Universidad de las Ciencias Informáticas

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Sistema de alarma para automóviles basado en dispositivos Arduino

Autor: Javier Rodríguez Rodríguez

Tutores:

Dr.C Antonio Cedeño Pozo

Ing. Yulia Fustiel Alvarez

La Habana, junio 2016

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Firma del autor: Javier Rodríguez Rodríguez.

_____

Firma del tutor: Ing. Yulia Fustiel Alvarez.

_____

Firma del tutor: Dr.C. Antonio Cedeño Pozo.

DATOS DE LOS CONTACTOS

Tutor: Ing. Yulia Fustiel Alvarez.

Edad: 29

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas en el 2009

Categoría docente: Instructor

E-mail: yfustiel@uci.cu

Tutor: Dr.C. Antonio Cedeño Pozo.

Edad: 31

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Doctor en Ciencias Técnicas (Informáticas) en el 2015

Categoría docente: Instructor

E-mail: acedeno@uci.cu

DEDICATORIA

A mis padres por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo mantenido a través del tiempo.

Todo este trabajo ha sido posible gracias a ellos.

AGRADECIMIENTOS

Especialmente para mis principales formadores, mis padres, estos que han sabido guiarme en todo el transcurso de mi vida y continuamente me han ofrecido lo mejor.

A mis abuelos que siempre se han preocupado por mí, apoyándome en todo momento y enseñándome muchas cosas de la vida.

A mis hermanos, especialmente mi hermana Yumara y Carmita, las que siempre me han apoyado y han estado presente cada vez que las he necesitado contribuyendo a la realización de este sueño.

A mis tíos queridos y mis primos que han formado gran parte de toda mi vida.

A mi tutora Yulia, la que me ofreció un gran apoyo desinteresado, sabiéndome guiar por el camino correcto, brindándome gran parte de su tiempo y conocimiento.

A mi tutor Tony, por guiarme y mostrarme este gran mundo de Arduino y Android en el cual me he adentrado para dar mis primeros pasos como profesional.

A todos, amistades y personas que han formado parte de mi vida y han contribuido al desarrollo de este trabajo.

RESUMEN

En el presente trabajo se describe el desarrollo de un sistema de alarma para automóviles basado en dispositivos Arduino. La necesidad del mismo surge, debido a que Cuba se encuentra en un proceso de actualización del modelo económico, político y social respondiendo a los objetivos de la sociedad cubana de informatización de los servicios. Proceso que se caracteriza por soluciones a corto plazo que potencien la generación de ingresos externos y la sustitución de importaciones, factor a tener en cuenta ante el desconocimiento de sistemas de alarmas para automóviles desarrolladas en Cuba. Además de que gran parte del parque de autos que circulan es obsoleto, lo que dificulta la instalación de sistemas de alarmas modernas producidas en el extranjero, por no estar diseñadas para estos modelos de autos, sin olvidar los elevados costos de adquisición e instalación que presentan.

Para conformar la propuesta de solución se realizó un análisis de algunos de los sistemas de alarmas para automóviles identificándose las características útiles que se podrían incorporar al desarrollo de la solución. Se seleccionaron las herramientas, tecnologías adecuadas y se guío el proceso de desarrollo conforme a lo establecido por la metodología Variación AUP para la UCI (AUP-UCI). Como resultado se obtuvo un prototipo de sistema de alarma para automóviles basado en dispositivos Arduino.

Palabras clave: Arduino, automóvil, sistema de alarma.

ÍNDICE

INTRODUCCIÓN

Debido a los actos delictivos como el hurto. El ser humano ha buscado métodos adecuados para cuidar y proteger sus pertenencias. Uno de estos métodos fue el de organizarse y cuidarse mutuamente. Sin embargo, con el pasar del tiempo la humanidad tuvo que desarrollar sistemas de seguridad con un alto nivel de funcionalidad a la hora de proteger bienes materiales que poseían un nivel elevado de valor. De esta manera se ha contribuido al aumento de la seguridad ante el crecimiento los actos delictivos.

Según (Edgardo Luna Melara, et al., 2013) ya desde el 1914 existían los robos de autos, los cuales fueron en incremento al pasar de los años. Por tal situación en el 1920 en el estado de Nebraska, se inventa la primera alarma de autos. Su diseño consistía en un interruptor de llave, una caja de acero con un embobinado y una sirena. Este sistema estaba puesto en uno de los ejes y al dar vueltas activaba el campo magnético, lo que generaba el voltaje y el sonido de la sirena, (prácticamente trabajaba igual que un alternador), siempre que su dueño la activara con el interruptor de llaves, ubicado en el guardafangos correspondiente al chofer.

En la actualidad, a nivel mundial gracias al adelanto tecnológico existen diversos sistemas de seguridad que han sido desarrollados para responder a las diferentes necesidades de la población que se ve expuesta a los robos. Además, muchos modelos de autos ya traen sistemas antirrobo y puedes elegir agregarles una alarma para mayor seguridad. Cuanto más sofisticado sea el sistema, mayor será su precio de compra e instalación. Algunas de las tecnologías utilizadas para fabricarlas son los sensores, sirenas, receptores de radio, aplicaciones móviles, baterías auxiliares y unidades de control.

Los sistemas de seguridad son un conjunto de dispositivos ubicados en sitios específicos y estratégicos con el objetivo de detectar y alertar la presencia de aquellos individuos que no posean el acceso autorizado a dichas zonas.

La seguridad desarrollada en la gama automotriz, tiene como objetivo reducir la posibilidad de que ocurran los actos delictivos del hurto de vehículos y así contribuir a la protección de los mismos.

No obstante, según el Centro Ecuatoriano de Análisis de Seguridad Integral (CEASI) en Ecuador, el índice de robos en los últimos años es impresionante a pesar de la existencia de los sistemas de seguridad, que son vulnerados en la mayoría de las veces por los ladrones. El robo de vehículos es recurrente en toda la sociedad, pues es un bien atrayente para los delincuentes, porque pueden comercializar tanto las partes como los vehículos completos. En segundo lugar, sirve como herramienta para cometer otro tipo de delitos (CEASI, 2013).

Actualmente, Cuba se encuentra inmersa en un proceso de actualización del modelo económico, político y social en sus artículos (PCC, 2011), lo que responde a los objetivos de la sociedad cubana de informatización de los servicios. Proceso que se caracteriza por soluciones a corto plazo encaminadas a eliminar el déficit de la balanza de pagos que potencien la generación de ingresos externos y la sustitución de importaciones, factor a tener en cuenta, ante el desconocimiento sobre los sistemas de alarmas para automóviles desarrollados en Cuba. Además, gran parte del parque de autos que circulan es obsoleto, lo que dificulta la instalación de sistemas de alarmas modernas producidas en el extranjero, puesto que no están diseñadas para estos modelos de autos, sin olvidar los elevados costos de adquisición e instalación que presentan.

Por todo lo anteriormente planteado se propone el siguiente **problema a resolver:** ¿Cómo contribuir a la seguridad de los autos en Cuba?

Lo que enmarca como **objeto de estudio:** Sistemas de alarma para automóviles.

Y para dar solución al problema planteado se ha formulado como **objetivo general:** Desarrollar un sistema de alarma para automóviles basado en dispositivos Arduino.

Por todo lo anterior se determina como **campo de acción:** Sistemas de alarma para automóviles basados en microcontroladores.

Para darle cumplimiento a dicho objetivo se trazaron las siguientes **tareas de investigación:**

- ✓ Fundamentación teórica a partir del estado del arte de los sistemas de alarmas para autos.
- ✓ Selección del microcontrolador adecuado para la solución.
- ✓ Definición de los requerimientos funcionales y no funcionales.
- ✓ Diseño e implementación del sistema de alarma.

- ✓ Diseño y ejecución de pruebas para validar la solución propuesta.

