresión, expansión y escape. (E-auto).

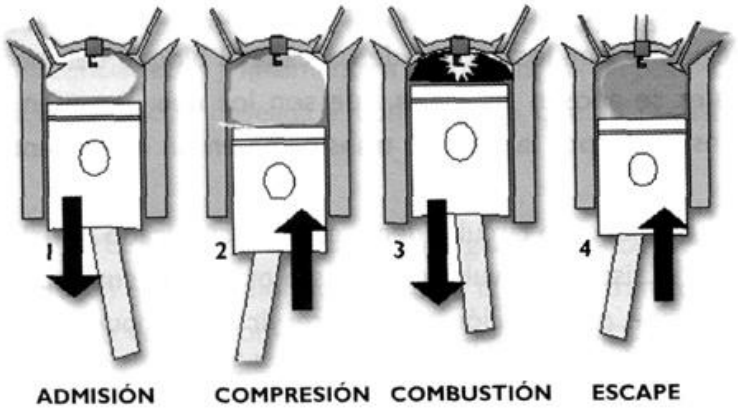

Figura 7. Ciclo de combustión en un motor 4 tiempos

Fuente página web: http://admin.banrepcultural.org/node/92121

Sistema de ignición

Se define como el proceso necesario para encender la mezcla de aire y combustible en la cámara de combustión justo en el momento adecuado, con la finalidad de una mejor eficiencia del motor a una presión máxima de combustión. Este tiempo desde la ignición de la mezcla del aire y combustible y la compresión de la misma, varía dependiendo de las revoluciones que lleva el motor. Si las revoluciones son altas, la ignición debe ocurrir antes. Caso contrario si la revolución es baja la ignición debe ser después. Esto sin tener en cuenta otros factores como la forma de la cámara de combustión, la temperatura de la misma, entre otras.

La generación del encendido por chispa es dada por la bobina de encendido, lugar en la que ésta genera suficiente energía para producir corriente eléctrica necesaria encendiendo la mezcla carburante. El campo magnético es generado por una corriente que circula en la bobina primaria, donde entre sus características, posee una resistencia muy

baja para generar tal corriente eléctrica, conjunto a un transistor de potencia que maneja las altas tensiones que necita la bobina primaria. (E-auto).

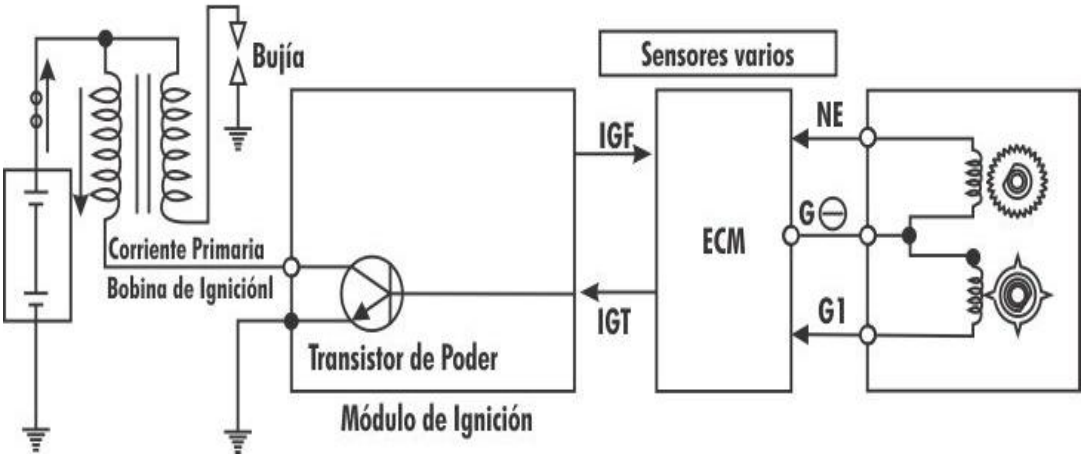

Figura 8. Módulo de ignición

Fuente página web: http://e-auto.com.mx/manual_detalle.php?manual_id=246

CDI

Es un circuito que se encarga de dar la señal a la bobina de alta para que induzca la chispa en la bujía incidiendo en el encendido de la moto. Por lo tanto, este dispositivo se aplica en motos de cuatro tiempos, cuando el pistón debe caer dos grados antes que el mismo llegue al punto muerto a 600 revoluciones por minuto, provocando que los intervalos de señal aumente para dar una señal más anticipada. Este es un dispositivo sencillo el cual contiene un cable hacia el sensor de chispa o un imán que se encuentra rotando en el volante del motor. En este componente, se encuentran condensadores con un interruptor de silicona, que genera un circuito abierto o cerrado, con el fin de excitar la compuerta donde al recibir señal induce una chispa en la bujía. (Todo Motos, 2013).

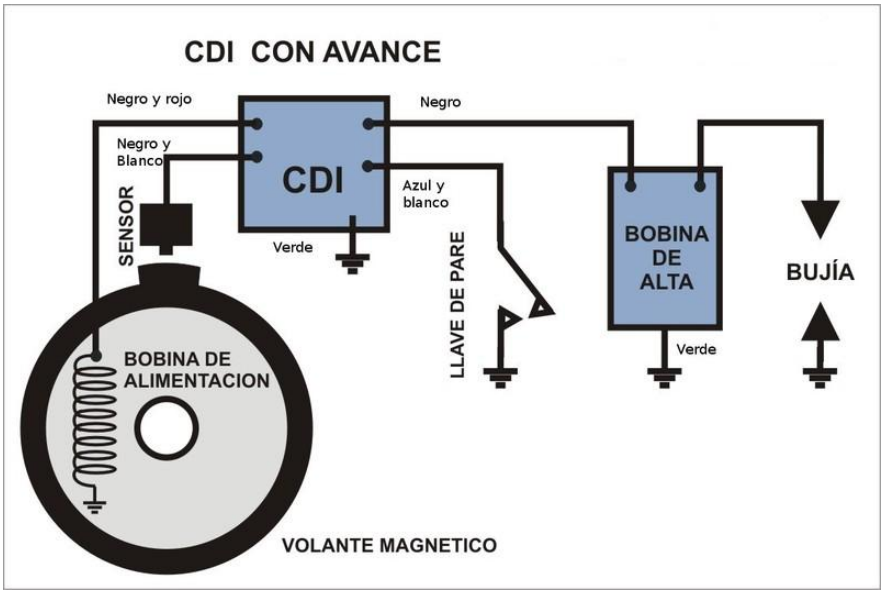

Figura 9. Estructura interna de un CDI.

Fuente página web http://www.todomotos.pe/mecanica/1345-cdi-importancia-mantenimiento-moto/

Emisor receptor de 315 MHz con 2262/2272

Posee integrados 2262 y 2272 de tecnología CMOS, tipo Encoder / Decoder, pensados para formar una comunicación sencilla entre dos puntos en modo unidireccional y tienen la claridad de ser sencillos y baratos. Se utilizan desde mandos a distancia de infrarrojos hasta pequeños módulos inalámbricos para puertas de garaje y similares.

Son básicamente un emisor de 4 canales con formato llavero, con antena retráctil, y un receptor basado en nuestro chip 2272, que activa uno de 4 pines en función del botón que se pulse en el mando remoto. Las conexiones son ligeras, donde se conecta la tensión y Ground. Después, cuando se pulsa un botón el mando remoto, si la comunicación es correcta levanta el pin VT para indicar que ha detectado una transmisión valida (Valid Trans). Por otra parte, los pines D0, D1, D2, D3 se activan con un HIGH, cuando se pulsa el botón A, B, C, D respectivamente. (Prometec.Net).

Figura 10. Emisor y receptor RF.

Fuente página web: http://www.prometec.net/control-remoto-rf/

Modulo sensor de vibración

Este módulo cuenta con un sensor que reacciona ante movimientos fuertes, impactos o vibraciones externas pero no a movimiento constantes. Al detectar una vibración, este genera una señal ya sea digital o análoga el cual finaliza al no detectar vibración.

El principio de este módulo se representa con dos contactos, los cuales están unidos en una varilla metálica dentro de un espiral que posee un comportamiento de resorte. Con los movimientos bruscos, el resorte se desplaza de un lado a otro, estrellándose con la varilla del centro y generando la señal eléctrica. Por otro lado un amplificador operacional, toma este valor y lo compara con el voltaje de referencia del potenciómetro, que actúa en la sensibilidad del sensor. (Dealextreme, 2012).

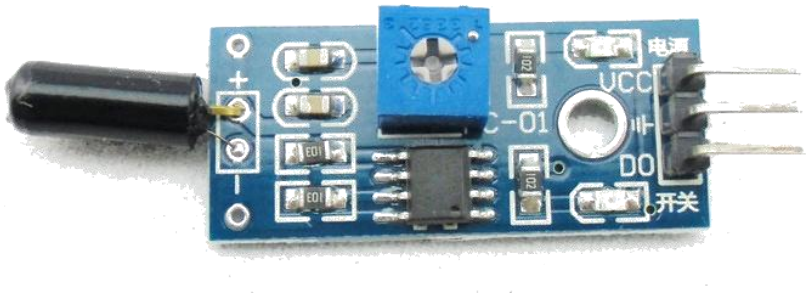

Figura 11. Modulo sensor de vibración.

Fuente página web: http://www.dx.com/es/p/vibration-sensor-alarming-module-for-arduino-150728#.V0T0wvnhC1s

Módulo de cuatro relés

Es un módulo que posee cuatro relevos con una bobina de excitación de 5 voltios, ya sea directamente del Arduino o por alguna fuente externa. Este módulo viene con optoacopladores para cada uno de los relés, con el fin de aislar ruidos que puedan afectar el funcionamiento normal del Arduino. Aparte cuenta con luces led que indican cuando el relé está activo.

El módulo actúa por medio de señales tipo Sink (o señales de voltaje negativo); de esta forma de garantiza de que el dispositivo se encuentra aislado de ruido. Cada relevo puede soportar hasta 10 Amperios a 250 Voltios lo cual es idear para proyecto de On/Off que involucren voltajes y corrientes moderadas. (Tolocka, 2015).

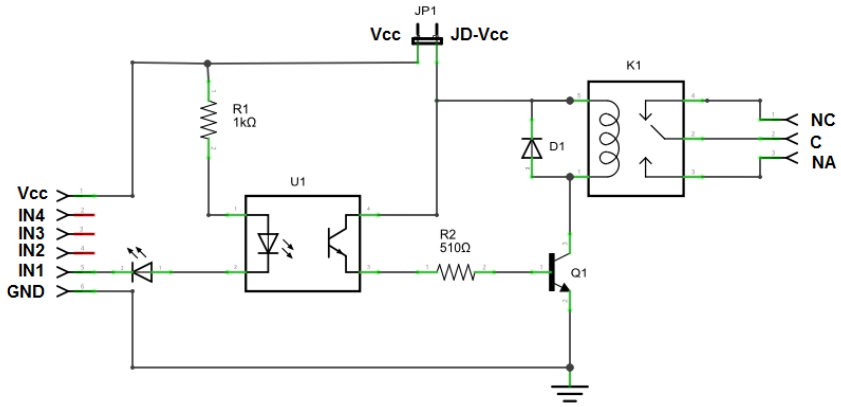

Figura 12. Esquemático de cada relé del módulo.

Fuente página web: http://www.profetolocka.com.ar/2015/05/09/modulo-de-4-reles-para-arduino/

El principio de funcionamiento del módulo es simple, el pin de entrada IN1 está conectado al cátodo del led correspondiente al optoacoplador y en serie con el led de indicación. El ánodo del optoacoplador está conectado a Vcc. El transistor del optoacoplador, se encuentra con el colector y uno de los terminales del relé está conectado a Vcc por medio de un Jumper. Este transistor se encuentra conectado en forma de emisor común, Con el fin de colocar un nivel bajo en IN1 el transistor entra en saturación y a su vez se activa el relé. (Tolocka, 2015).

Figura 13. Módulo de cuatro relés.

Fuente página Web: http://www.profetolocka.com.ar/2015/05/09/modulo-de-4-reles-para-arduino

Proceso metodológico

Hardware

Etapa de control. El sistema de control lógico, se desarrolla por medio del Arduino Mega ADK, encargado principalmente de la recolección de datos y control de entradas y salidas del dispositivo. Este componente se relacionara con los Shield y módulos por medio de los puertos seriales con los cuales cuenta. De ahí, se comunicara con el Shield GPRS SM5100b, el Bluetooth HC-06 y el Modulo Neo 6m Ublox.

El Shield GPRS, cuenta con una antena para poder obtener una mejor recepción de datos cuando se conecta por medio de la tecnología GSM. Este Shield se encargara de recibir y enviar mensajes de texto hacia nuestro dispositivo móvil por medio de una SIM-Card, la cual necesita un servicio de mensajería de texto habilitada para su uso.

El módulo HC-06 recibirá datos directos desde nuestra aplicación para sistema operativo Android, utilizando tecnología Bluetooth. El trabajo de este módulo es de recibir los comandos básicos que se pueda efectuar como mínimo a una distancia de 10 metros, sin generar ningún tipo de costo por el uso del dispositivo. (Jesus, 2014).

El módulo Neo 6m Ublox, será nuestro GPS, el cual estará alimentando al Arduino de una serie de datos donde se especifica las coordenadas en la cual se encuentra, con un rango promedio de error de 5 a 15 mts. Este dispositivo se alimenta con 5v pero tiene una

buena estabilidad a la hora de mantener la interconexión entre los satélites. (Rodriguez, 2010).

Ya que el Arduino Mega ADK posee cuatro puertos seriales, facilita entrelazar las comunicaciones seriales de forma rápida y estable. Este Arduino cuenta con varias salidas de alimentación con su respectiva regulación, pero es necesario utilizar reguladores externos. Los componentes son conectados de forma apilada para tener un espacio más reducido.

Los puerto seriales se distribuirán de tal forma que; el puerto serial 0 será el encargado de la comunicación entre el Arduino y el computador para facilitar el seguimiento de cada una de las variables del programa; el puerto serial 1 comunicara el Arduino con el modulo Bluetooth; el puerto serial 2 recibirá los datos obtenidos por el modulo GPS; y por último el puerto serial 3 se encargara del envío y recepción de mensajes de texto por medio del Shield GPRS.

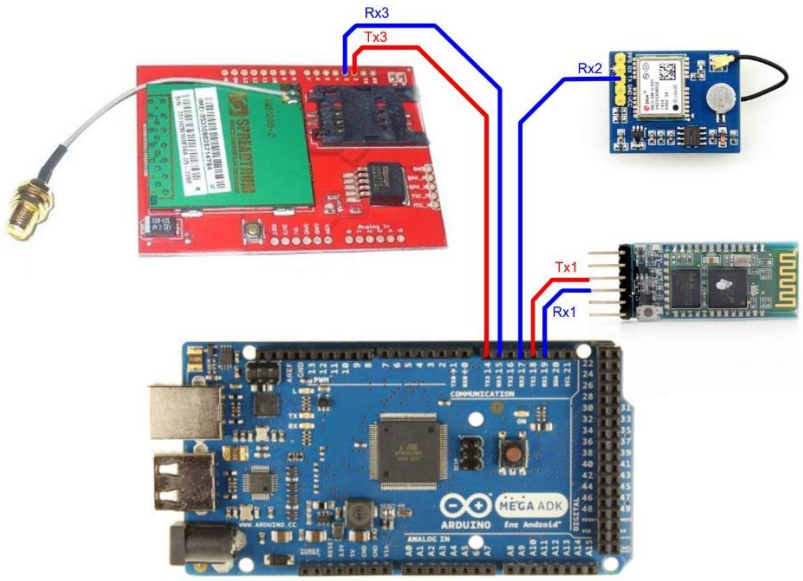

Figura 14. Comunicación del Arduino con los demás Shields y módulos.

Fuente, autor.