En la investigación se emplearon los siguientes métodos científicos:

El método dialéctico materialista como método general de la ciencia. Para su desarrollo se utilizaron métodos de carácter teórico y empírico, que permitieron un acercamiento a la esencia del fenómeno estudiado, y al mismo tiempo, permitieron encontrar las explicaciones al problema objeto estudio.

Entre los métodos teóricos utilizados se encuentran:

Histórico–Lógico: se utiliza este método para realizar el estudio del arte, y conocer acerca de los sistemas o soluciones similares, además de los lenguajes, herramientas y metodologías para el desarrollo del sistema que se propone.

Analítico-Sintético: permitió el análisis y estudio de las bibliografías existentes sobre el tema en cuestión, lo que posibilitó obtener de manera sintetizada el contenido necesario y suficiente para la realización del presente trabajo.

Inductivo-deductivo: facilitó determinar las regularidades que demuestran la necesidad de la inserción teórica y práctica del sistema de alarma para automóviles basado en dispositivos Arduino y llegar a conclusiones parciales y generales.

Modelación: permitió realizar los diagramas necesarios para darle cumplimiento a los requisitos funcionales y no funcionales asociados al sistema.

Entre los métodos empíricos se encuentran:

Estudio-documental: posibilitó la consulta bibliográfica referente a la arquitectura de información, así como otras fuentes que tributan al adecuado desarrollo de la investigación.

Observación: permitió detectar el problema y sus posibilidades de solución. Se observaron diferentes sistemas de alarmas que aportaron información para la realización del proyecto.

Prueba de validación: se realizaron las pruebas pertinentes a la propuesta de solución, como constancia del cumplimiento de los objetivos.

Los contenidos del documento aparecen distribuidos en la siguiente estructura capitular:

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA: se recoge el marco teórico de la investigación y se precisa lo referente al estado del arte, así como una serie de definiciones respecto a la

temática planteada, pasos establecidos que se deben seguir para desarrollar la aplicación y metodologías de desarrollo de software.

CAPÍTULO 2. CARÁCTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA: se presenta la fundamentación de la propuesta de solución. Se describen las actividades de análisis de la solución, los procesos del sistema y las etapas de diseño e implementación que conllevan a la obtención de la aplicación.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA: se describe la etapa de implementación que conlleva a la obtención de la aplicación. Además, se elaboran y documentan las pruebas realizadas a las funcionalidades desarrolladas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se precisan un conjunto de elementos para conformar el marco teórico relacionado con los aspectos definidos en el objeto de estudio. Se analizan los conceptos más significativos recogidos de diferentes fuentes. Además, se identifican, analizan y se seleccionan las tecnologías y herramientas existentes para la conformación del prototipo.

1.1 Sistema de seguridad vehicular

El sistema de seguridad vehicular también conocido como alarma vehicular es un dispositivo que tiene por objetivo intentar alertar mediante la señal de una sirena y de luces en caso de un intento de acceso no autorizado al interior del vehículo.

Generalmente, estos sistemas de seguridad cuentan con varios componentes, ejemplo de ellos son: unidad central de proceso, mandos a distancia, sensores y sirena. Todos ellos ubicados en sitios específicos del auto (ver anexo 1) y monitoreados por el microcontrolador que se localiza en el interior de la unidad central de proceso con el objetivo de captar cualquier anomalía a través de sus sensores haciendo disparar el sistema.

Figura 1. Alarma Delta

1.2 Microcontrolador

Debido al avance tecnológico, cada vez existen más dispositivos electrónicos que incorporan microcontroladores dentro de su hardware. Ejemplo de esto son los televisores, lavadoras, microondas e impresoras. Esto se debe, a las diversas ventajas que brindan como son: tamaño y costo reducido, mejor fiabilidad, menor consumo energético ente otras.

Existen numerosos tipos de microcontroladores, entre los más comunes se encuentran:

Atmel (AVR), Hitachi (H8), Intel de 8 bits (8XC42, MCS51, 8xC251) o Intel de 16 bits (MCS96, MXS296), National Semiconductor (COP8), Microchip, Motorola de 8 bits (68HC05, 68HC08, 68HC11) o de 16 bits (68HC12, 68HC16) o de 32 bits (683xx), NEC (78K), Texas Instruments (TMS370) y Zilog (Z8, Z86E02).

Según (Artengo, 2013) un microcontrolador (ver figura 2) es un circuito integrado o chip (es decir, un dispositivo electrónico que integra en un solo encapsulado un gran número de componentes) que tiene la característica de ser programable. Es decir, que es capaz de ejecutar de forma autónoma una serie de instrucciones previamente definidas por los desarrolladores. Por definición, un microcontrolador (también llamado comúnmente “micro”) ha de incluir en su interior tres elementos básicos:

- ✓ **CPU (Unidad Central de Proceso):** es la parte encargada de ejecutar cada instrucción y de controlar que dicha ejecución se realice correctamente. Normalmente, estas instrucciones hacen uso de datos disponibles previamente (los “datos de entrada”), y generan como resultado otros datos diferentes (los “datos de salida”), que podrán ser utilizados (o no) por la siguiente instrucción.
- ✓ **Diferentes tipos de memorias:** son en general las encargadas de alojar tanto las instrucciones como los diferentes datos que estas necesitan. De esta manera posibilitan que toda esta información (instrucciones y datos) esté siempre disponible para que la CPU pueda acceder y trabajar con ella en cualquier momento. Generalmente, se encuentran dos tipos de memorias: las que su contenido se almacena de forma permanente incluso tras cortes de alimentación eléctrica (llamadas “persistentes”), y las que su contenido se pierde al dejar de recibir alimentación (llamadas “volátiles”). Según las características de la información a guardar, esta se grabará en un tipo u otro de

memoria de forma automática, habitualmente.

- ✓ **Diferentes E/S (entrada/salida):** Encargadas de comunicar el microcontrolador con el exterior. En los pines de entrada del microcontrolador se pueden conectar diversos sensores para que este pueda recibir datos provenientes de su entorno, y en sus pines de salida se pueden conectar actuadores para que el microcontrolador pueda enviarles órdenes y así interactuar con el medio físico. De todas formas, muchos pines de la mayoría de microcontroladores no son exclusivamente de entrada o de salida, sino que pueden ser utilizados indistintamente para ambos propósitos (de ahí el nombre de E/S).

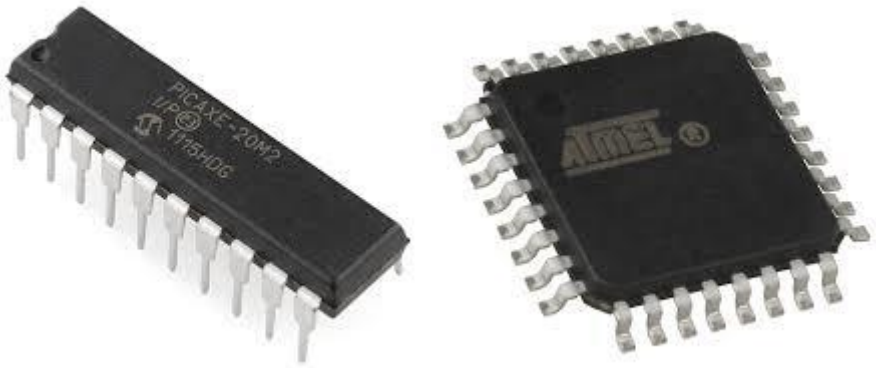


Figura 2. Microcontrolador

Es decir, un microcontrolador es un computador completo (aunque con prestaciones limitadas) en un solo chip, que está especializado en ejecutar constantemente un conjunto de instrucciones predefinidas. Estas instrucciones irán teniendo en cuenta en cada momento la información obtenida y enviada por los pines de E/S y reaccionarán en consecuencia. Lógicamente, las instrucciones serán diferentes según el uso que se le quiera dar al microcontrolador, en el caso de esta investigación es un sistema de alarma para automóviles basado en dispositivos Arduino.

1.2.1 Arduino

Según (Arduino.org, 2016) Arduino es una plataforma electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar.

El hardware consiste en una placa con un microcontrolador reprogramable y varios puertos de entrada/salida, tanto digitales como analógicos, así como salidas PWM y de comunicaciones.

Arduino puede tomar información del entorno a través de sus entradas analógicas y digitales, y controlar luces, motores y otros actuadores.

Los microcontroladores más usados son ATmega168, ATmega328, ATmega1280 por su sencillez y bajo costo. Desde 2012, Arduino se usa también con microcontroladores de ARM.

ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar con el mismo IDE de Arduino y hacerse programas que compilen sin cambios en las dos plataformas.

El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en *Wiring*) y el entorno de desarrollo Arduino (basado en *Processing*).

Con Arduino se pueden realizar multitud de proyectos, de temas como: robótica, domótica, monitorización de sensores ambientales, sistemas de navegación y telemática.

Realmente, las posibilidades de esta plataforma para el desarrollo de productos electrónicos son prácticamente infinitas y tan solo limitadas por la imaginación.

1.2.2 Características de Arduino

Existen muchas otras placas de diferentes fabricantes que, aunque incorporan diferentes modelos de microcontroladores, son comparables y ofrecen una funcionalidad más o menos similar a la de las placas Arduino, como podrían ser Parallax Basic Stamp, BX-24 de NEtmedia, Phidgets o Handyboard de MIT por citar algunas.

Todas ellas también vienen acompañadas de un entorno de desarrollo agradable y cómodo y de un lenguaje de programación sencillo y completo.

No obstante, la plataforma Arduino (Hardware + Software) ofrece una serie de ventajas:

- ✓ **Multiplataforma:** se puede instalar y ejecutar en sistemas Windows, MAC OS y Linux. Esto no ocurre con el software de muchos otros microcontroladores.
- ✓ **Entorno de programación simple:** el entorno de programación de Arduino es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados. Además, existe mucha documentación con ejemplos detallados y gran cantidad de proyectos publicados en diferentes formatos.
- ✓ **Arduino tiene una gran comunidad:** muchas personas lo utilizan, enriquecen la documentación y comparten continuamente sus ideas.
- ✓ **Libre y extensible:** cualquiera que desee ampliar y mejorar tanto el diseño hardware de las placas como el entorno de desarrollo software y el propio lenguaje de programación, puede hacerlo sin problemas.

- ✓ **Asequible:** las placas Arduino son más asequibles comparadas con otras plataformas de microcontroladores. La placa estándar, llamada Arduino UNO, pre ensamblada y lista para funcionar cuesta alrededor de 20 euros y la versión más cara cuesta alrededor de 60 euros.
- ✓ **Reutilizables y versátiles:** reutilizables porque se puede aprovechar la misma placa para varios proyectos, pues es muy fácil de desconectar, reconectarla y reprogramarla. Versátiles porque proveen varios tipos de entradas y salidas de datos, que permiten capturar información de sensores y enviar señales a actuadores de múltiples formas.

Por todas estas ventajas se seleccionó la plataforma de Arduino para el desarrollo de la propuesta de solución. Esta plataforma posibilita la comunicación con otros dispositivos por ejemplo: los teléfonos móviles inteligentes, que con el avance tecnológico han sido capaces de reproducir archivos de audio, también, sistemas operativos y conexión a internet. De esta manera, los usuarios empezaron a entender el móvil como una prolongación de sus ordenadores en movimiento, así nació el concepto de dispositivo móvil inteligente (Acosta, 2011).

1.3 Dispositivo móvil inteligente

Un dispositivo móvil inteligente es un dispositivo electrónico, por lo general conectado a otros dispositivos o redes a través de diferentes protocolos como Bluetooth, NFC, Wi-Fi, 3G, X10, entre otros, que puede funcionar hasta cierto punto de forma interactiva y autónoma. Los más conocidos y reconocidos internacionalmente son los *smartphones* o teléfonos inteligentes o la mayoría de los dispositivos con sistema operativo integrado, como los *tablets* o tabletas. El campo se ha expandido además a otros equipos como televisores, relojes y automóviles. Este tipo de dispositivos incorporan funciones que hace poco parecían futuristas, como juegos, reproducción de música MP3 y otros formatos, correo electrónico, servicio de mensajes cortos (SMS por sus siglas en inglés), agenda electrónica PDA, fotografía digital y video digital, video llamada, navegación por Internet, Sistema de Posicionamiento Global (GPS por sus siglas en inglés) y hasta televisión digital.

Para el uso de estos dispositivos se han desarrollado varias plataformas, o sistemas operativos móviles. Entre las más usadas se encuentran Android (de Google), iOS (de Apple), Symbian (de Nokia), BlackBerry OS (de BlackBerry) y Windows Phone (de Microsoft) (Hyers, 2014).

En la sociedad del conocimiento los avances tecnológicos dan respuesta a las necesidades

que plantea la sociedad. Así, en una sociedad en movimiento surgen las tecnologías móviles para dar respuesta a las necesidades de la población.

1.3.1 El sistema operativo Android

A nivel internacional Android es un Sistema Operativo (SO) basado en Linux, de código abierto y provee una capa de bibliotecas escritas en C y C++, además de un marco de trabajo para el desarrollo de aplicaciones. Originalmente, este marco de trabajo estaba solo basado en Java y posteriormente, se liberó para aplicaciones nativas en C, aunque no es recomendable a menos que se necesite utilizar de manera excesiva al microprocesador. Incluye además una suite de aplicaciones iniciales (Firtman, 2013).

Google es su principal impulsor, pues es él quien lo mantiene, pero en realidad para hacerlo más abierto y que no sea un sistema operativo manejado por una sola empresa, creó una organización sin fines de lucro denominada la Alianza Abierta de Dispositivos (*Open Handset Alliance*). Incluye a más de 65 empresas del sector de fabricación de dispositivos móviles, entre los que se encuentran a Google, fabricantes de equipos Motorola, HTC, LG, Samsung, Alcatel; operadores móviles como Telefónica, Movistar, T-Mobile, Sprint, China Mobile y fabricantes de microprocesadores y placas de video Intel, nVidia, Qualcomm, Texas Intruments, Huawei, entre otros. Estas empresas son unas de las pocas que forman la alianza, responsables de mantener a Android como sistema operativo (Firtman, 2013).

Como se evidencia en el anexo 2, el sistema operativo Android domina las ventas del mercado, producto a que se desarrolla de forma abierta, lo que permite acceder tanto al código fuente como a la lista de incidencias, donde se pueden ver problemas aún no resueltos y reportar problemas nuevos, además Android como software libre permite la distribución de las mejoras hechas en su código fuente de forma gratuita, lo que logra que todos los usuarios que lo utilicen salgan beneficiados (Hyers, 2014).

Por estas razones el autor de la presente investigación centra el interés en el sistema operativo Android y en los teléfonos móviles inteligentes por los beneficios que ofrecen. Además de poder funcionar como un mando a distancia, al contar con una aplicación capaz de activar y desactivar vía Bluetooh el prototipo de sistema de alarma. Para la implementación de esta aplicación se utilizará el IDE de desarrollo Android Studio. Entorno de desarrollo que se encuentra disponible para desarrolladores de forma gratuita, además fue diseñado específicamente para desarrollar soluciones para Android y su uso es recomendado por Google.

Android Studio presenta las siguientes características (Android, 2014):

- ✓ Renderización en tiempo real.
- ✓ Consola de desarrollador: consejos de optimización, ayuda para la traducción y estadísticas de uso.
- ✓ Refactorización específica de Android y arreglos rápidos.
- ✓ Herramientas para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- ✓ Plantillas para crear diseños comunes de Android y otros componentes.