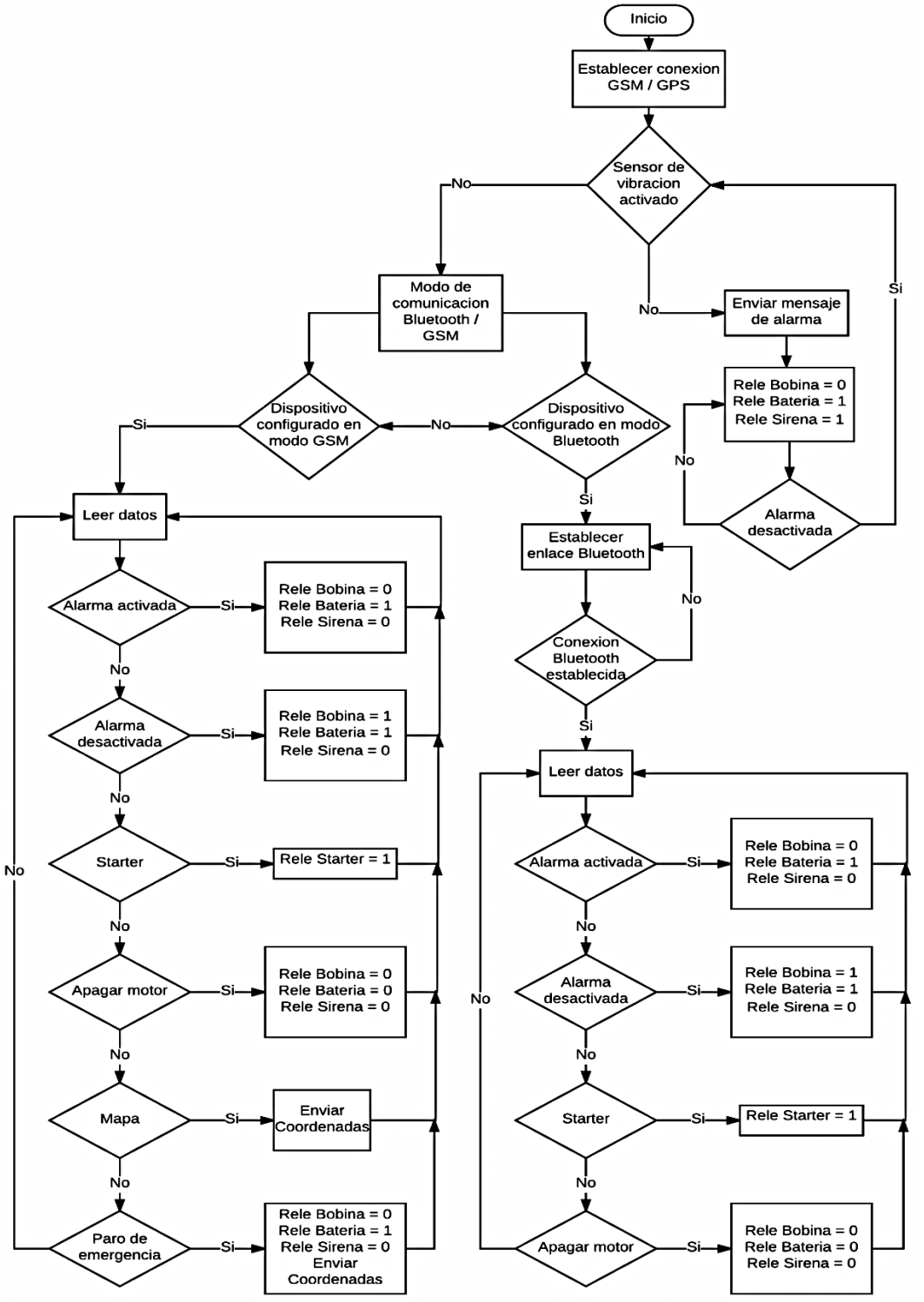

Figura 15. Proceso de Hardware

Fuente: Autor.

Etapa de potencia. Los anteriores componentes solo se encargara de la parte lógica del sistema de alarma, ya que solo manejaran valores de 5v, por tal razón hay que utilizar un circuito que ayude al Arduino a controlar los voltajes que genera la bobina de encendido de la moto por medio de relés que serán que actuaran directamente con la etapa de potencia.

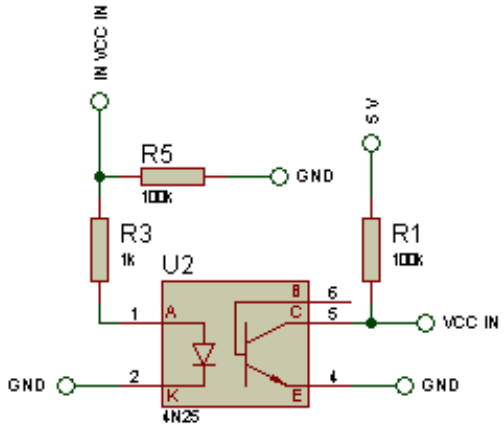

Figura 16. Diagrama eléctrico de las entradas de la etapa de potencia.

Fuente, autor.

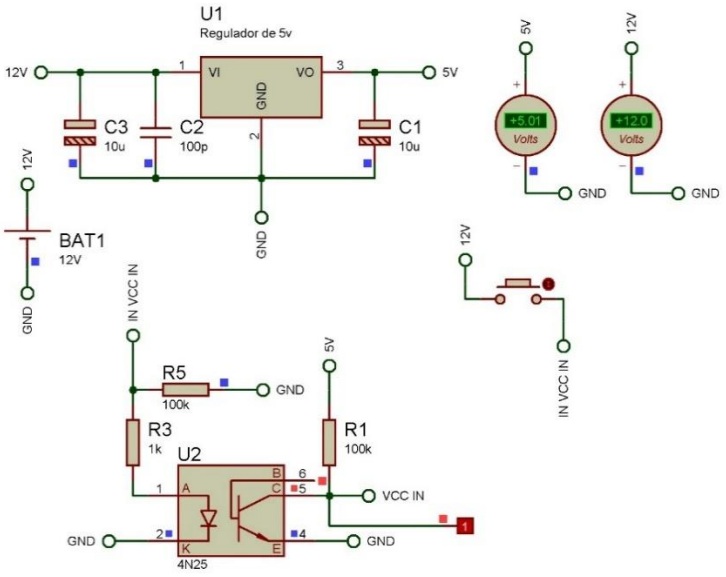

Figura 17. Simulación de entradas sin señal de excitación.

Fuente, autor.

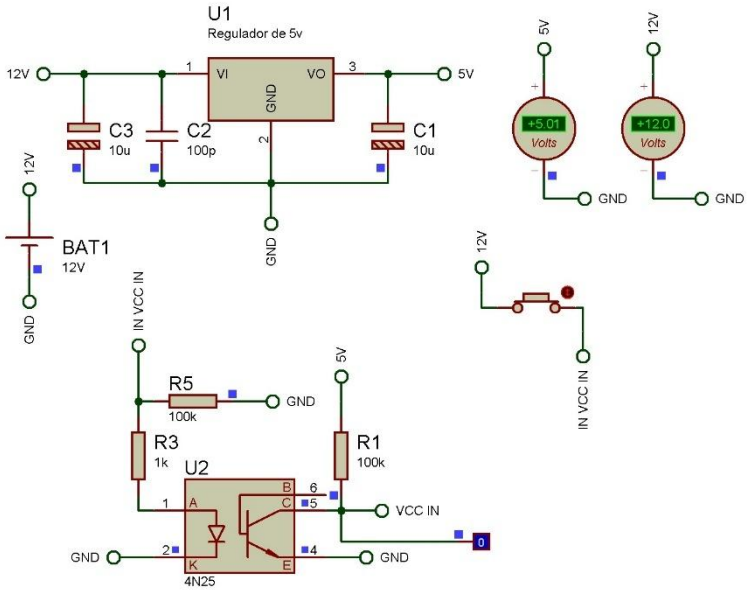

Figura 18. Simulación de entradas con señal de excitación.

Fuente, autor.

Los relevos serán funcionales a 12v, y accionados por medio de transistores de baja potencia 2n3904 y aislados por medio de optoacopladores. Igualmente las señales de entrada de 12 v, se reducirán a señales lógicas de 5 voltios por medio de optoacopladores 4n25.

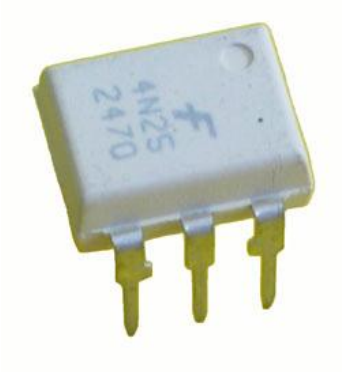

Figura 19. Optoacoplador 4n25.

Fuente, Pagina Web: http://www.futurlec.com/LED/4N25.shtml

Tabla 1. Distribución de las entradas y salidas

I/O	*Control*		*Tipo*
	Transistor	*Relé*	
Vcc	*X*		*Entrada*
Neutro	*X*		*Entrada*
Pito	*X*		*Salida*
Bobina de encendido		*X*	*Salida*
Interruptor de llave		*X*	*Salida*
Starter		*X*	*Salida*

Nota. Fuente autor.

Una vez definidas las entradas, se procede a crear el plano eléctrico donde se incluyen alimentación, regulación y entradas.

El circuito posee dos entradas, doble regulación, terminales hembra para conectar los módulos de Bluetooth, GPS y sensor de vibración, así como acoples rápidos y borneras.

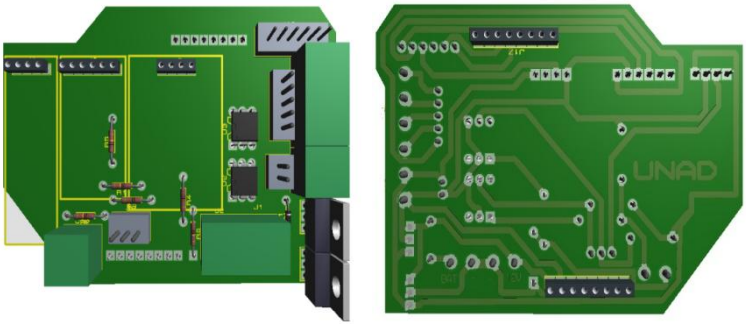

Figura 20. Layout en 3D.

Fuente Autor.

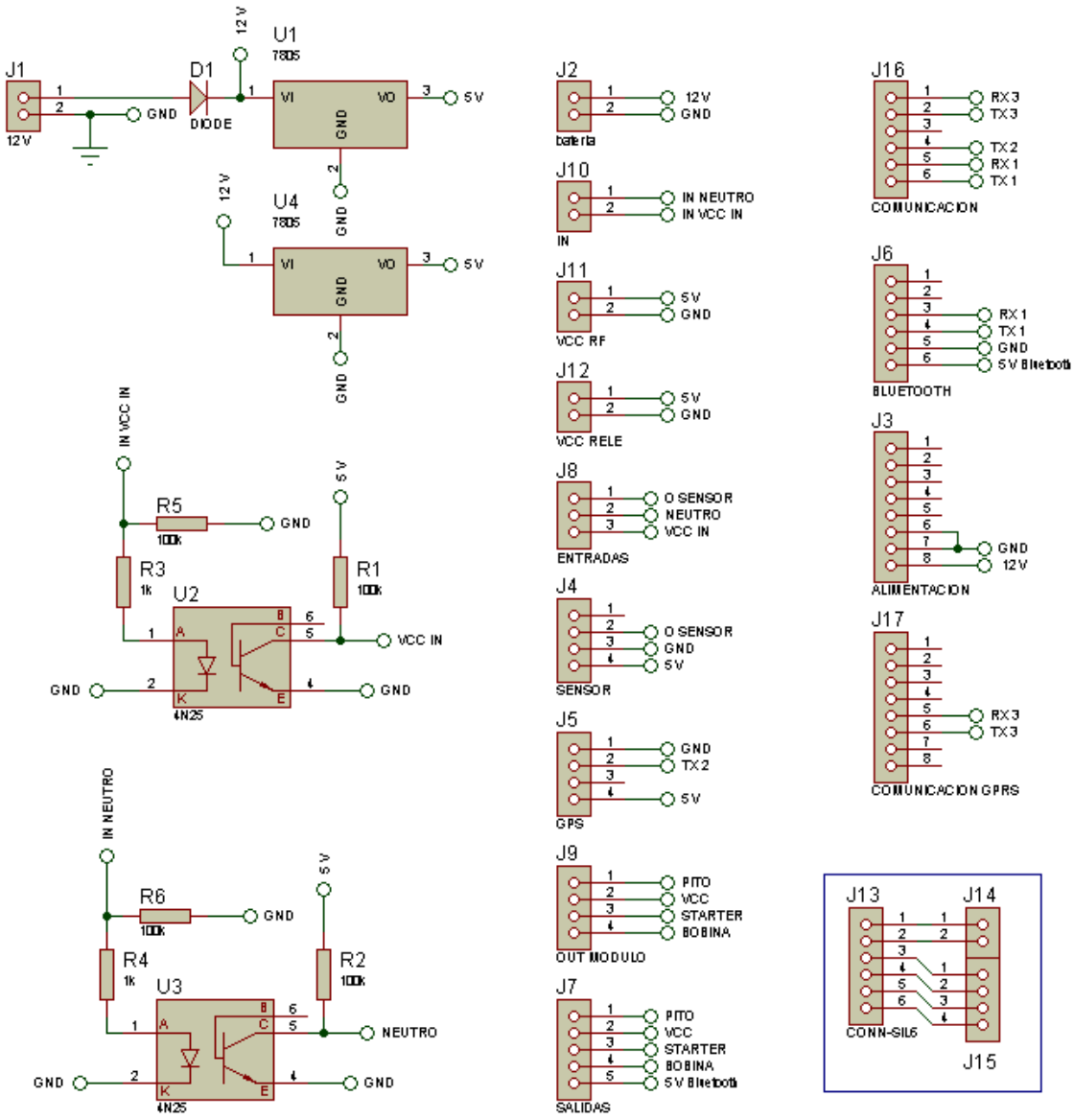

Figura 21. Plano eléctrico completo de las entradas.

Fuente autor.

Teniendo listo el modelo, se imprime el Layout y se quema en váquela donde se soldaran sus respectivos componentes electrónicos.

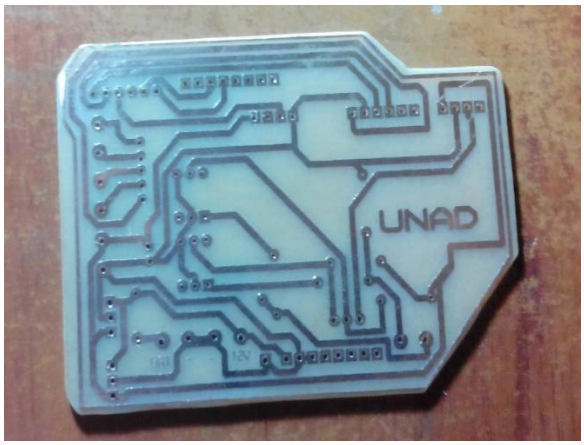

Figura 22. Circuito de entradas en físico.

Fuente, Autor.