Lenguaje de programación: Java

Al elegir el IDE de desarrollo Android Studio para la creación de la aplicación Android surgió la necesidad de utilizar el lenguaje de programación Java. Se selecciona porque es el utilizado por el IDE y por la máquina virtual Dalvik, que es una optimización de la máquina virtual de Java.

Java es un lenguaje de programación y la primera plataforma informática creada por la empresa Sun Microsystems en 1995. Se ejecuta en más de 850 millones de ordenadores personales de todo el mundo y en miles de millones de dispositivos, como dispositivos móviles y aparatos de televisión (Groussard, 2009).

La creación de este lenguaje y plataforma se inspiró en las funcionalidades de otros lenguajes tales como C++, Eiffel, SmallTalk, Objetive C, Cedar/Mesa, Ada, Perl. El resultado es una plataforma y un lenguaje idóneo para el desarrollo de aplicaciones sencillas, orientadas a objetos, distribuidas, interpretadas, robustas, seguras, independientes de las arquitecturas, portables, eficaces, multitareas y dinámicas (Groussard, 2009).

1.4 Tecnologías y metodología

Además de la implementación de la aplicación, se hizo necesario realizar una selección de tecnologías necesarias para la conformación de la propuesta de solución, lo que garantiza un resultado recomendable por parte del desarrollador.

1.4.1 Selección del microcontrolador

Para la selección del microcontrolador se realizó una comparación de la familia Arduino, cuyos resultados se encuentran en la tabla 1, y se precisaron las características técnicas y generales de varios microcontroladores.

Tabla 1. Familia Arduino (Arduino.org, 2016)

Nombre	Procesador	Operating/Input Voltage	Velocidad de CPU	Analog In/Out	Digital IO/PWM	Flash [KB]	USB	UART
ArduinoBT	ATmega328P	5 V / 2.5-12 V	16 MHz	6/0	14/6	32	-	1
Due	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12/2	54/12	512	2 Micro	4
Esplora	ATmega32U4	5 V / 7-12 V	16 MHz	-	-	32	Micro	-
Ethernet	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/4	32	Regular	-
Fio	ATmega328P	3.3 V / 3.7-7 V	8 MHz	8/0	14/6	32	Mini	1
Gemma	ATtiny85	3.3 V / 4-16 V	8 MHz	1/0	3/2	8	Micro	0
Leonardo	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	32	Micro	1
LilyPad	ATmega168V ATmega328P	2.7-5.5 V / 2.7-5.5 V	8MHz	6/0	14/6	16	-	-
LilyPad SimpleSnap	ATmega328P	2.7-5.5 V / 2.7-5.5	8 MHz	4/0	9/4	32	-	-

		V						
LilyPad USB	ATmega32U4	3.3 V / 3.8-5 V	8 MHz	4/0	9/4	32	Micro	-
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	256	Regular	4
Mega ADK	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	256	Regular	4
Micro	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	32	Micro	1
Mini	ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/6	32	-	-
Nano	ATmega168 ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/6	16 32	Mini	1
Pro	ATmega168 ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	16 32	-	1
Pro Mini	ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	16	-	1
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	32	Regular	1
Yùn	ATmega32U4 AR9331 Linux	5 V	16 MHz 400MHz	12/0	20/7	32 64MB	Micro	1
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	256	2 Micro	2

Se optó por elegir el microcontrolador ATmega32U4 que forma parte del módulo Arduino Micro, debido a que sus especificaciones se ajustan a las del proyecto, que cuenta con un tamaño reducido y un costo asequible de $25. Además, dispone de 20 pines de entrada/salida, 12 de ellas de entradas analógicas en lugar de 6 disponibles en modelos anteriores y puede operar con un voltaje de entrada comprendido entre los 5 y 12 volt.

Figura 5. Arduino Micro

1.4.2 Sensores

Según el diccionario de la lengua española un sensor es un dispositivo que detecta una determinada acción externa, temperatura, presión y la transmite adecuadamente (Española, 2016).

Sensor de inclinación

Se optó por elegir el sensor de inclinación de mercurio (ver figura 6) que forma parte del módulo de Arduino para poder detectar una posible inclinación del auto.

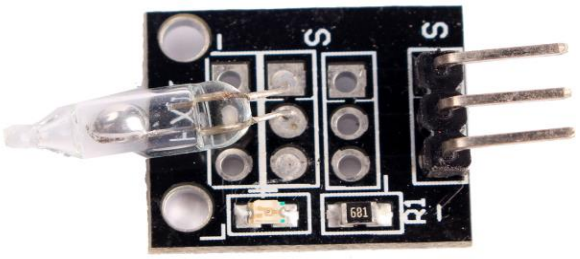


Figura 6. Sensor de inclinación

Sensor de campo magnético

Se optó por elegir el sensor de campo magnético (ver figura 7) del módulo Arduino para comprobar que todas las puertas al igual que el capó y el maletero del carro se mantuvieran cerrados para evitar la entrada al auto.

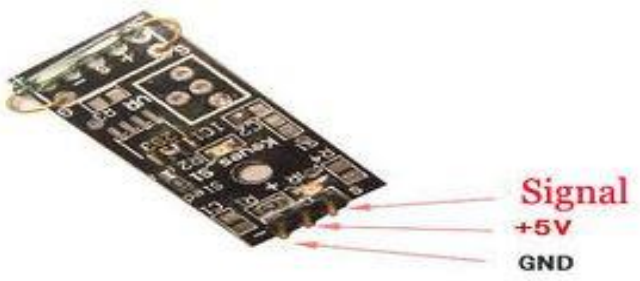


Figura 7. Sensor de campo magnético

1.4.3 Módulo de relé para Arduino

Para realizar la activación y desactivación de las luces y el claxon del auto se decidió utilizar un módulo de relé (ver figura 8), que funciona como un interruptor controlado por el microcontrolador. El módulo de relé es capaz de controlar un circuito de salida de mayor potencia que el de entrada como es el circuito de 12V de un automóvil, mientras que el del microcontrolador es de 5V. Sus características técnicas son:

- ✓ Señal de control: TTL.
- ✓ Tensión máxima de conmutación: 30 VDC/250VAC
- ✓ Carga nominal: 10A 28 VDC/125VAC
- ✓ Dimensiones: 4 x 2,7 x 1,8 cm
- ✓ Rango de temperatura: - 25ºC a +70ºC.
- ✓ Corriente (Bobina): 85 mA.
- ✓ LED indicador.
- ✓ Tipo de entrada: Digital (3,3V a 5V).

Figura 8. Módulo de relés para Arduino

1.4.4 Módulo de Bluetooth para Arduino

El Bluetooth es un estándar de comunicación inalámbrica que permite la transmisión de datos a través de radiofrecuencia en la banda de 2,4 GHz. Existen muchos módulos de Bluetooth, pero los más utilizados son los módulos de JY-MCU, debido a su bajo costo de adquisición y localización en el mercado. Son módulos pequeños y con un consumo muy bajo, que permiten agregar funcionalidades Bluetooth al Arduino. Estos módulos contienen un chip con una placa de desarrollo con los pines necesarios para la comunicación serie.

Según (DIYMakers, 2012) existen dos modelos de módulos Bluetooth: el HC-05 que puede ser maestro/esclavo, y el HC-06 que solo puede actuar como esclavo. La diferencia entre maestro y esclavo es que en modo esclavo es el dispositivo quien se conecta al módulo, mientras que en modo maestro es el módulo quien se conecta con un dispositivo. Físicamente, los dos módulos son muy parecidos, solo varían algunas conexiones. Se contará con los siguientes pines:

- ✓ **Vcc**: alimentación del módulo entre 3,6V y 6V.
- ✓ **GND**: la masa del módulo.
- ✓ **TXD**: transmisión de datos.
- ✓ **RXD**: recepción de datos a un voltaje de 3,3V.
- ✓ **KEY**: poner a nivel alto para entrar en modo configuración del módulo (solo el modelo HC-05)
- ✓ **STATE**: para conectar un led de salida para visualizar cuando se comuniquen datos.

Para la elaboración del prototipo de alarma, el autor optó por elegir el modelo HC-06 que se muestra en la figura 9, que permite tener una comunicación única entre el usuario y el prototipo de sistema de alarma, pues actúa como esclavo y brinda una gran seguridad al contar necesariamente para su conexión con un pin de seguridad.

Figura 9. Módulo de Bluetooth HC-06 para Arduino

1.4.5 Metodología para el desarrollo de la solución

Cada proceso de desarrollo de una solución informática, debe estar regido por alguna metodología de desarrollo de software que guíe los procesos y permita tener un registro detallado del avance de la investigación y la creación de la aplicación. Estas se dividen en dos tipos: las robustas y las ágiles. Las metodologías robustas o pesadas están concebidas para aplicaciones grandes que requieran gran nivel de detalles y que posean un tiempo largo de creación, estas metodologías generan gran cantidad de documentación debido a que es posible que un proceso de desarrollo de software pase por varios equipos de trabajo. Su funcionamiento está basado en iteraciones que cubren todo el ciclo de vida del proyecto, lo que permite que se minimicen los riesgos, y se desarrolle el software en corto tiempo.

Por lo planteado anteriormente se decidió utilizar una adaptación de la metodología Proceso Unificado Ágil (AUP por sus siglas en inglés) para los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI) en su escenario número 4, debido a que esta es recomendable para proyectos de mediano alcance, en los que el negocio a informatizar esté bien definido y el equipo de desarrollo esté siempre acompañado por el cliente para convenir los detalles de los requerimientos y así poder implementarlos, probarlos y validarlos. De este modo se le da cumplimiento a las buenas prácticas que define CMMI-DEV v1.3.

1.5 Conclusiones parciales

Con el desarrollo del marco teórico se organizó y encaminó el trabajo hacia el desarrollo del objetivo general. Se profundizó en el estudio de algunos conceptos fundamentales para el desarrollo del sistema de alarma para automóviles basado en dispositivos Arduino. Además se realizó un estudio de los sistemas similares a nivel internacional, lo que posibilitó la selección de las funcionalidades, tecnologías libres a utilizar como: Arduino 1.6.1 y Android Studio, los lenguajes de programación Arduino y Java y la metodología de desarrollo AUP-UCI en su escenario número 4, propuesta para la solución del problema.

CAPÍTULO 2: CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA

En el presente capítulo se precisan un conjunto de elementos para conformar la propuesta de solución de la presente investigación. Se analizan las características, componentes y particularidades; además de describir el funcionamiento general del sistema de alarma para automóviles que se desea implementar. Se realiza una descripción de la arquitectura propuesta para la solución, así como el uso de patrones presentes en el mismo. Se emplea la metodología de desarrollo seleccionada y se define la documentación significativa que esta propone en sus primeras fases del ciclo de desarrollo.

2.1 Propuesta de solución

Para satisfacer las necesidades del cliente y dar solución a la problemática de la investigación se decide desarrollar firmware, a partir del uso la plataforma de hardware Arduino y una aplicación Android para la conformación del sistema de alarma. Entre las bondades de la solución se aprecia que permitirá a los usuarios controlar la activación y desactivación de la alarma remotamente, mediante un mando infrarrojo o por una aplicación Android vía Bluetooth, la que contará con el privilegio de poder habilitar y deshabilitar el mando infrarrojo en caso de una posible pérdida de este.

El firmware permitirá al microcontrolador dominar una serie de sensores como son los de inclinación, el de campo magnético para controlar el acceso al interior del vehículo, el maletero y el capó. Además controlará dos relés utilizados para prender y apagar las luces y el claxon en caso de que ocurra una anomalía, un módulo de Bluetooth para establecer la conexión con un móvil Android que permitirá la activación y desactivación mediante la aplicación desarrollada y un sensor infrarrojo para captar la señal infrarroja trasmitida desde el mando, lo que permitirá a su vez, la activación y desactivación del sistema de alarma.

Para su desarrollo se emplearán los IDEs de desarrollo Arduino 1.6.1 y Android Studio a mediante los lenguajes de programación Arduino y Java respectivamente. Además el proceso de desarrollo de las distintas funcionalidades del sistema será guiado por la adaptación de la metodología AUP propuesta para los proyectos productivos de la UCI (AUP-UCI), y se utilizará su escenario número 4, lo que modela el sistema con Historias de Usuarios (HU).

2.2 Requerimientos del sistema

Los requerimientos de software se clasifican en funcionales y no funcionales. Los funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son las propiedades o cualidades del producto.

2.2.1 Requerimientos funcionales

Como bien define la metodología aplicada, una de las formas de encapsular los requerimientos funcionales son las HU. A continuación, se presentan las 11 HU desarrolladas.

- ✓ HU1: Activar alarma por aplicación.
- ✓ HU2: Desactivar alarma por aplicación.
- ✓ HU3: Activar alarma por mando infrarrojo.
- ✓ HU4: Desactivar alarma por mando infrarrojo.
- ✓ HU5: Detectar acceso no autorizado.
- ✓ HU6: Detectar inclinación del vehículo.
- ✓ HU7: Habilitar mando infrarrojo.
- ✓ HU8: Deshabilitar mando infrarrojo.
- ✓ HU9: Alertar mediante señal de luces.
- ✓ HU10: Alertar mediante señal del claxon.
- ✓ HU11: Mostrar el estado del sistema de alarma.

2.2.2 Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema. Para la realización del sistema propuesto se describen los siguientes requerimientos no funcionales.

Tabla 2. Requerimiento no funcional 1

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	
Objetivo	La aplicación Android debe contar con una interfaz de fácil entendimiento para que los usuarios inexpertos puedan interactuar fácilmente.
Origen	Externo
Artefacto	
Entorno	Ejecución de la aplicación Android
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Interacción con el sistema	
Medida de respuesta	

Tabla 3. Requerimiento no funcional 2

Atributo de Calidad	Software
Sub-atributos/Sub-características	
Objetivo	La aplicación Android debe poder instalarse en Android 4.0 IceCream o superior.
Origen	Externo
Artefacto	
Entorno	Ejecución de la aplicación Android
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Interacción con el sistema	
Medida de respuesta	

Tabla 4. Requerimiento no funcional 3

Atributo de Calidad	Seguridad
Sub-atributos/Sub-características	
Objetivo	Se debe garantizar la seguridad de la conexión vía Bluetooth con el sistema de alarma.
Origen	Externo
Artefacto	
Entorno	Conexión de la aplicación Android con el sistema de alarma.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Interacción con el sistema	
Medida de respuesta	

2.3 Descripción de las historias de usuarios

Las historias de usuarios permiten encapsular los documentos de especificación funcional. Estas HU son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar (Pérez, 2012). Las principales características de las historias de usuario son según (Beck, 1999) independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables.

Para la elaboración de las historias de usuario se realizaron diferentes entrevistas con los miembros de la línea Sistemas Empotrados perteneciente al Centro de Informática Industrial (CEDIN) de la Facultad 5 de la UCI. Si bien el cliente no fue quien escribió personalmente las HU, fue quien esbozó su contenido, asignó la prioridad y tuteló la redacción de cada una de ellas, por lo que su redacción fue clara y breve.

El cliente asignó la prioridad de cada una teniendo en cuenta la necesidad de su pronta implantación y su criticidad en el flujo de eventos, mientras el equipo de desarrollo revisó la prioridad, analizó la dependencia entre ellas y asignó el costo de cada una, este último se traduce en las semanas para su desarrollo. Se sugiere dividir las HU en más pequeñas si estas se demoran más tiempo en desarrollarse de lo planificado.