La Conexión del sistema de potencia con el sistema eléctrico de una moto se hace de la siguiente forma:

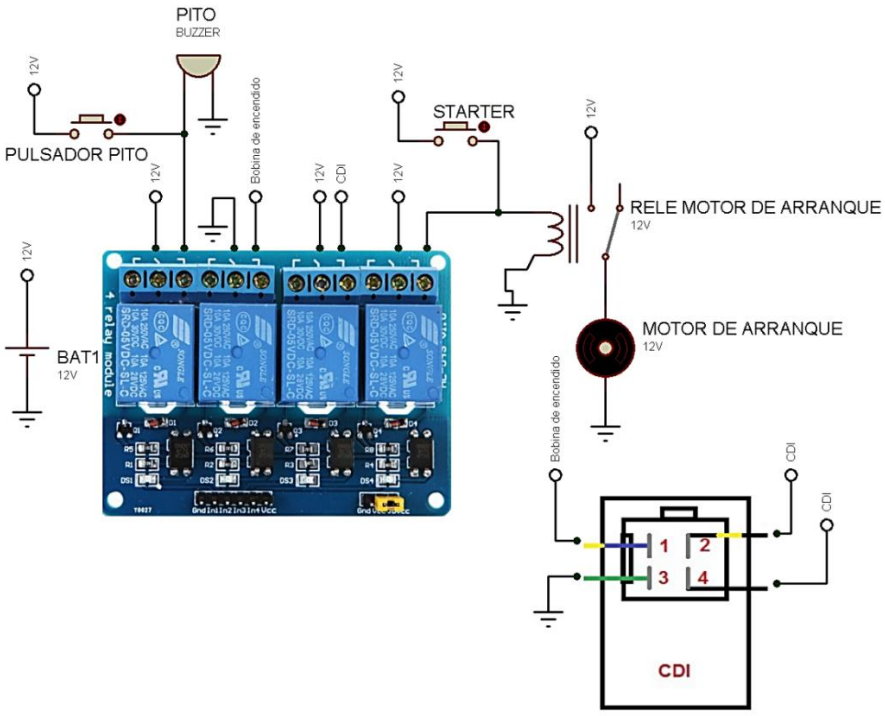

Figura 23. Distribución eléctrica de una moto.

Fuente: Autor.

Software

Aplicación. El lenguaje de programación para Android es por medio de lenguaje JAVA. La cual no es necesaria una licencia para poder crear aplicaciones para celulares o tablas, utilizando un nivel de programación por líneas de código. Sin embargo, existe otro tipo de software de programación en línea llamado Appinventor basado en diagrama de bloques. (IBM Blueix, 2014).

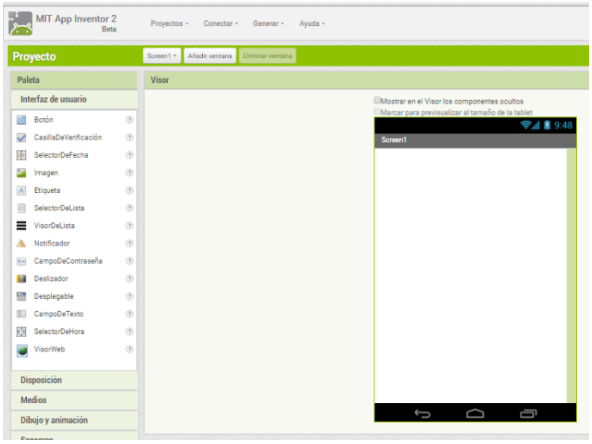

Figura 24. Ventana principal de Appinventor.

Fuente autor.

Ya que Appinventor estada dada en programación por bloques, da facilidad a la hora de hacer una aplicación APK, teniendo en cuenta que este software posee menos herramientas disponibles. Este programa en línea, no necesita licencia alguna y a pesar de su sencillez, se pueden generar una gran variedad de aplicaciones ya que cuenta con el control de muchas funciones internas del celular (llamar aplicaciones, Bluetooth, mandar textos, recibir textos, temporizadores, manipulación de sensores del dispositivo Android, etc.).

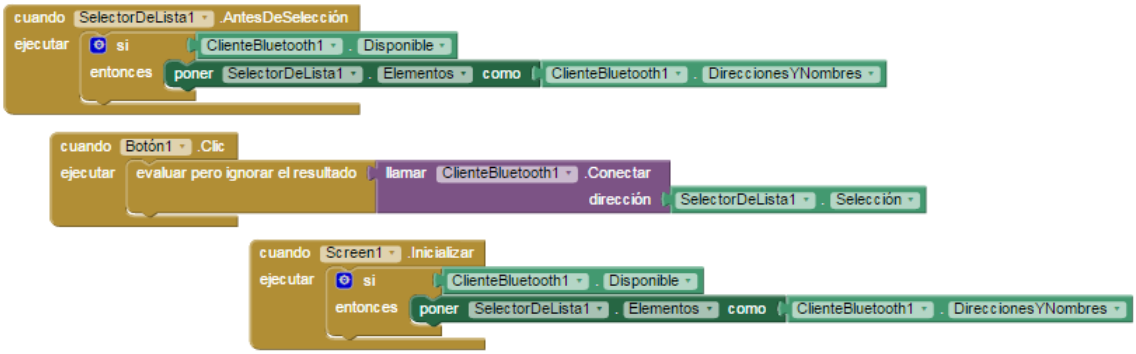

Figura 25. Diagrama de bloques de Appinventor.

Fuente Autor.

Appinventor cuenta con dos ventanas. La primera, es la ventana de la aplicación, aquí se diseñara el área de trabajo de la aplicación; botones, fondo de pantalla, comandos, diseño gráfico, etc., la segunda ventana cuenta con la programación por bloques. Esta última será la encargada de la parte lógica de la aplicación teniendo en cuenta que cantidad de componentes se agregaron en la primera ventana, definiendo así el comportamiento de la aplicación.

Pantalla principal. La aplicación cuenta con un total de 9 botones interactivos.

Tipo de conexión. Este botón se activa al dejarlo oprimido por más de tres segundos, encargándose de definir si la comunicación con el Arduino será por mensajes de texto o por enlace Bluetooth.

Estado del Bluetooth. Una vez tengamos vinculado nuestro dispositivo con el modulo Bluetooth, se escoger entre conectar o desconectar el enlace, dependiendo del

estado en la que se encuentra la interfaz, siendo definido por medio del color en el que se encuentre el icono del Bluetooth.

Estado de la alarma. Es una lista desplegable donde el usuario determinara si se activa o se desactiva la alarma y terminar con el proceso de paro de emergencia.

Starter. Este será nuestro pulso para iniciar el motor de encendido, teniendo en cuenta que la motocicleta este encendida y que esta se encuentra en neutro.

Apagar moto. Sera nuestro comando de apagado automático de la motocicleta, mandando las señales de pulso del CDI a masa y a su vez desconectado todo el sistema de la batería.

Activar mapa. Por medio de un mensaje de texto, recibirá las coordenadas generadas por el GPS para indicar la posición actual sin afectar sistema eléctrico de la motocicleta.

Paro de emergencia. Al igual que el anterior, recibirá las coordenadas del GPS pero estas coordenadas llegaran cada 30 segundos. Aparte, desactivara el sistema eléctrico de la moto.

Reiniciar. Cuando la aplicación reciba las coordenadas del GPS, inmediatamente quedan registrada en el campo de texto en la parte superior de este botón. Una vez accionado este botón borrara tal registro del programa.

Mapa. Cuando el registro de coordenadas queda guardado en la pantalla, se puede acceder manualmente a Google Maps accionando este botón.

Figura 26. Botones interactivos del APK

Fuente Autor

Bloques. Por medio de este tipo de lenguaje de programación, cada uno de los bloques, es enlazado unos con otros en forma de fichas de rompecabezas. Cada uno de los

bloques representa las variables, comando, condicionales, constantes y temporizadores que el diseñador necesita para desarrollar el APK.

Cada conjunto de bloques representa una función específica en el código.

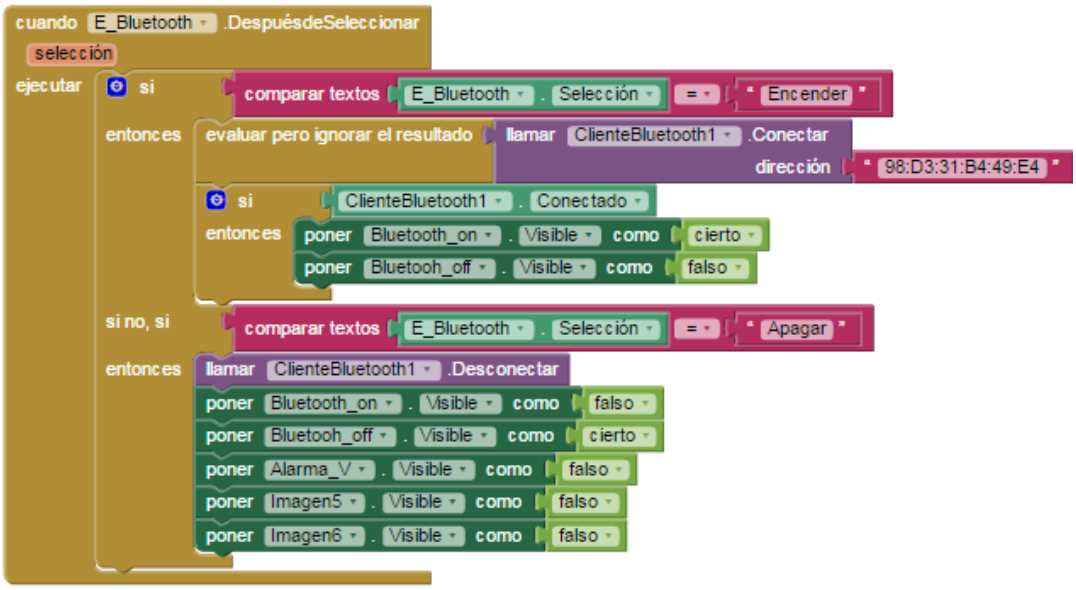

Figura 27. Conexión Bluetooth

Fuente: Autor.

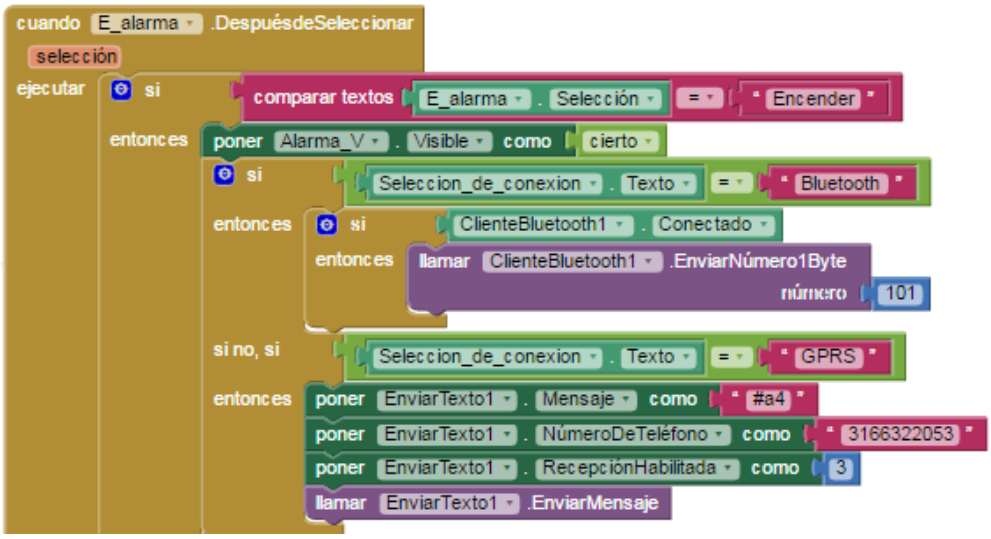

Figura 28. Comandos de Alarma Bluetooth y GSM

Fuente: Autor.

Para el análisis del diagrama de bloques completo, revisar Apéndice 1 al final del documento.

Programa Arduino. Arduino cuenta con un lenguaje de programación C++ (Doutel, 2015), fácil de manejar gracias a su extensa librería que hace más practico el código. Contiene una ventana principal donde se escribirá el código indicando primero las librerías y variables a manejar. Posterior a esto se definen los parámetros iníciales por medio del comando Void Setup, y por último se define nuestro código de ciclo infinito Void Loop. Desde este software definiremos que pines se utilizaran en nuestra aplicación, si son entradas y salidas, que puertos de comunicación se utilizaran, así como la secuencia lógica por medio de condicionales.

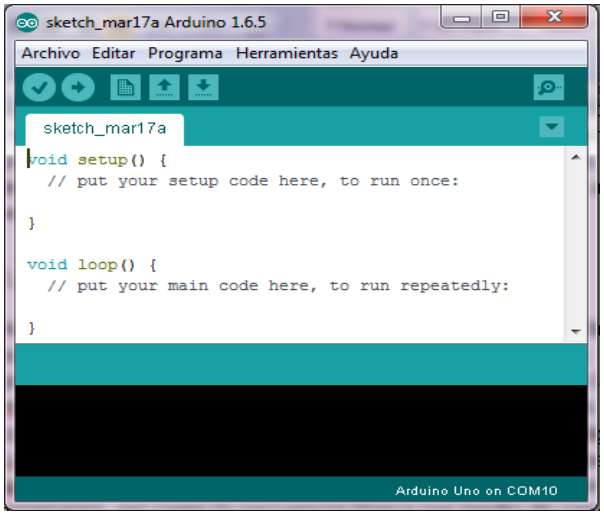

Figura 29. Ventana principal de Arduino.

Fuente autor.

Para la programación de nuestro ADK, se debe recurrir a dos librerías externas para facilitar el análisis de datos obtenidos en nuestros puertos seriales. Una de estas librerías es

WString.h, encargada de almacenar y analizar cadenas de caracteres dadas por el Shield SM5100B; la segunda librería se llama TinyGPS.h, la cual es ideal para extraer solo la latitud y longitud de nuestro GPS.

Diagrama de bloques. Los datos serán trasmitidos en una serie de códigos alfanuméricos los cuales el programa reconoce y toma la respectiva acción.

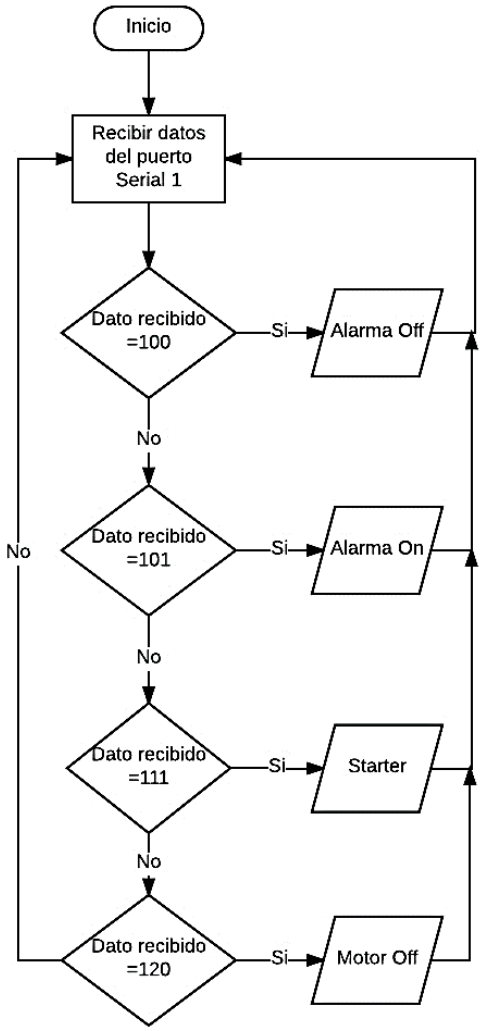

Figura 30. Comunicación Bluetooth

Fuente. Autor.

La imagen 30, comprende que valor debe ingresar por el puerto serial 1 del Arduino donde se conectara el modulo Bluetooth estableciendo cuatro comandos por medio de un valor numérico. Apagar vehículo y enviar coordenadas cada 45 segundos por mensaje de texto.