Las HU se representan mediante tablas, que contienen las siguientes secciones:

- ✓ Código: las siglas de HU más un número consecutivo, este permite identificar la historia.
- ✓ Nombre: nombre que la identifica.
- ✓ Referencia: es el conjunto de códigos de las diferentes HU de las cuales depende la que actualmente se encuentra en desarrollo.
- ✓ Prioridad: esta característica es dada por el cliente con los valores: alta, media o baja en dependencia de la importancia y orden en que desean que sean implementadas.
- ✓ Iteración Asignada: número de la iteración en la que se desarrollará la HU.
- ✓ Puntos Estimados: tiempo estimado en semanas que se le asignará.
- ✓ Descripción: breve descripción del proceso que define la historia.
- ✓ Observaciones: alguna acotación importante de señalar acerca de la historia.

Se elaboraron 11 HU, y se tomaron para los puntos estimados, la unidad como una semana de trabajo. Se confeccionaron 4 de prioridad alta, 6 de prioridad media y 1 de prioridad baja.

Tabla 5: HU_1 Activar alarma por aplicación.

Historia de Usuario	
Código: HU_1	Nombre: Activar alarma por aplicación
Referencia: Ninguna	Prioridad: Alta
Iteración Asignada: 2	Puntos Estimados: 2.0
Descripción: La aplicación Android debe activar el sistema de alarma vía Bluetooth.	
Observaciones: 1. Debe permitir activar el Bluetooth del móvil.	
Prototipo de interfaz de usuario:	

Tabla 6: HU_2 Desactivar alarma por aplicación.

Historia de Usuario	
Código: HU_2	Nombre: Desactivar alarma por aplicación
Referencia: HU_1, HU_3	Prioridad: Alta
Iteración Asignada: 2	Puntos Estimados: 2.0
Descripción: La aplicación Android debe desactivar el sistema de alarma vía Bluetooth.	
Observaciones: 1. Debe permitir activar el Bluetooth del móvil.	
Prototipo de interfaz de usuario:	

Tabla 7: HU_3 Activar alarma por mando infrarrojo.

Historia de Usuario	
Código: HU_3	Nombre: Activar alarma por mando infrarrojo
Referencia: Ninguna	Prioridad: Alta
Iteración Asignada: 2	Puntos Estimados: 2.0
Descripción: El sistema de alarma debe permitir activarse por un mando a distancia.	
Observaciones: 1. Debe ser un mando infrarrojo.	
Prototipo de interfaz de usuario: Ninguna	

Tabla 8: HU_4 Desactivar alarma por mando infrarrojo.

Historia de Usuario	
Código: HU_4	Nombre: Desactivar alarma por mando infrarrojo
Referencia: HU_1, HU_3	Prioridad: Alta
Iteración Asignada: 2	Puntos Estimados: 2.0

Descripción: El sistema de alarma debe permitir desactivarse por un mando a distancia.
Observaciones: 1. Debe ser un mando infrarrojo.
Prototipo de interfaz de usuario: Ninguna

Tabla 9: HU_5 Detectar acceso no autorizado.

Historia de Usuario	
Código: HU_5	Nombre: Detectar acceso no autorizado
Referencia: Ninguna	Prioridad: Media
Iteración Asignada: 3	Puntos Estimados: 1.0
Descripción: El sistema de alarma debe detectar cualquier intento de acceso no autorizado al interior del vehículo.	
Observaciones:	
Prototipo de interfaz de usuario: Ninguna	

Tabla 10: HU_6 Detectar inclinación del vehículo.

Historia de Usuario	
Código: HU_6	Nombre: Detectar inclinación del vehículo.
Referencia: Ninguna	Prioridad: Media
Iteración Asignada: 3	Puntos Estimados: 1.0
Descripción: El sistema de alarma debe detectar cualquier intento de inclinación del vehículo.	
Observaciones:	
Prototipo de interfaz de usuario: Ninguna	

Tabla 11: HU_7 Habilitar mando infrarrojo.

Historia de Usuario	
Código: HU_7	Nombre: Habilitar mando infrarrojo
Referencia: Ninguna	Prioridad: Media
Iteración Asignada: 3	Puntos Estimados: 1.0
Descripción: La aplicación Android debe permitir habilitar el mando infrarrojo.	
Observaciones: 1. La aplicación móvil debe estar conectada al prototipo de sistema de alarma.	
Prototipo de interfaz de usuario:	
	

Tabla 12: HU_8 Deshabilitar mando infrarrojo.

Historia de Usuario	
Código: HU_8	Nombre: Deshabilitar mando infrarrojo
Referencia: Ninguna	Prioridad: Media
Iteración Asignada: 3	Puntos Estimados: 1.0
Descripción: La aplicación Android debe permitir habilitar el mando infrarrojo.	
Observaciones:	

1. La aplicación móvil debe estar conectada al prototipo de sistema de alarma.
Prototipo de interfaz de usuario:


Tabla 13: HU_9 Alertar mediante señal de luces.

Historia de Usuario	
Código: HU_9	Nombre: Alertar mediante señal de luces.
Referencia: Ninguna	Prioridad: Media
Iteración Asignada: 1	Puntos Estimados: 1.0
Descripción: El sistema de alarma debe alertar mediante la señal de las luces en caso de que se detecte algún incidente.	
Observaciones:	
Prototipo de interfaz de usuario: Ninguna	

Tabla 14: HU_10 Alertar mediante señal del claxon.

Historia de Usuario	
Código: HU_10	Nombre: Alertar mediante señal del claxon.
Referencia: Ninguna	Prioridad: Media
Iteración Asignada: 1	Puntos Estimados: 1.0
Descripción: El sistema de alarma debe alertar mediante la señal del claxon en caso de que se detecte algún incidente.	
Observaciones:	
Prototipo de interfaz de usuario: Ninguna	

Tabla 15: HU_11 Mostrar el estado del sistema de alarma.

Historia de Usuario	
Código: HU_11	Nombre: Mostrar el estado del sistema de alarma.
Referencia: HU_1, HU_2, HU_3, HU_4	Prioridad: Baja
Iteración Asignada: 1	Puntos Estimados: 1.0
Descripción: El sistema de alarma debe indicar el estado de la alarma mediante un led.	
Observaciones: El led estará encendido mientras el sistema de alarma esté activo.	
Prototipo de interfaz de usuario: Ninguna	

2.4 Plan de iteraciones

Después de identificar y redactar cada una de las HU, se debe elaborar el plan de iteraciones.

Cada HU se convierte en tareas específicas de programación, de la misma forma para cada HU se establecen las pruebas de aceptación. Las pruebas fallidas en el ciclo anterior son analizadas para evaluar su corrección para prever que no vuelvan a ocurrir.

El plan fue dividido en 3 iteraciones, para las que se desarrollaron partes funcionales del

sistema de alarma. A continuación, se describen cada una de las iteraciones anteriormente mencionadas:

- ✓ **Iteración 1:** para esta iteración se desarrollan las HU de la 9 a la 11 correspondientes a alertar mediante señal de luces, alertar mediante señal del claxon y mostrar el estado del sistema de alarma. Las HU seleccionadas para esta iteración permiten intentar alertar mediante la señal del claxon y de las luces en caso de que se dispare el sistema de alarma, además de indicar el estado del sistema mediante el led indicador. Esta última HU depende de las HU 1, 2, 3 y 4. Se le realizan las pruebas de aceptación y se hace una entrega funcional al finalizar la iteración.

- ✓ **Iteración 2:** para esta iteración se desarrollan las HU de la 1 a la 4 correspondientes con activar alarma por aplicación, desactivar alarma por aplicación, activar alarma por mando infrarrojo y desactivar alarma por mando infrarrojo. Para la implementación de estas se apoyan en las HU implementadas en la iteración anterior. Además en esta iteración aparte de las pruebas de aceptación de realizan pruebas de integración para verificar que no han ocurrido afectaciones en las historias anteriores. Al finalizar la iteración se realiza una entrega funcional, que permite activar y desactivar el sistema de alarma indicando su estado en un led.