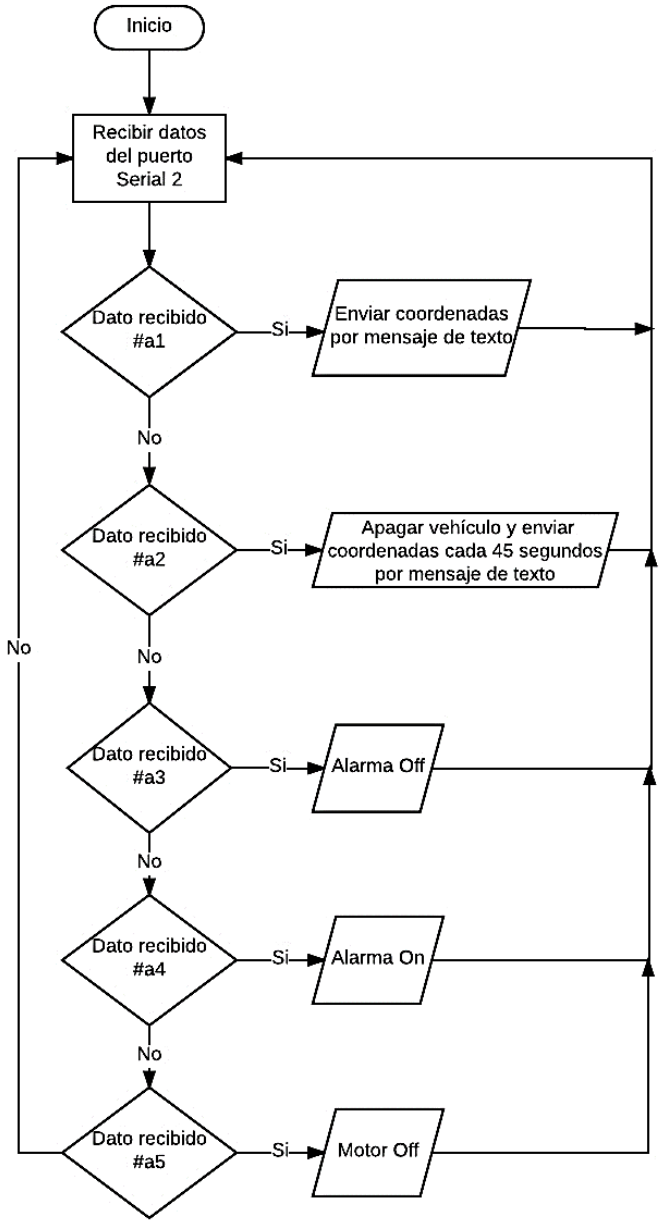

Figura 31. Comunicación GSM

Fuente. Autor.

Por medio del puerto serial 2, ingresara el cuerpo de los mensajes de texto, ya que en este puerto se comunica el módulo GSM/GPRS, siendo comparado con códigos almacenados en el Arduino para para definir su función en el dispositivo.

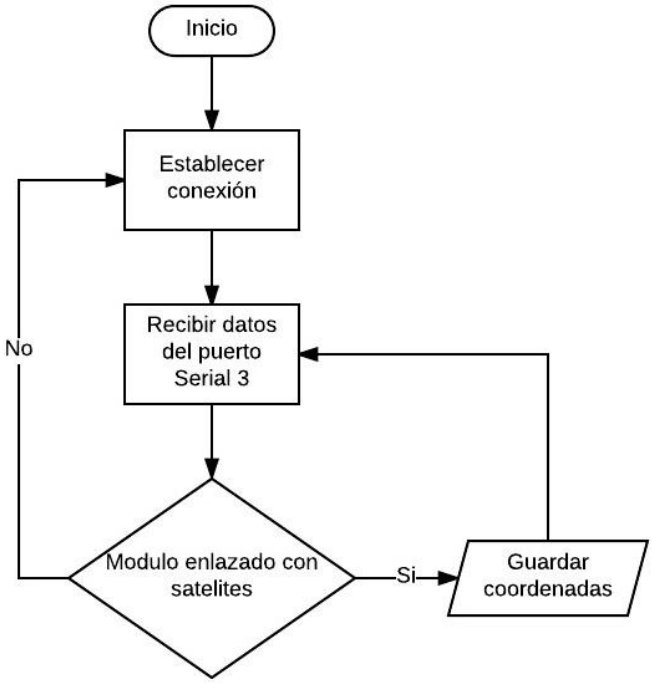

Figura 32. Comunicación GPS

Fuente. Autor.

Los datos ingresados en el puerto serial 3, son de fácil proceso, puesto que el modulo GPS es independiente de todo el sistema. Una vez se enlaza con los satélites, este empieza a enviar por su Rx del puerto serial, las coordenadas al dispositivo Arduino, así este no los esté exigiendo.

Código Arduino. En la estructuración del código en lenguaje de programa se inicia con la definición de variables, librerías y condiciones iniciales del algoritmo.

```
#include <WString.h>
#include <TinyGPS.h>
#define GPSBAUD 9600
TinyGPS gps;
void getgps(TinyGPS &gps);

byte neutro=36, vccin=38, sensor=34
byte alarma, inicio, modotex
int bluetooth, clave[]={2,2,0,2},
unsigned long timer1=0,
float lati, longi,
char phone_no[]="          ";
String cad;
```

Figura 33. Variables iniciales

Fuente. Autor.

Una vez definidas, el código configura Los puertos seriales y establece la conexión GSM, configurando el modulo en modo texto.

```
Serial.begin(57600);
Serial1.begin(9600);
Serial2.begin(9600);
Serial3.begin(115200);


Serial3.println("AT+CMGF=1");
delayMicroseconds(10000);
Serial3.println("AT+CNMI=2,2,0,0");
delayMicroseconds(10000);
Serial3.println("AT+CMGD=1,4");
delayMicroseconds(10000);
```

```
while (Serial1.available() && manual==0){
                                        bluetooth=Serial1.read();
                                       }


while ( Serial2.available() && manual==0){
                                         int bt =  Serial2.read();
                                         if (gps.encode(bt)){
                                                             getgps(gps);
                                                            }
                                        }

while (Serial3.available() && manual==0) {
                                         char c = Serial3.read();
                                         cad += c;
                                         delayMicroseconds(16000);
                                         Serial.print(cad);
                                         }
```

Figura 34. Configuración de puertos.

Fuente. Autor.

Los comandos que manejan los dos tipos de comunicaciones son comparados y a su vez independiente que tipo de comunicación se utilizó, hará la respectiva acción en el algoritmo.

```
if (bluetooth == 100){
    alarma=0;
    paro_de_emergencia=0;
    a_automatica=1;
    sirena=0;
    estado_alarma=1;
    digitalWrite(pito,HIGH);
    digitalWrite(motor,LOW);
    digitalWrite(vcc,LOW);
    Serial.println("Alarma off");
    for(int i=0;i<3;i++){
                         digitalWrite(pito,LOW);
                         delay(50);
                         digitalWrite(pito,HIGH);
                         delay(150);
                        }
    bluetooth=0;
   }
```

```
if (cad.lastIndexOf("#a5") > -1){
    alarma=0;
    paro_de_emergencia=0;
    sirena=0;
    a_automatica=1;
    estado_alarma=1;
    digitalWrite(pito,HIGH);
    digitalWrite(motor,LOW);
    digitalWrite(vcc,LOW);
    Serial.println("Alarma off");
    for(int i=0;i<3;i++){
                         digitalWrite(pito,LOW);
                         delay(50);
                         digitalWrite(pito,HIGH);
                         delay(150);
                        }
    Serial3.println("AT+CMGF=1");
    delay(100);
    Serial3.print("AT+CMGS=");
    Serial3.print((char)34);
    Serial3.print(phone_no);
    Serial3.println((char)34);
    delay(100);
    Serial3.print("La alarma ha sido desactivada");
    Serial3.print((char)26);
    timer6=reloj;
```

Figura 35. Comparación de los comandos Alarma Off.

Fuente. Autor.

Para facilitar ciertas líneas de códigos, es necesario la utilización de bibliotecas que nos ayuden a desencriptar datos para poder extraer lo necesario. Un claro ejemplo son los datos que entrega nuestro módulo de GPS, los cuales pueden ser de fácil acceso por medio de la biblioteca TinyGPS, ayudando a extraer solo la longitud y la latitud de dicho módulo.

```
void getgps(TinyGPS &gps)

{
 float latitude, longitude;
 gps.f_get_position(&latitude, &longitude);
 lati=latitude;
 longi=longitude;
 Serial.print("Lat/Long: ");
 Serial.print(lati,5);
 Serial.print(", ");
 Serial.println(longi,5);

}
```

Figura 36. Biblioteca TinyGPS.

Fuente. Autor.

Para el control de acceso manual, el código guarda por medio de una matriz una serie de combinaciones que se compara con una configuración predeterminada, cambiando el estado de una variable que se encargara de inhabilitar el resto del algoritmo.

```
if((sec[0]==clave[0])&&(sec[1]==clave[1])&&(sec[2]==clave[2])&&(sec[3]==clave[3])){
   manual=1;
   sirena=0;
   paro_de_emergencia=0;
   alarma_automatica=0;
   alarma=0;
   estado_alarma=0;
   digitalWrite(pito,HIGH);
   digitalWrite(motor,LOW);
   digitalWrite(vcc,LOW);
   Serial.print("Sistemas desactivados");
   for(int i=0;i<4;i++){
                        digitalWrite(pito,LOW);
                        delay(50);
                        digitalWrite(pito,HIGH);
 }                      delay(150);
 else {
       Serial.print("Clave Erronea");
       for(int i=0;i<1;i++){
                             digitalWrite(pito,LOW);
                             delay(50);
                             digitalWrite(pito,HIGH);
                             delay(150);
                            }
     }
```

Figura 37. Control manual.

Fuente. Autor.

Para el análisis completo del código en Arduino, diríjase al apéndice 2.

Desarrollo y pruebas del proyecto

Para las pruebas del dispositivo, no es necesario el montaje de una motocicleta, si se tiene en cuenta el principio del sistema eléctrico de un motor de combustión interna. Tal como se ilustra en la figura 23, la conexión principal del dispositivo se hará en el CDI de la motocicleta, direccionando la señal de la bobina de encendido al chasis del vehículo. De esta forma, no puede encenderse el motor.

Las señales de Sirena, Batería y Starter, se conecta en paralelo a las conexiones ya establecida en el sistema eléctrico de la motocicleta. Teniendo en claro estas conexiones, todo el sistema se puede simular con un tablero de leds cada una de las conexiones que se harán en la motocicleta.

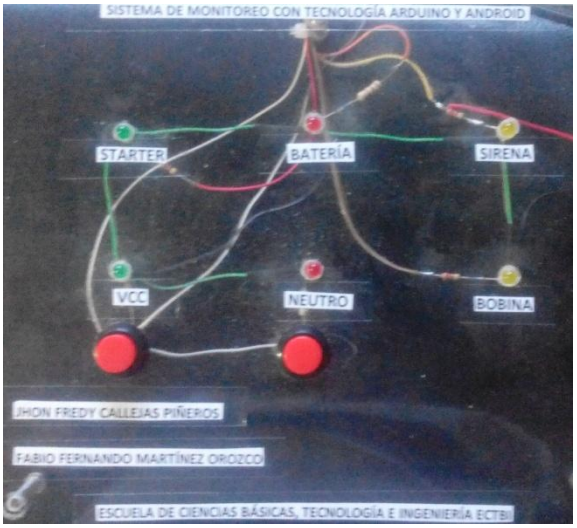

Figura 38. Tablero de simulación.

Fuente. Autor.

Pruebas iniciales.

Las conexiones establecidas en nuestro Arduino quedan de la siguiente manera:

Tabla 2. Distribución de pines.

Arduino Mega ADK		
I/O Digital Arduino	Pin Componente	Componente
Tx1	Rx	Bluetooth
Rx1	Tx	
39	Vcc	
Rx2	Tx	GPS
Tx3	Rx	GPRS/GSM
Rx3	Tx	
34	D0	Sensor Vibración
36	5	Optoacoplador 1
38	5	Optoacoplador 2
39	VT	Receptor RF
41	D3	
43	D2	
45	D1	
47	D0	

Nota. Fuente Autor

Los pines de alimentación del Arduino estarán conectados a una batería de respaldo para garantizar de que el dispositivo se mantenga activo, así, hallan desconectado la batería principal de la motocicleta. Esta batería se recargara paralelamente a la batería de la moto por medio de la bobina de luces que se encuentra en el volante de la moto, sin que haya retorno de corriente gracias a un diodo rectificador ubicado en la entrada de alimentación de nuestro dispositivo.

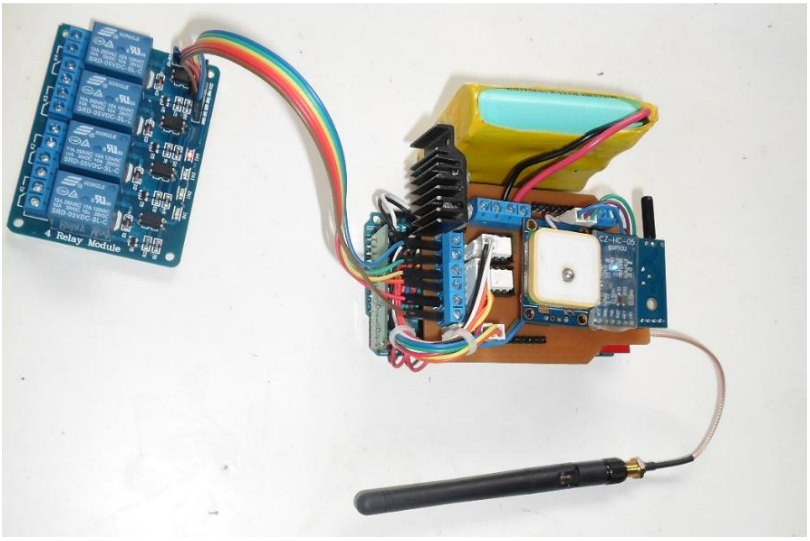

Figura 39. Circuito de control y potencia terminado.

Fuente. Autor

Simulación del dispositivo

Una vez enlazado nuestro dispositivo Bluetooth con el celular, verificamos la comunicación por medio del cambio de color de nuestro símbolo Bluetooth.

Figura 40. Estado de conexión del Bluetooth

Fuente. Autor.

Los comandos de encender y apagar alarma, apagar moto y starter se simularon con un tablero de led donde indica el accionamiento los relés, así como el comportamiento de nuestro dispositivo al detectar una señal del Vcc de nuestra moto o de la señal de neutro.

Tabla 3. Estado lógico de las I/O del dispositivo.

Comandos	Entradas		Salidas			
	Vcc	Neutro	Sirena	Motor	Batería	Motor de encendido
Activar alarma	0	x	0	0	1	0
Desactivar Alarma	x	x	0	1	1	0
Starter	1	1	0	1	1	1
Apagar moto	x	x	0	0	0	0
Paro de emergencia	x	x	1	0	1	0

Nota. Fuente Autor

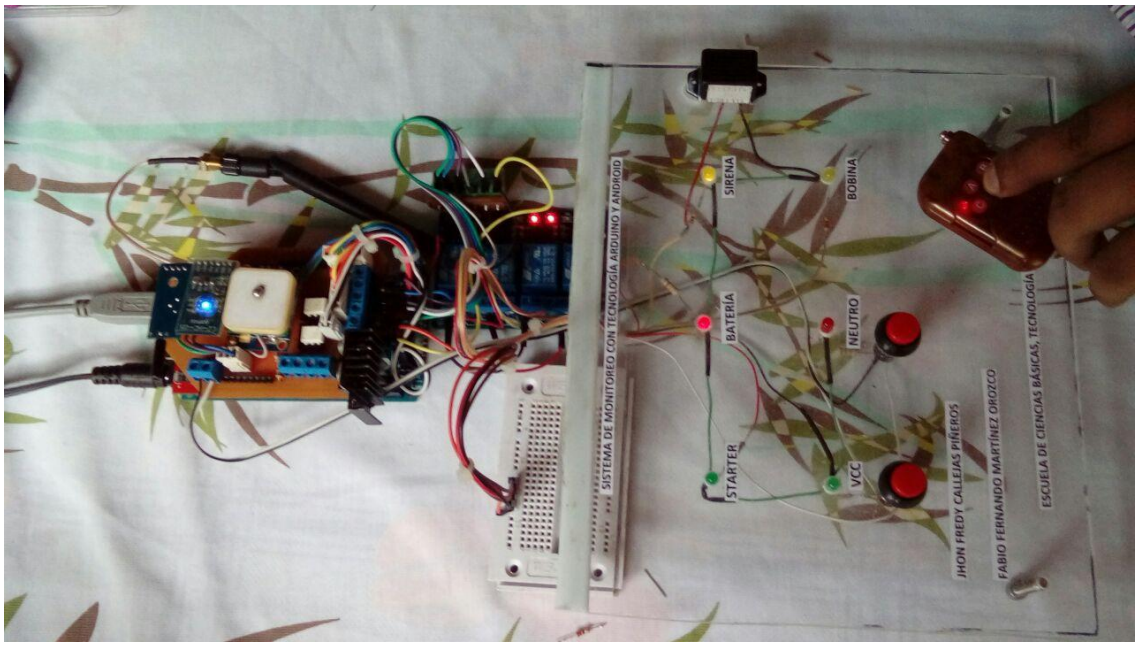

Figura 41. Diseño completo para la simulación.

Fuente. Autor.

Figura 42. Estado de reposo del dispositivo.

Fuente. Autor

En esta imagen representa el sistema cuando está en reposo, evidenciando que solo está habilitado la batería, con el fin de que haya energía a la hora de activar la sirena en caso de ser accionada.

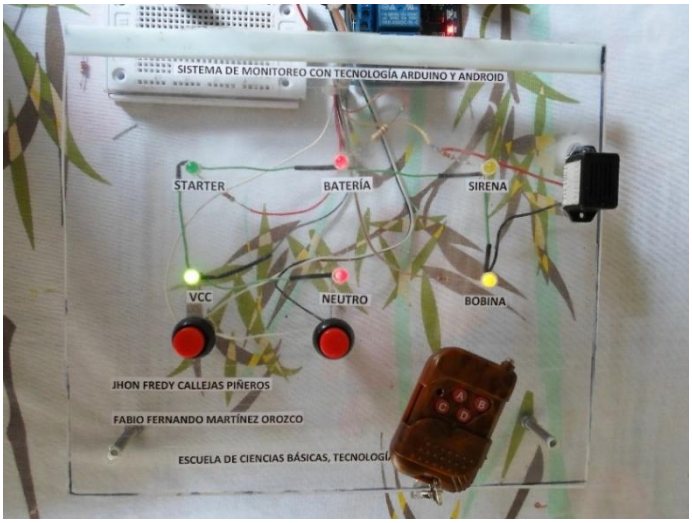

Figura 43. Sistemas habilitados.

Fuente. Autor.

En este caso, la batería y la señal de la bobina de encendido están habilitadas para el funcionamiento normal de la moto.

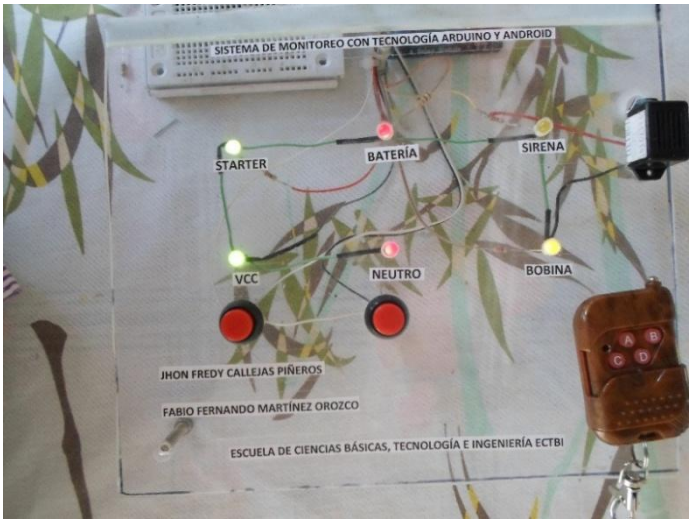

Figura 44. Accionamiento del Starter.

Fuente. Autor.

A la hora de ser accionado el botón del Starter, este solo se activara cuando Vcc y Neutro están encendidos, con el fin de que el motor de arranque solo funcione cuando la caja de la motocicleta está en neutro.

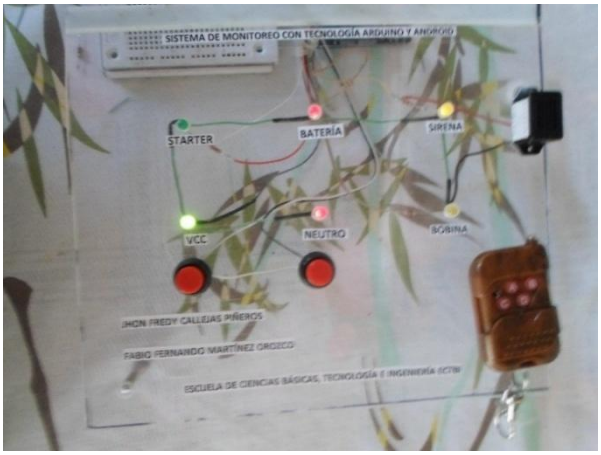

Figura 45. Accionamiento de la alarma

Fuente. Autor.

Esta salida es activada cuando el sistema tiene habilitada la alarma y el sensor de vibración es accionado. Cuando se ejecuta la alarma inmediatamente se activa la sirena y desactiva la bobina de encendido, con el fin de que la moto no pueda se encendida.

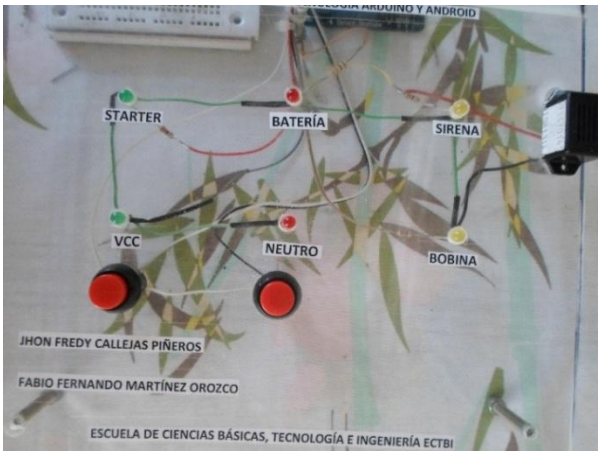

Figura 46. Accionamiento del botón" Apagar moto".

Fuente. Autor.

En esta imagen se evidencia que todos los sistemas eléctricos de la motocicleta son desactivados al ser accionado el botón “Apagar moto”.

Pruebas del GPS

Para la obtención de las coordenadas del sistema GPS, el módulo de GPRS/GSM debe tener una SIM-Card con servicio de mensajería de texto. De esta forma al ser accionado el botón “Activación de mapa” o “Paro de emergencia”, nuestra aplicación envía un mensaje de texto solicitando las coordenadas. El Arduino extrae las coordenadas del módulo GPS que a su vez, envía las coordenadas por medio de un mensaje de texto hacia nuestro celular.

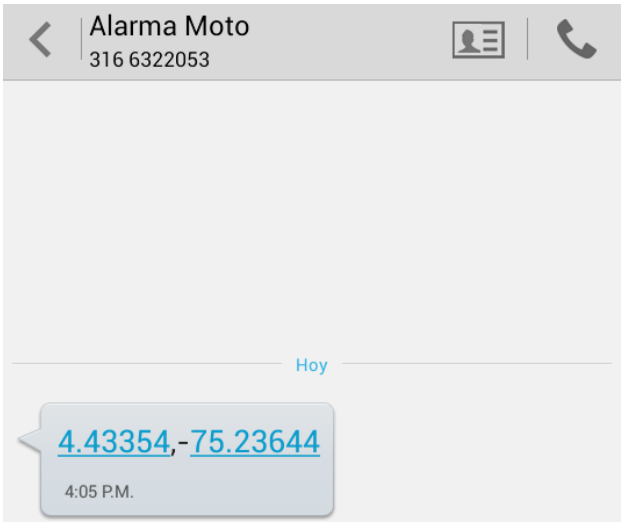

Figura 47. Recepción de coordenadas por mensaje de texto.

Fuente. Autor.

Una vez obtenido las coordenadas, el APK abre automáticamente Google Maps y posiciona automáticamente esta coordenada indicando la posición de nuestra moto.

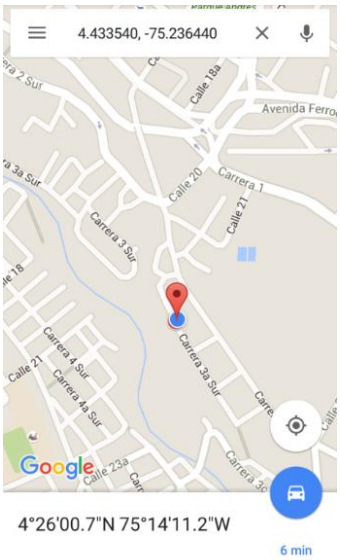

Figura 48. Ubicación de las coordenadas obtenidas por nuestro dispositivo.

Fuente. Autor.

Para las pruebas se hacen varias pruebas de localización del dispositivo, siendo llevado por un periodo de tiempo de 7 minutos oprimiendo el botón “activación de mapa”.

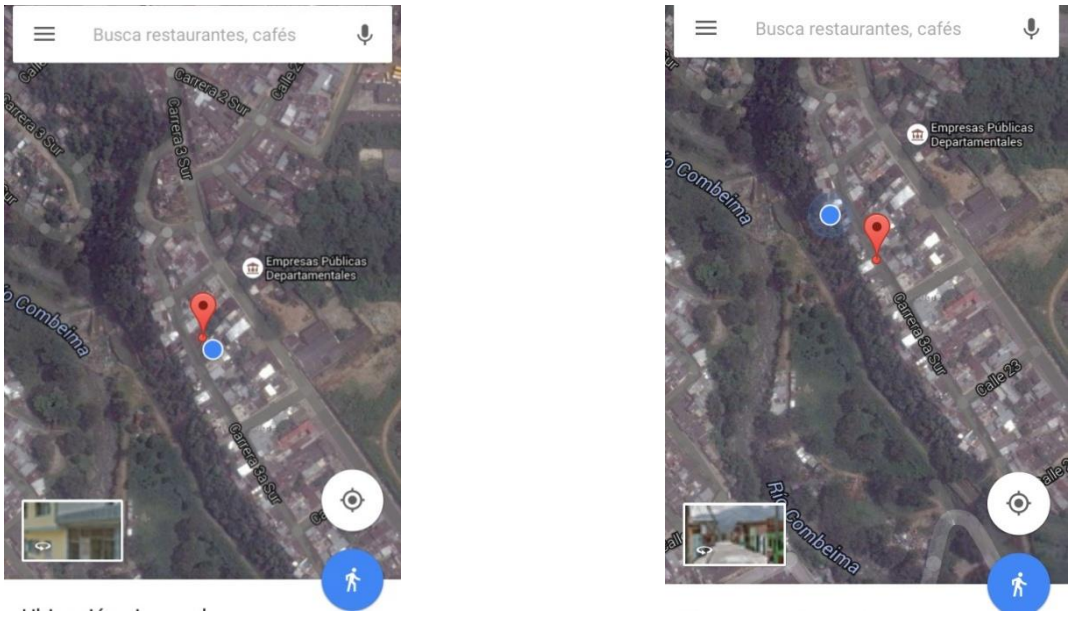

Figura 49. Ubicación del dispositivo inicial y después de 30 segundos caminando.

Fuente. Autor.

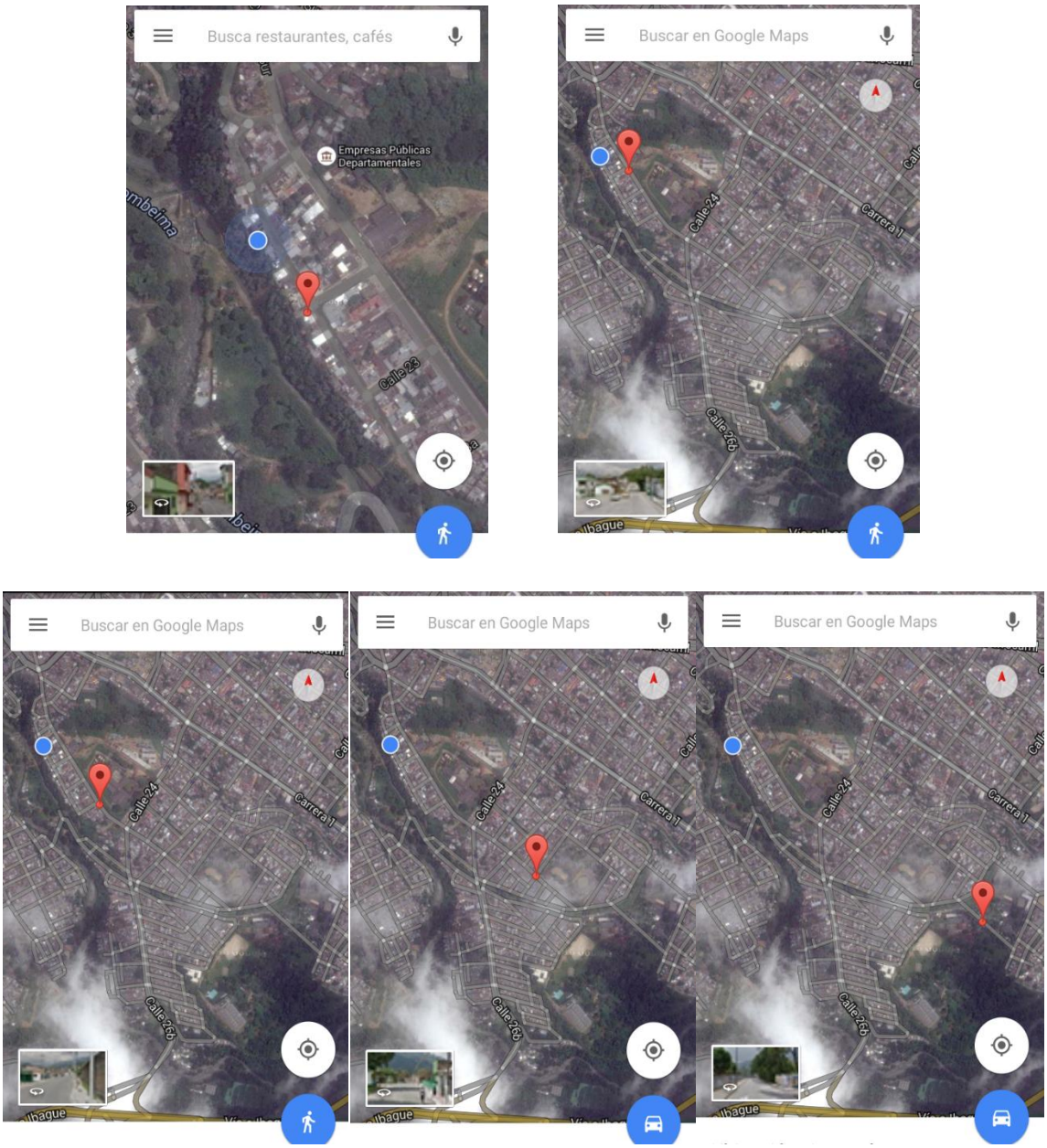

Figura 50. Dispositivo después de caminar 7 minutos.

Fuente. Autor.

En esta prueba, se visualiza como las coordenadas obtenidas, no tienden a ser lineales a medida que pasa el tiempo, esto se debe a que los mensajes de texto no son de respuesta inmediata. Por lo tanto, habrá mensajes que lleguen a 5 segundos después de ser

enviados del dispositivo, así, como mensajes que lleguen a los 20 segundos. Este tiempo tiende a variar dependiendo de la disposición de red en la cual se encuentra el servicio de comunicación GSM.

Para analizar la precisión del dispositivo, se hacen dos lecturas consecutivas del dispositivo en el mismo punto, obteniendo dos coordenadas diferentes. Dichas coordenadas son medidas a través de google Maps. Una vez ingresada las dos coordenadas y midiendo la distancia de una con la otra, se obtiene una variación en ambientes abiertos de 5 metros Aproximadamente.

Figura 51. Precisión del dispositivo.

Fuente. Autor.

El manual del usuario de esta aplicación para celulares Android, se encuentra al final del documento en el apéndice 3.