- ✓ **Iteración 3:** para esta iteración se desarrollan las HU de la 5 a la 8 correspondientes con detectar acceso no autorizado y detectar inclinación del vehículo, habilitar mando infrarrojo y deshabilitar mando infrarrojo. En esta iteración al igual que en la anterior, se realizan las pruebas de aceptación e integración y al finalizar la iteración se realiza la entrega final de la propuesta de solución.

  En cada iteración, para aproximar el tiempo de ejecución se tomó como medida, que cada semana contaba de 5 días (lunes, martes, miércoles, jueves y viernes) en los que se trabajaban 6 horas. En la tabla 16, se muestra el plan de iteraciones y se incorpora el tiempo estimado para cada una de las iteraciones y las HU que se van a desarrollar.

Tabla 16: Plan de iteraciones.

Iteración	Historias de usuario	Puntos de estimación (semanas)
1	Alertar mediante señal de luces	3.0
	Alertar mediante señal del claxon	
	Mostrar el estado del sistema de alarma	
2	Activar alarma por aplicación	6.0
	Desactivar alarma por aplicación	
	Activar alarma por mando infrarrojo	
	Desactivar alarma por mando infrarrojo	
3	Detectar acceso no autorizado	6.0
	Detectar inclinación del vehículo	
	Habilitar mando infrarrojo	
	Deshabilitar mando infrarrojo	

2.5 Plan de entregas

El plan de entregas establece que HU son agrupadas para conformar una entrega, y el orden de las mismas. Este plan es el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores y gerentes). El cronograma se realiza sobre la base de las estimaciones de tiempos de desarrollo realizadas por los desarrolladores.

A partir del plan de iteraciones analizado en el acápite anterior, y en correspondencia con el mismo se realiza el plan de entregas, que se muestra en la tabla 17. En él, se proponen 2 versiones funcionales y una última entrega de iteraciones de la propuesta de solución.

Tabla 17: Plan de entregas.

Historias de usuario	1ra Iteración 22 de enero del 2016	2da Iteración 18 de marzo del 2016	3ra Iteración 20 de mayo del 2016
Alertar mediante señal de luces	V1.0		
Alertar mediante señal del claxon			

Mostrar el estado del sistema de alarma			
Activar alarma por aplicación		V1.1	
Desactivar alarma por aplicación			
Activar alarma por mando infrarrojo			
Desactivar alarma por mando infrarrojo			
Detectar acceso no autorizado			V1.2
Detectar inclinación del vehículo			
Habilitar mando infrarrojo			
Deshabilitar mando infrarrojo			

2.6 Tareas de ingeniería o programación

Asociado a cada iteración, se encuentra la planificación de las tareas de ingeniería o programación. Cada una se caracteriza por estar asociada a una historia de usuario y varias tareas de ingeniería pueden pertenecer a una historia de usuario. Para la confección de cada una de las tareas se utilizaron tablas cuyo modelo cuenta con los siguientes campos:

- ✓ No. de tarea: numeración continua que identifica a la tarea
- ✓ No. de HU: número de la HU a la que pertenece
- ✓ Nombre de la tarea: identificación literal de la tarea
- ✓ Tipo de tarea: tipo de tarea, dígase diseño, desarrollo, prueba
- ✓ Puntos estimados: representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizará para su realización
- ✓ Fecha inicio: fecha estimada de inicio de realización
- ✓ Fecha fin: fecha estimada de fin de realización
- ✓ Descripción: se describe en qué consiste la tarea y qué elementos deben cumplirse para declarar la tarea terminada.

Seguidamente, se presentan las tareas de ingeniería perteneciente a cada historia de usuario:

Tabla 18: Tarea de ingeniería 1.1 de la historia de usuario Alertar mediante señal de luces.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 9
Nombre de la tarea: Encender y apagar luces	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.9
Fecha inicio: 1/1/2016	**Fecha fin:** 6/1/2016
Descripción: Se realiza la implementación de un *firmware* que le permita al microcontrolador activar y desactivar un relé utilizado para prender y apagar las luces del vehículo.	

Tabla 19: Tarea de ingeniería 1.2 de la historia de usuario Alertar mediante señal de luces.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 9
Nombre de la tarea: Conexiones del relé de luces.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.1
Fecha inicio: 7/1/2016	**Fecha fin:** 7/1/2016
Descripción: Se realiza las conexiones pertinentes del relé de las luces con las del microcontrolador.	

Tabla 20: Tarea de ingeniería 1.1 de la historia de usuario Alertar mediante señal del claxon.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 10
Nombre de la tarea: Encender y apagar claxon	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.9
Fecha inicio: 8/1/2016	**Fecha fin:** 13/1/2016
Descripción: Se realiza la implementación de un *firmware* que le permita al microcontrolador activar y desactivar un relé utilizado para prender y apagar el claxon del vehículo.	

Tabla 21: Tarea de ingeniería 1.2 de la historia de usuario Alertar mediante señal del claxon.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 10
Nombre de la tarea: Conexión de relé de claxon	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.1
Fecha inicio: 14/1/2016	**Fecha fin:** 14/1/2016
Descripción: Se realizan las conexiones pertinentes del relé del claxon con las del microcontrolador.	

Tabla 22: Tarea de ingeniería 1.1 de la historia de usuario Mostrar el estado del sistema de alarma.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 11
Nombre de la tarea: Encender y apagar led indicador	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.9
Fecha inicio: 15/1/2016	**Fecha fin:** 20/1/2016
Descripción: Se realiza la implementación de un *firmware* que le permita al microcontrolador prender y apagar un led para indicar con este el estado de la alarma.	

Tabla 23: Tarea de ingeniería 1.2 de la historia de usuario Mostrar el estado del sistema de alarma.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 11
Nombre de la tarea: Conexión de led indicador	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.1
Fecha inicio: 21/1/2016	**Fecha fin:** 21/1/2016
Descripción: Se realizan las conexiones pertinentes del led indicador con las del microcontrolador.	

Tabla 24: Tarea de ingeniería 1.1 de la historia de usuario Activar alarma por aplicación.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 1
Nombre de la tarea: Desarrollo de la interfaz de la aplicación Android.	

Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 25/1/2016	**Fecha fin:** 29/1/2016
Descripción: Se realiza el diseño e implementación de la vista para activar y desactivar la alarma por una aplicación Android.	

Tabla 25: Tarea de ingeniería 1.2 de la historia de usuario Activar alarma por aplicación.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 1
Nombre de la tarea: Desarrollo de conexión vía Bluetooth de la aplicación Android.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 1/2/2016	**Fecha fin:** 5/2/2016
Descripción: Se realiza la implementación de la clase para la conexión de la aplicación Android por Bluetooth para la activación de la alarma.	

Tabla 26: Tarea de ingeniería 1.1 de la historia de usuario Desactivar alarma por aplicación.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 2
Nombre de la tarea: Desarrollo de la interfaz de la aplicación Android.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 8/2/2016	**Fecha fin:** 12/2/2016
Descripción: Se realiza el diseño e implementación de la vista para activar y desactivar la alarma por una aplicación Android.	

Tabla 27: Tarea de ingeniería 1.2 de la historia de usuario Desactivar alarma por aplicación.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 2
Nombre de la tarea: Desarrollo de conexión vía Bluetooth de la aplicación Android.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 15/2/2016	**Fecha fin:** 19/2/2016
Descripción: Se realiza la implementación de la clase para la conexión de la aplicación Android por Bluetooth para la desactivación de la alarma.	

Tabla 28: Tarea de ingeniería 1.1 de la historia de usuario Activar alarma por mando infrarrojo.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 3
Nombre de la tarea: Activar alarma por mando infrarrojo.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 22/2/2016	**Fecha fin:** 26/2/2016
Descripción: Se realiza la implementación de un *firmware* que le permita al microcontrolador leer una tecla de un mando infrarrojo.	