Limitantes y recomendaciones del sistema

Al utilizar mecanismos convencionales, existen varias limitantes del dispositivo que afectan su funcionalidad normal. Las recomendaciones y limitantes son las siguientes:

- La antena GPS no funciona en sitios muy cerrados, por lo tanto el dispositivo enviara la última coordenada obtenida antes de perder el enlace con los satélites.
- Para las ciudades que se encuentren cerca de la frontera con otros países, no podrán operar la alarma por medio GSM si el vehículo ha pasado la frontera, ya que esta comunicación no cuenta con función Roaming y solo funciona en redes nacionales. Esta limitante estará hasta que haya un convenio mercantil entre empresas prestadoras de servicio GSM entre otros países.
- Al activarse la alarma, el mensaje de confirmación no llega de forma instantánea, puesto que la velocidad de recepción del mensaje depende de la disponibilidad de la red en la que se encuentre el operador GSM que posea.
- La función GSM depende de un saldo en la SimCard de la alarma. Si esta se pasa del tiempo de caducidad o se acaba el saldo para mensajes, no se podrá recibir datos por parte del módulo GSM.

Conclusiones

○ Se evidencio que el funcionamiento del software en línea AppInventor, es apto para el desarrollo de programas que trabajen con manejo de datos a distancia, ya sea por Bluetooth o GSM, contando con una gran cantidad de herramientas que posee un celular. (Cámara, sensor de inclinación, almacenamiento interno, reproductor de imágenes y video, entre otras.).

○ Los resultados de las pruebas hechas con el GPS Neo 6m Ublox, mostro que posee un rango de error de 5 metros dependiendo de la cantidad de satélites que se encuentran enlazados con el modulo GPS. Pero este enlace se ve afectado en zonas cerradas como por ejemplo sótanos o garajes cubiertos, como los de un centro comercial.

○ Las pruebas obtenidas de alcance para el Bluetooth y RF, comprueban que tienen un excelente rendimiento en cuanto a transmisión y recepción de datos, manejando una comunicación de comandos rápidos con nuestra motocicleta con un alcance de trasmisión de 10 metros para el Bluetooth y unos 55 metros de alcance con el llavero RF, teniendo en cuenta que el entorno sea libre de obstáculos como paredes o vitrinas.

○ Se comprobó que el módulo GPRS/GSM tiene una comunicación efectiva con el teléfono celular a la hora de recibir y transmitir los mensajes de texto, los cuales accionan el dispositivo y de acuerdo al tipo de mensaje ejecuta la operación asignada. Sin embargo en las pruebas realizadas, la comunicación entre este dispositivo y el teléfono celular,

tiende a ser lenta en comparación con los módulos Bluetooth y RF, ya que estos dispositivos trabajan punto a punto, mientras que el módulo GPRS y el teléfono celular deben pasar su información por las repetidoras bidireccionales y demás equipos de comunicaciones que retardan la entrega de la información desde un dispositivo a otro.

○ Los resultados demostraron que Arduino puede ser una excelente plataforma para desarrollar diversidad de prototipos electrónicos, gracias a su amplia gama de componentes que se desarrollan bajo esta plataforma, así como componentes externos comunes, dejando atrás el concepto de ser una herramienta estudiantil.

○ Con el desarrollo del primer dispositivo, se puede implementar otros sistemas de seguridad como lectores de tarjetas magnéticas o lectores biométricos al proyecto, dando una característica exclusiva al sistema donde el prototipo se acomoda al requerimiento del usuario y no que el usuario se acomode al proyecto, siempre y cuanto se respete las características que posee cada Shield y modulo del sistema con el fin de evitar problemas en su funcionalidad normal.

○ Se tuvo en cuenta que el sistema electrónico desarrollado, debió ser instalado lo más lejano posible del calor que irradia el motor de combustión interna y a su vez, protegido con una carcasa hermética que lo proteja de la humedad.

○ Se demostró por medio de una gran cantidad de pruebas tanto individuales como conjuntas de cada módulo o Shield del proyecto, el sistema promete ser una excelente herramienta para minimizar el caso de hurto a las motocicletas, con la ventaja de manipular y modificar el encendido del motor y su geoposicionamiento. Por tal razón este proyecto cumple con los objetivos indagados al comienzo de este documento.

Referencias

Alex, T. (2011). *Tutorial detallado de Arduino.* España: BricoGeek. Recuperado el 20 de Enero de 2016, de BricoGeek: http://blog.bricogeek.com/noticias/tutoriales/tutorial-arduino-gps-logger-con-em406a-gps-shield-y-microsd-shield/28/05/2016.

Ávila Jiménez, C. (2015). *Todos los días se roban 64 motos en ciudades capitales de Colombia.* Bogota, Cundinamarca, Colombia: El Tiempo. Recuperado el 12 de Marzo de 2016, de http://www.eltiempo.com/colombia/otras-ciudades/robo-de-motos-en-colombia-aumentan-los-casos-en-el-2015/15831737

Barahona, C. D., & Quitian, R. A. (2014). *Sistema prototipo de seguridad y vigilancia aplicado a medios de transporte público.* Bogotá: Universidad Católica de Colombia. Obtenido de http://repository.ucatolica.edu.co/bitstream/10983/2004/1/SISTEMA%20PROTOTIPO%20DE%20SEGURIDAD%20Y%20VIGILANCIA%20APLICABLE%20A%20MEDIOS%20DE%20TRANSPORTE%20P%C3%9ABLICO.pdf

Bellacetin, M. A. (2013). *Sistema de localización y bloqueo de maquinaria agrícola vía GSM/GPS.* México: Escuela Superior De Ingeniería Mecánica y Eléctrica. Obtenido de http://itzamna.bnct.ipn.mx:8080/dspace/bitstream/123456789/13222/1/43%2014.pdf

Bretto. (2015). *Alarma con localizador GPS w200.* España. Obtenido de http://bretto.co/es/alarma-para-moto/8-alarma-de-moto-con-localizador-gps-w200.html

Dealextreme. (2012). *Vibración sensor: SW-18020P.* Recuperado el 2 de Marzo de 2016, de http://www.dx.com/es/p/vibration-sensor-alarming-module-for-arduino-150728#.V0n_uTUeRHx/

Doutel, F. (2015). *Arduino.* Xataka. Recuperado el 20 de Enero de 2016, de http://www.xataka.com/especiales/guia-del-arduinomaniaco-todo-lo-que-necesitas-saber-sobre-arduino

E-auto. (s.f.). *Conceptos del funcionamiento del sistema de ignición.* México. Recuperado el 15 de Febrero de 2016, de http://e-auto.com.mx/manual_detalle.php?manual_id=246

Electan. (2016). *Shield SM5100B.* España. Recuperado el 26 de Enero de 2016, de Electan.com. (2016). [onlihttp://www.electan.com/arduino-shield-gsmgprs-sparkfun-con-sm5100b-p-3151.html

Fernández de Soto, A., Moreno, J. M., Restrepo, C. A., & Villegas, L. C. (2016). *Logros de la Política de Defensa y Seguridad Todos por un Nuevo País* (Vol. I). Bogotá, Cundinamarca, Colombia: Observatorio del delito de la DIJIN Policía Nacional. Recuperado el 10 de Marzo de 2016, de https://www.mindefensa.gov.co/irj/go/km/docs/Mindefensa/Documentos/descargas/estudios%20sectoriales/info_estadistica/Logros_Sector_Defensa.pdf

IBM Blueix. (2014). *Desarrolle aplicaciones Android con Eclipse.* DeveloperWorks. Recuperado el 3 de Marzo de 2016, de https://www.ibm.com/developerworks/ssa/opensource/tutorials/os-eclipse-android/

ICF Bike. (2012). *Alarma de moto con localizador GPS notificacion por MSM y aviso de caída y accidente.* España. Obtenido de http://www.lcfbike.com/

Jesus, R. (2014). *Tutorial de configuración Bluetooth HC-05 y HC-06.* México D.F, México: Geek Factory . Recuperado el 20 de Febrero de 2016, de http://www.geekfactory.mx/tutoriales/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion

Lipa Chouqe, D. (s.f.). *Sistema operativo Android.* Putina, Puno, Perú: Monografias.com. Recuperado el 7 de Febrero de 2016, de http://www.monografias.com/trabajos101/sistema-operativo-android/sistema-operativo-android.shtml

Marqués, A. (2008). *Diseño de un sistema de seguridad y de monitoreo de vehiculos.* Sartenejas: Universidad Simón Bolívar. Obtenido de http://159.90.80.55/tesis/000138576.pdf

Pelacrash. (2014). *Alarma Scorpio.* España. Obtenido de http://www.pelacrash.com/alarma-scorpio/

Principio de funcionamiento y cinemática del motor de combustión interna. (2005). Colombia: Biblioteca Virtual Arango, Luis Ángel. Recuperado el 2 de Febrero de 2016, de http://admin.banrepcultural.org/node/92121/

Prometec.Net. (2014). *Ardudroid, o controlando Arduino con el móvil.* Bilbao, España: Tag Archives of Bluetooth. Recuperado el 10 de Febrero de 2016, de http://www.prometec.net/tag/bluetooth/28/05/2016

Prometec.Net. (s.f.). *Control remoto RF sencillo, controlando a distancia con un mando.* Bilbao, España. Recuperado el 24 de Febrero de 2016, de http://www.prometec.net/control-remoto-rf/

Rodríguez, E. (2010). *¿Dónde estoy en este momento? Pregúntale al GPS.* Chile: Explora. Recuperado el 13 de Febrero de 2016, de http://www.explora.cl/455-sabias-que/sabias-fisica/647-donde-estoy-en-este-momento-preguntale-al-gps

Todo Motos. (2013). *¿Qué es el CDI?* Perú. Recuperado el 15 de Enero de 2016, de http://www.todomotos.pe/mecanica/1345-cdi-importancia-mantenimiento-moto/

Tolocka, E. (2015). *Módulo de 4 relés para Arduino.* Córdoba, Argentina. Recuperado el 15 de Enero de 2016, de http://www.profetolocka.com.ar/2015/05/09/modulo-de-4-reles-para-arduino/

Anexos

Apéndice 1

Código de programación de la aplicación en App Inventor

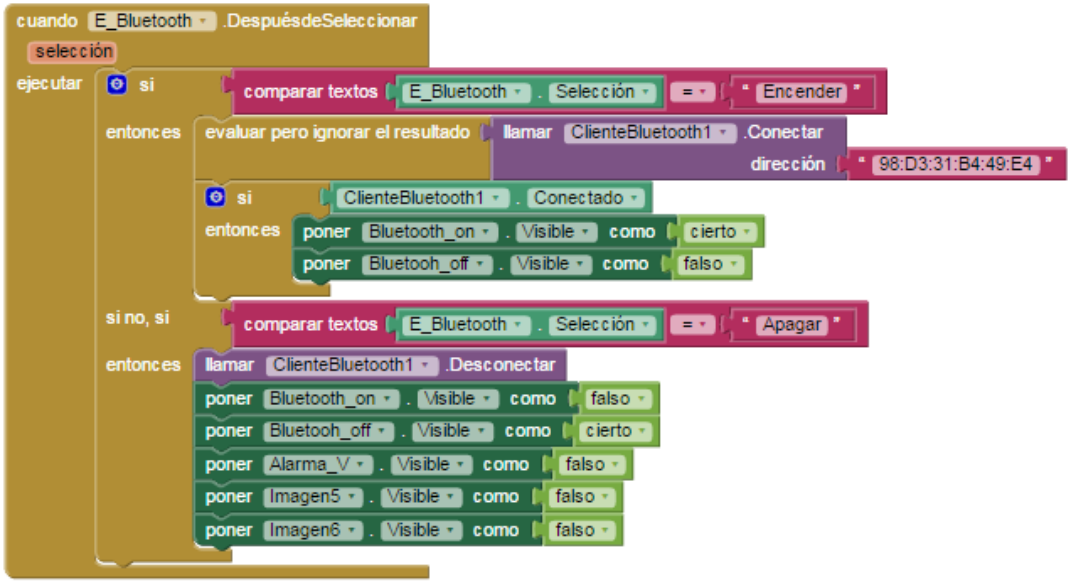

Figura 52. Conexión y desconexión del Bluetooth

Fuente. Autor.

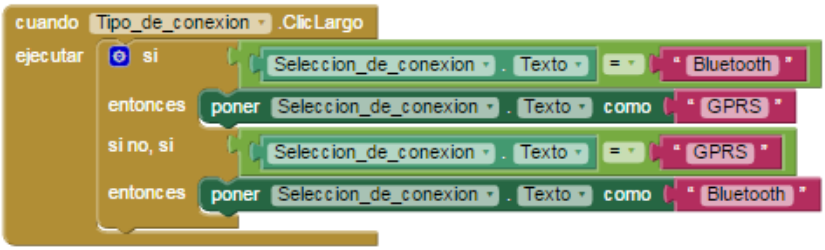

Figura 53. Selección de comunicación.

Fuente. Autor.

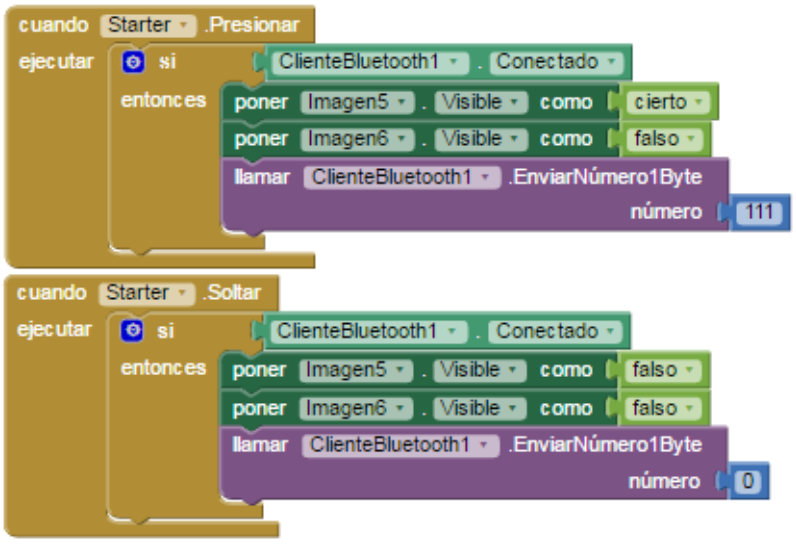

Figura 54. Botón Starter.

Fuente. Autor.

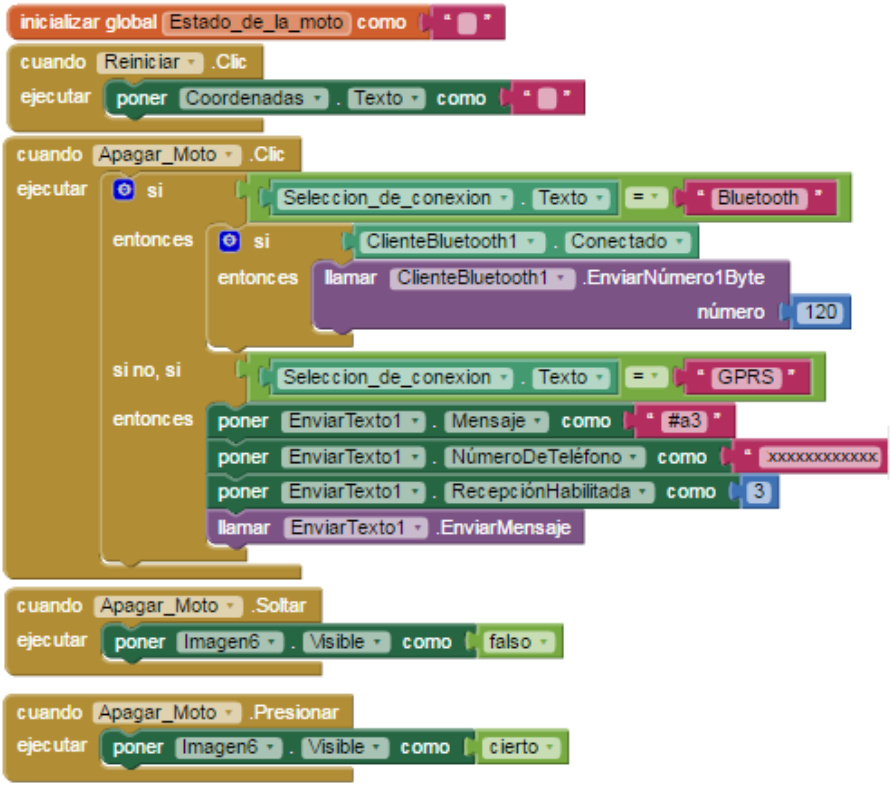

Figura 55. Botón Apagar moto.

Fuente. Autor.

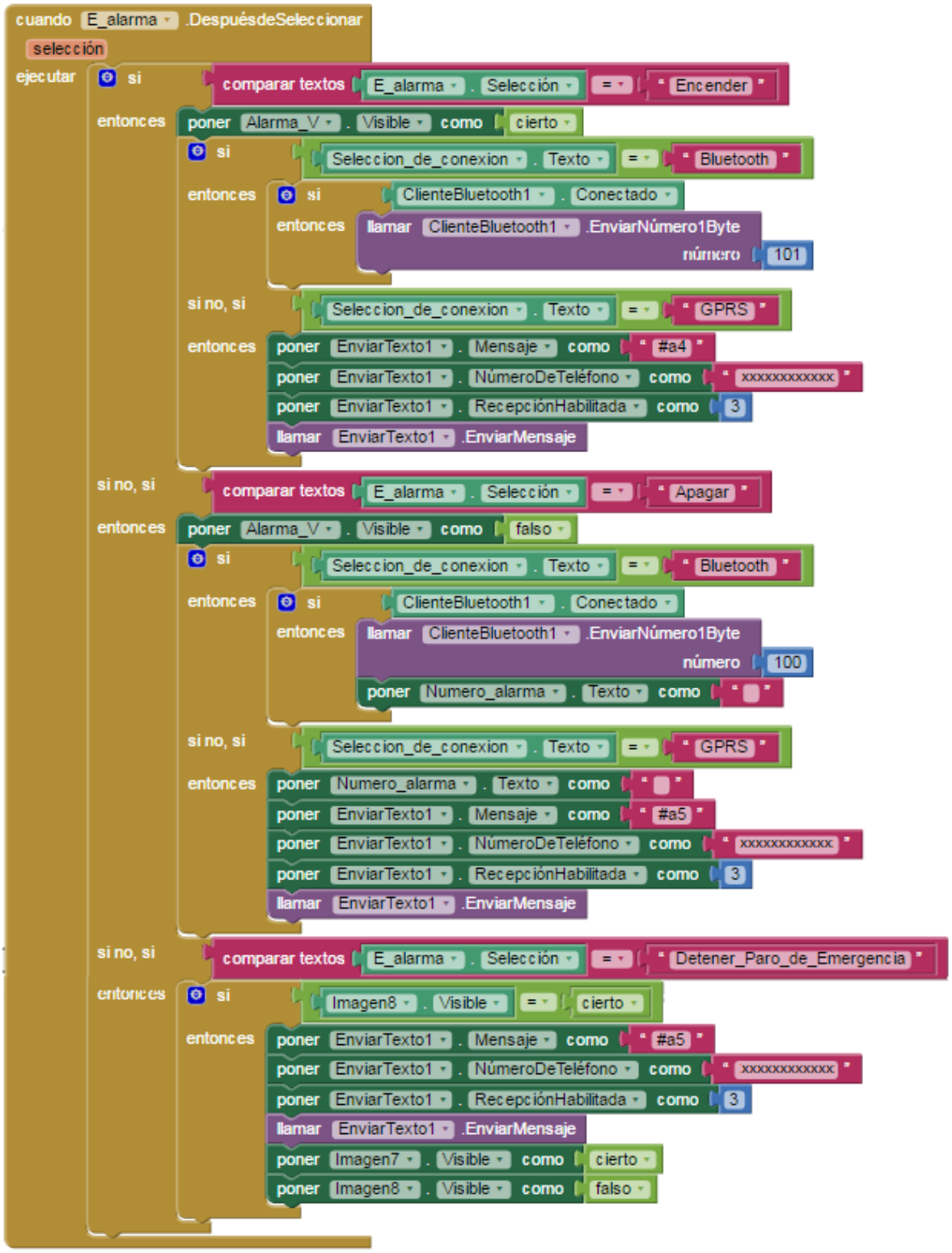

Figura 56. Botones encender y apagar alarma y Paro de emergencia.

Fuente. Autor.

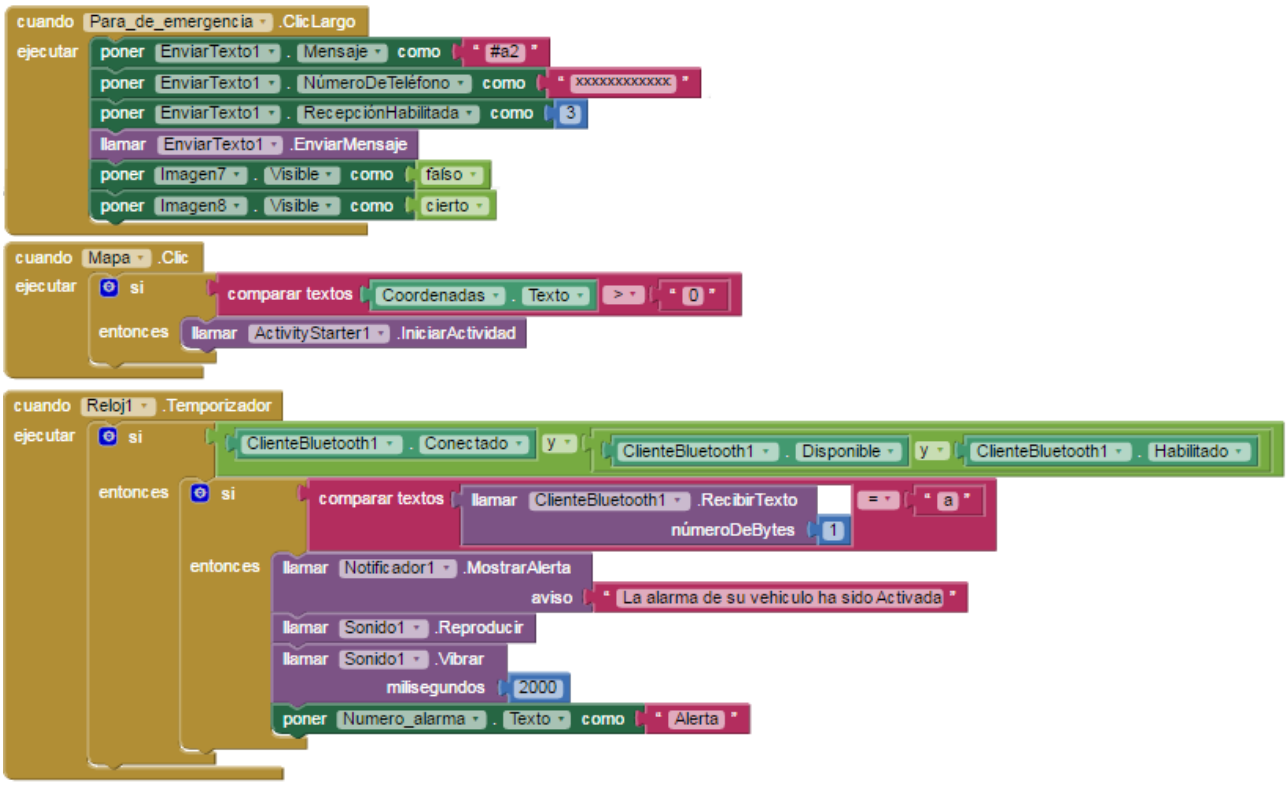

Figura 57. Alarma, selección de paro de emergencia y mapa.

Fuente. Autor.

Figura 58. Coordenadas y botón atrás.

Fuente. Autor.

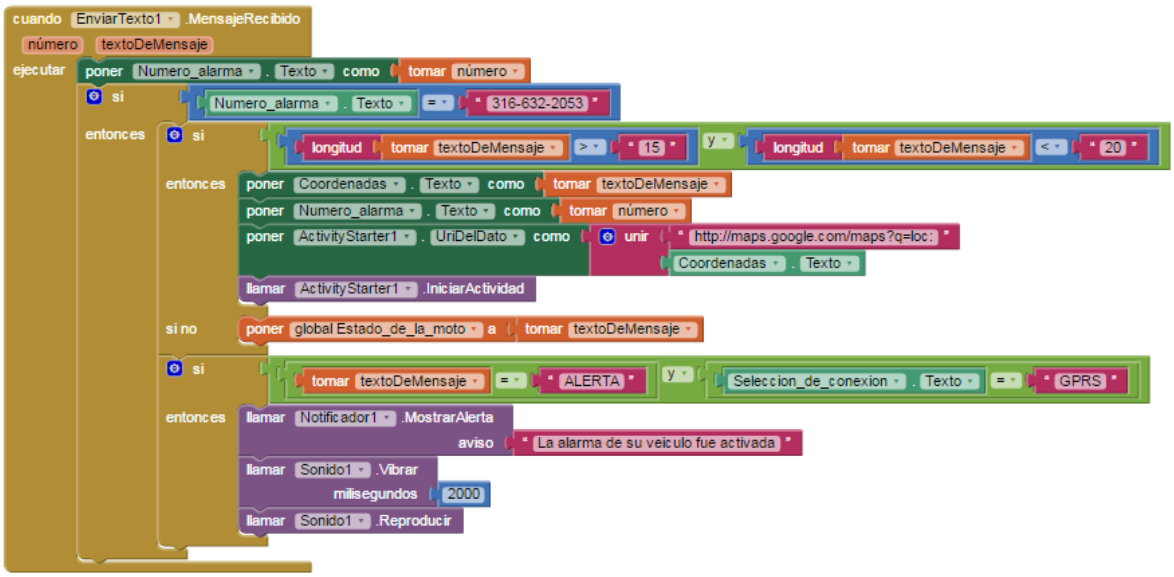

Figura 59. Activación de Google Maps y activación de alerta.

Fuente. Autor.

Apéndice 2

Código de programación Arduino

```
//----------------------------- Definición de Bibliotecas, variables y subrutinas----------------------
-
#include <WString.h>
#include <TinyGPS.h>
#define GPSBAUD 9600
TinyGPS gps;
void getgps(TinyGPS &gps);
byte neutro=36, vccin=38, sensor=34, pito=40, starter=42, vcc=44,
motor=46,vccbluetooth=48, a=41, b=43, c=47, d=45, led=39;
byte alarma, inicio, modotex, coordenadas, manual, estado, paro_de_emergencia, sirena,
a_automatica=1, estado_alarma=1, alarma_automatica, blue, borrar_mensaje;
int bluetooth, clave[]={2,2,0,2}, sec[]={100,100,100,100}, estadopito=HIGH;
unsigned long timer1=0, intervalo1=35000, timer2=0, intervalo2=30000, timer3=0,
intervalo3=500, timer4=0, intervalo4=10000, timer5=0, intervalo5=1000, timer6=0,
intervalo6=10000;
float lati, longi, latitud, longitud;
char phone_no[]="3132602143";
String cad;
```

```
//---------------------------------- Configuración inicial del dispositivo----------------------------
void setup()
{
 Serial.begin(57600);
 Serial1.begin(9600);
 Serial2.begin(9600);
 Serial3.begin(115200);
 pinMode (neutro,INPUT);
 pinMode (vccin,INPUT);
 pinMode (sensor,INPUT);
 pinMode (a,INPUT);
 pinMode (b,INPUT);
 pinMode (c,INPUT);
 pinMode (d,INPUT);
 pinMode (led,INPUT);
 pinMode (vccbluetooth,OUTPUT);
 pinMode (pito,OUTPUT);
 pinMode (motor,OUTPUT);
 pinMode (starter,OUTPUT);
 pinMode (vcc,OUTPUT);
//------------------------------- Definición de valores iniciales y constantes-----------------------
 inicio=1; modotex=1; alarma=0; coordenadas=0; manual=0; blue=1, borrar_mensaje=0;
 digitalWrite(motor,HIGH);
```

```
 digitalWrite(pito,HIGH);
 digitalWrite(starter,HIGH);
 digitalWrite(vcc,LOW);
 digitalWrite(vccbluetooth,HIGH);
}
//---------------------------------------- Inicio de ciclo infinito----------------------------------------
void loop()
{
 unsigned long reloj = millis();
 cad = "";
----------------------------------------Lectura y estado de la clave maestra-------------------------
 if(digitalRead(a)){
                  sec[estado]=0;
                  estado++;
                  delay(250);
                 }
 if(digitalRead(b)){
                  sec[estado]=1;
                  estado++;
                  delay(250);
                 }
if(digitalRead(c)){
                  sec[estado]=2;
```

```
                estado++;
                delay(250);
               }
if(digitalRead(d)){
                manual=0;
                estado=0;
                bluetooth=0;
                a_automatica=1;
                sirena=0;
                paro_de_emergencia=0;
                borrar_mensaje=0;
                blue=0;
                timer5=reloj;
                timer6=reloj;
                estado_alarma=0;
                Serial.println("reset sistema");
                digitalWrite(pito,HIGH);
                digitalWrite(vcc,LOW);
                delay(250);
                for(int i=0;i<3;i++){
                        digitalWrite(pito,LOW);
                        delay(50);
                        digitalWrite(pito,HIGH);
```

```
                    delay(150);
                   }
            }
if(estado==4){
if((sec[0]==clave[0])&&(sec[1]==clave[1])&&(sec[2]==clave[2])&&(sec[3]==clave[3])){
                                                    manual=1;
                                                    sirena=0;
                                                    paro_de_emergencia=0;
                                                    alarma_automatica=0;
                                                    alarma=0;
                                                    estado_alarma=0;
                                                    digitalWrite(pito,HIGH);
                                                    digitalWrite(motor,LOW);
                                                    digitalWrite(vcc,LOW);
                                                    Serial.print("Sistemas
desactivados");
                                                    for(int i=0;i<4;i++){
digitalWrite(pito,LOW);
                                                        delay(50);
digitalWrite(pito,HIGH);
                                                        delay(150);
                                                        }
                                                   }
```

```
                                                    else {
                                                        Serial.print("Clave
Erronea");
                                                        for(int i=0;i<1;i++){
digitalWrite(pito,LOW);
                                                            delay(50);
digitalWrite(pito,HIGH);
                                                            delay(150);
                                                        }
                                                    }
          estado=0;
          }
//------------------------------ Configurar el módulo SM5100B en modo texto----------------------
if (inicio==1){
           timer1=reloj;
           inicio=0;
          }
if(reloj>timer1+intervalo1 && modotex==1){
                              Serial3.println("AT+CMGF=1");
                              delayMicroseconds(10000);
                              Serial3.println("AT+CNMI=2,2,0,0");
                              delayMicroseconds(10000);
                              Serial3.println("AT+CMGD=1,4");
```

```
                    delayMicroseconds(10000);
                    modotex=0;
                   }
//-----------------------------------------Lectura del Bluetooth-----------------------------------
while (Serial1.available() && manual==0){
                   bluetooth=Serial1.read();
                  }
//---------------------------------------Lectura del GPS-------------------------------------------
while ( Serial2.available() && manual==0){
                    int bt =  Serial2.read();
                    if (gps.encode(bt)){
                                getgps(gps);
                               }
                   }
//-------------------------------------Lectura del GPRS/GSM---------------------------------------
while (Serial3.available() && manual==0) {
                    char c = Serial3.read();
                    cad += c;
                    delayMicroseconds(16000);
                    Serial.print(cad);
                    }
//-----------------------------------Comandos para Bluetooth-------------------------------------
if (bluetooth == 100){
```

```
                alarma=0;
                paro_de_emergencia=0;
                a_automatica=1;
                sirena=0;
                estado_alarma=1;
                digitalWrite(pito,HIGH);
                digitalWrite(motor,LOW);
                digitalWrite(vcc,LOW);
                Serial.println("Alarma off");
                for(int i=0;i<3;i++){
                                digitalWrite(pito,LOW);
                                delay(50);
                                digitalWrite(pito,HIGH);
                                delay(150);
                               }
                bluetooth=0;
               }
if (bluetooth == 101){
                alarma=1;
                sirena=0;
                estado_alarma=1;
                digitalWrite(motor,HIGH);
                digitalWrite(pito,HIGH);
```

```
                digitalWrite(vcc,LOW);
                Serial.println("Alarma On");
                for(int i=0;i<2;i++){
                        digitalWrite(pito,LOW);
                        delay(50);
                        digitalWrite(pito,HIGH);
                        delay(150);
                       }
                bluetooth=0;
               }
if (bluetooth == 111 && (digitalRead(neutro)==LOW) && (digitalRead(vccin)==LOW)){
                                                digitalWrite(starter,LOW);
                                                Serial.println("Starter");
                                               } else {
                                                    digitalWrite(starter,HIGH);
                                                   }
if (bluetooth == 120){
                digitalWrite(pito,HIGH);
                digitalWrite(motor,HIGH);
                digitalWrite(vcc,HIGH);
                Serial.println("moto off");
                sirena=0;
                bluetooth=0;
```

```
                }
//-------------------------------------cadenas del módulo GPRS/GSM-------------------------
if (cad.lastIndexOf("#a1") > -1){
                        Serial3.println("AT+CMGF=1");
                        delay(100);
                        Serial3.print("AT+CMGS=");
                        Serial3.print((char)34);
                        Serial3.print(phone_no);
                        Serial3.println((char)34);
                        delay(100);
                        Serial3.print(lati,5);
                        Serial3.print(",");
                        Serial3.print(longi,5);
                        Serial3.print((char)26);
                        Serial.println("Coordenadas Enviadas");
                        timer6=reloj;
                        borrar_mensaje=1;
                       }
if (cad.lastIndexOf("#a2") > -1){
                        paro_de_emergencia=1;
                        timer6=reloj;
                        borrar_mensaje=1;
                       }
```

```
if (cad.lastIndexOf("#a3") > -1){
                        sirena=0;
                        digitalWrite(pito,HIGH);
                        digitalWrite(motor,HIGH);
                        digitalWrite(vcc,LOW);
                        Serial.println("moto off");
                        Serial3.println("AT+CMGF=1");
                        delay(100);
                        Serial3.print("AT+CMGS=");
                        Serial3.print((char)34);
                        Serial3.print(phone_no);
                        Serial3.println((char)34);
                        delay(100);
                        Serial3.print("La moto ha sido apagada");
                        Serial3.print((char)26);
                        timer6=reloj;
                        borrar_mensaje=1;
                       }
if (cad.lastIndexOf("#a4") > -1){
                        alarma=1;
                        sirena=0;
                        estado_alarma=1;
```

```
digitalWrite(motor,HIGH);
digitalWrite(pito,HIGH);
digitalWrite(vcc,LOW);
Serial.println("Alarma On");
for(int i=0;i<2;i++){
                digitalWrite(pito,LOW);
                delay(50);
                digitalWrite(pito,HIGH);
                delay(150);
               }
Serial3.println("AT+CMGF=1");
delay(100);
Serial3.print("AT+CMGS=");
Serial3.print((char)34);
Serial3.print(phone_no);
Serial3.println((char)34);
delay(100);
Serial3.print("La alarma ha sido activada");
Serial3.print((char)26);
timer6=reloj;
borrar_mensaje=1;
}
```

```
if (cad.lastIndexOf("#a5") > -1){
                        alarma=0;
                        paro_de_emergencia=0;
                        sirena=0;
                        a_automatica=1;
                        estado_alarma=1;
                        digitalWrite(pito,HIGH);
                        digitalWrite(motor,LOW);
                        digitalWrite(vcc,LOW);
                        Serial.println("Alarma off");
                        for(int i=0;i<3;i++){
                                digitalWrite(pito,LOW);
                                delay(50);
                                digitalWrite(pito,HIGH);
                                delay(150);
                               }
                        Serial3.println("AT+CMGF=1");
                        delay(100);
                        Serial3.print("AT+CMGS=");
                        Serial3.print((char)34);
                        Serial3.print(phone_no);
                        Serial3.println((char)34);
                        delay(100);
```

```
                    Serial3.print("La alarma ha sido desactivada");
                    Serial3.print((char)26);
                    timer6=reloj;
                    borrar_mensaje=1;
                   }
//------------------------------Activación del paro de emergencia-----------------------------
if(paro_de_emergencia==1){
                if (reloj>timer2+intervalo2){
                                timer2 = reloj;
                                alarma=1;
                                digitalWrite(motor,HIGH);
                                digitalWrite(vcc,LOW);
                                Serial.println("Paro de emergecia activado");
                                Serial3.println("AT+CMGF=1");
                                delay(100);
                                Serial3.print("AT+CMGS=");
                                Serial3.print((char)34);
                                Serial3.print(phone_no);
                                Serial3.println((char)34);
                                delay(100);
                                Serial3.print(lati,5);
                                Serial3.print(",");
                                Serial3.print(longi,5);
```

```
                                    Serial3.print((char)26);
                                   }
                  }
//--------------------------------------Activación de la alarma-----------------------------------
if(digitalRead(sensor)==LOW && alarma==1){
                                    sirena=1;
                                   }
if(alarma==1 && sirena==1){
                   digitalWrite(vcc,LOW);
                   if(reloj>timer3+intervalo3){
                                    timer3 = reloj;
                                    if (estadopito == HIGH) {
                                                    estadopito = LOW;
                                                   }else {
                                                        estadopito = HIGH;
                                                       }
                                                       digitalWrite(pito, estadopito);
                                    }
                  }
                  if(digitalRead(vccin)==LOW){
                    timer4=reloj;
                   }
```

```
if(digitalRead(vccin)==HIGH && alarma==0 && a_automatica==1 && manual==0){
                                                timer4=reloj;
                                                alarma_automatica=1;
                                                a_automatica=0;
                                               }
if(reloj>timer4+intervalo4 && alarma_automatica==1 && alarma==0 &&
digitalRead(vccin)==HIGH){
                                                        for(int i=0;i<2;i++){
digitalWrite(pito,LOW);
                                                                delay(50);
digitalWrite(pito,HIGH);
                                                                delay(150);
                                                               }
                                                        digitalWrite(motor,HIGH);
                                                        alarma_automatica=0;
                                                        alarma=1;
                                                       }
if(blue==0){
        digitalWrite(vccbluetooth,LOW);
        if (reloj>timer5+intervalo5){
                        digitalWrite(vccbluetooth,HIGH);
                        blue=1;
                       }
```

```
        }
//-----------------------------------Eliminación de mensajes recibidos----------------------------
if(borrar_mensaje==1){
                if (reloj>timer6+intervalo6){
                                        Serial3.println("AT+CMGD=1,4");
                                        borrar_mensaje=0;
                                       }
                }
//--------------------------------------Accionamiento de la alarma---------------------------------
if(sirena==1 && estado_alarma==1){
                                Serial.println("Alarma");
                                Serial3.println("AT+CMGF=1");
                                delay(100);
                                Serial3.print("AT+CMGS=");
                                Serial3.print((char)34);
                                Serial3.print(phone_no);
                                Serial3.println((char)34);
                                delay(100);
                                Serial3.print("ALERTA");
                                Serial3.print((char)26);
                                estado_alarma=0;
                               }
}
```

```
//-----------------------------Extracción de coordenadas del módulo GPS-------------------------


void getgps(TinyGPS &gps)
{
 float latitude, longitude;
 gps.f_get_position(&latitude, &longitude);
 lati=latitude;
 longi=longitude;
 Serial.print("Lat/Long: ");
 Serial.print(lati,5);
 Serial.print(", ");
 Serial.println(longi,5);
}
```

Apéndice 3

Manual de la aplicación para Android

Se copia el instalador en el celular y se selecciona para instalar.

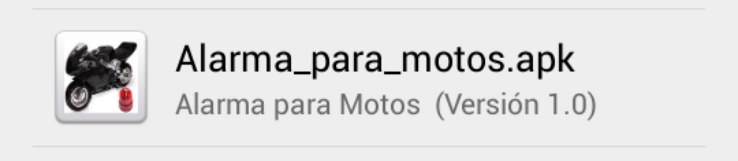

Figura 60. APK.

Fuente. Autor.

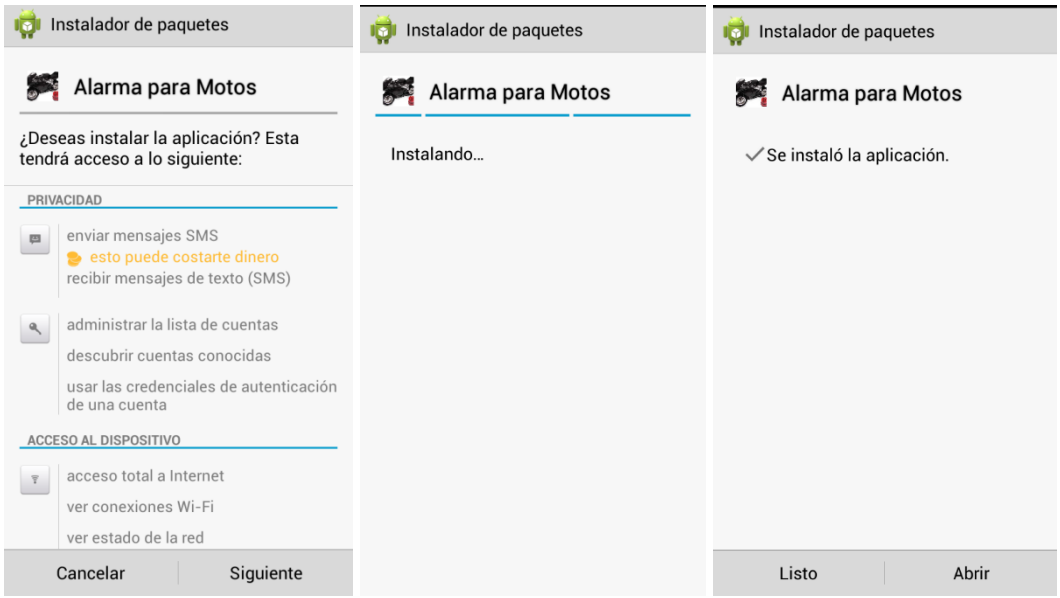

Figura 61. Instalación del APK.

Fuente. Autor.

Una vez instalado, se procede a enlazar el Bluetooth ingresando el PIN de seguridad al dispositivo.

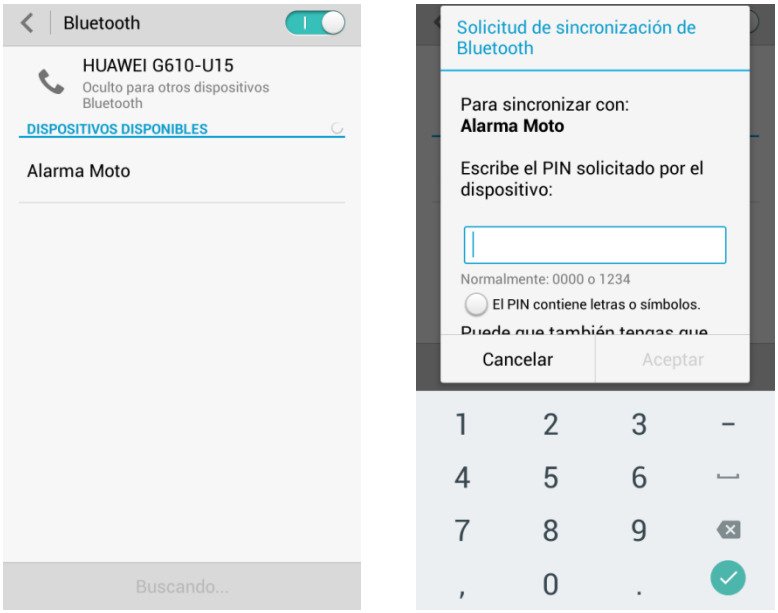

Figura 62. Enlace Bluetooth.

Fuente. Autor.

En la aplicación se hace la activación del Bluetooth seleccionando encender en Estado del Bluetooth. El enlace se habrá establecido cuando el icono cambie de color.

Figura 63. Activar enlace Bluetooth.

Fuente. Autor.

El botón Estado de la alarma, Activa y desactiva la alarma de la moto y desactiva el modo paro de emergencia.

Figura 64. Botón Estado de la alarma.

Fuente. Autor.

El botón Starter, hará la ignición de la moto, siempre y cuando la moto se encuentre en Neutro. Por otro lado, el botón Apagar moto, desactivara todo el sistema eléctrico de la motocicleta incluyendo la bobina de alta

Figura 65. Botón Starter y Apagar moto.

Fuente. Autor.

El último grupo de botones, son correspondientes al geoposicionamiento de la motocicleta.

El botón Coordenadas, Envía un mensaje de texto para obtener la posición del vehículo, a través de Google Maps de forma automática, sin afectar su función. El botón Paro de emergencia, tiene la misma función que el botón coordenadas, con la diferencia de que este desactiva todo el sistema eléctrico de la moto. Una vez se obtiene el mensaje de texto que contiene las coordenadas, se visualizan en la parte superior del botón mapa. Al

oprimir este botón, se reactiva Google Maps y define la última posición guardada en la aplicación. Por último, el botón Reiniciar, borra dichas coordenadas de la aplicación.

Figura 66. Botón Activar mapa, Paro de emergencia, Reiniciar y mapa.

Fuente. Autor.

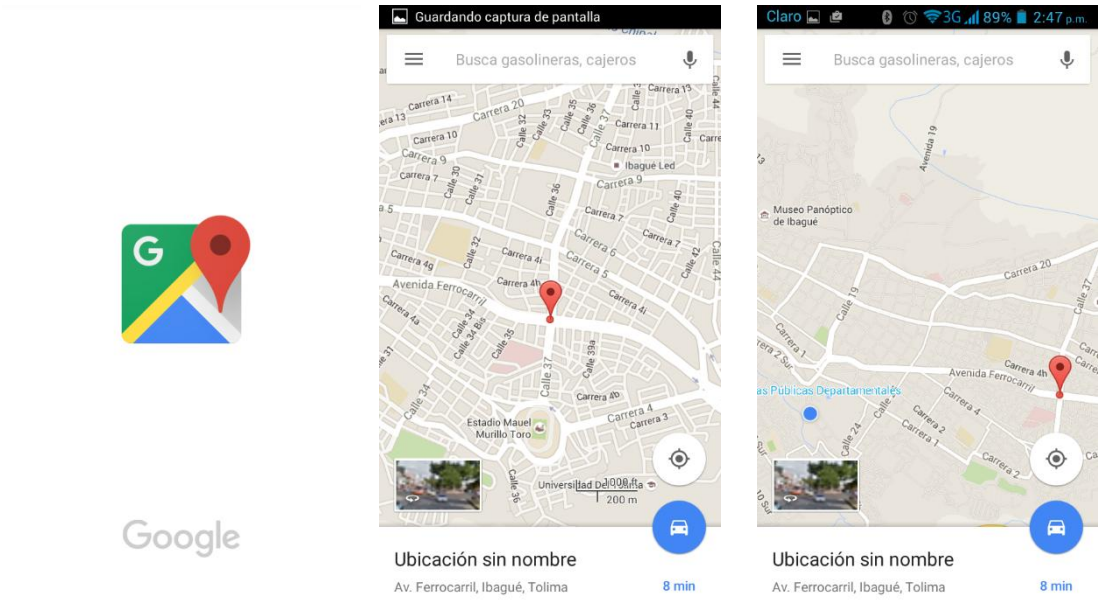

Figura 67. Indicación automática del vehículo.

Fuente. Autor.