Tabla 29: Tarea de ingeniería 1.2 de la historia de usuario Activar alarma por mando infrarrojo.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 3
Nombre de la tarea: Desarrollo de conexión infrarroja del microcontrolador.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 29/2/2016	**Fecha fin:** 4/3/2016
Descripción: Se realizan las conexiones pertinentes del sensor infrarrojo con las del microcontrolador.	

Tabla 30: Tarea de ingeniería 1.1 de la historia de usuario Desactivar alarma por mando infrarrojo.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 4
Nombre de la tarea: Desactivar alarma por mando infrarrojo.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 7/3/2016	**Fecha fin:** 11/3/2016
Descripción: Se realiza la implementación de un *firmware* que le permita al microcontrolador leer una tecla de un mando infrarrojo.	

Tabla 31: Tarea de ingeniería 1.2 de la historia de usuario Desactivar alarma por mando infrarrojo.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 4
Nombre de la tarea: Desarrollo de conexión infrarrojo del microcontrolador.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0

Fecha inicio: 14/3/2016	**Fecha fin:** 18/3/2016
Descripción: Se realizan las conexiones pertinentes del sensor infrarrojo con el microcontrolador.	

Tabla 32: Tarea de ingeniería 1.1 de la historia de usuario Detectar acceso no autorizado.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 5
Nombre de la tarea: Mantener puertas, capó y maletero cerrado.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.9
Fecha inicio: 21/3/2016	**Fecha fin:** 25/3/2016
Descripción: Se realiza la implementación de un *firmware* que le permita al microcontrolador comprobar que no exista un acceso no autorizado al interior del vehículo.	

Tabla 33: Tarea de ingeniería 1.2 de la historia de usuario Detectar acceso no autorizado.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 5
Nombre de la tarea: Conexiones de los sensores magnéticos.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.1
Fecha inicio: 28/3/2016	**Fecha fin:** 28/3/2016
Descripción: Se realizan las conexiones pertinentes de los sensores magnéticos con el microcontrolador.	

Tabla 34: Tarea de ingeniería 1.1 de la historia de usuario Detectar inclinación del vehículo.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 6
Nombre de la tarea: Detectar inclinación.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.9
Fecha inicio: 4/4/2016	**Fecha fin:** 7/4/2016

Descripción: Se realiza la implementación de un *firmware* que le permita al microcontrolador controlar que el vehículo no sea remolcado ni inclinado.

Tabla 35: Tarea de ingeniería 1.2 de la historia de usuario Detectar inclinación del vehículo.

Tarea de Ingeniería	
No. de tarea: 1.2	**No. de HU:** 6
Nombre de la tarea: Conexiones de los sensores de inclinación.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.1
Fecha inicio: 8/4/2016	**Fecha fin:** 8/4/2016
Descripción: Se realizan las conexiones pertinentes de los sensores de inclinación con el microcontrolador.	

Tabla 36: Tarea de ingeniería 1.1 de la historia de usuario Habilitar mando infrarrojo.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:** 7
Nombre de la tarea: Desarrollo de la funcionalidad habilitar mando infrarrojo.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 11/4/2016	**Fecha fin:** 15/4/2016
Descripción: Se implementan los métodos para la habilitación del mando en la aplicación Android.	

Tabla 37: Tarea de ingeniería 1.1 de la historia de usuario Deshabilitar mando infrarrojo.

Tarea de Ingeniería	
No. de tarea: 1.1	**No. de HU:**8
Nombre de la tarea: Desarrollo de la funcionalidad deshabilitar mando infrarrojo.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio: 18/4/2016	**Fecha fin:** 22/4/2016
Descripción: Se implementan los métodos para la deshabilitación del mando infrarrojo en la aplicación Android.	

2.7 Descripción de la arquitectura

La arquitectura del software alude a la estructura global del software y a las formas en que la estructura proporciona la integridad conceptual de un sistema. En su forma más simple, la arquitectura es la estructura jerárquica de los componentes del programa, la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes. Sin embargo, en un sentido más amplio, los componentes se pueden generalizar para representar los elementos principales del sistema y sus interacciones (Pressman, 2010).

La propuesta de solución está conformada por una placa Arduino micro y una aplicación Android que se conectan vía Bluetooth. Con el objetivo de exponerla claramente se presenta la figura 10, que muestra la estructura de la arquitectura, los elementos que la componen y la forma en que interactúan entre ellos.

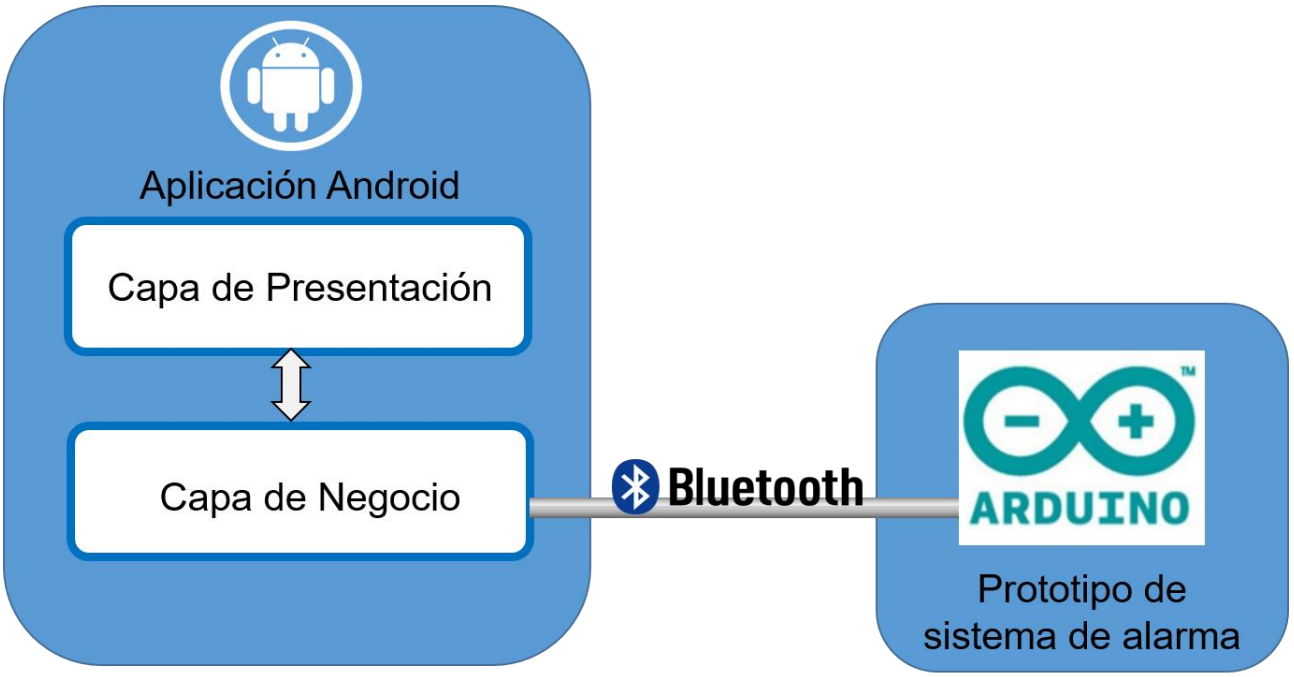


Figura 10. Representación de la arquitectura del sistema (Fuente: elaboración propia).

La placa Arduino es la encargada de monitorear todos los actuadores y sensores conectados a esta, a partir de una programación estructurada. El programa diseñado para ejecutarse sobre una plataforma Arduino se compone por tres secciones:

- ✓ **La sección de declaraciones de variables globales**: no tiene ningún tipo de símbolo delimitador de inicio y fin. En esta sección se realizaron las declaraciones de variables

necesarias para la programación de la propuesta de solución.

- ✓ **La sección llamada "void setup()":** dentro de sus llaves se escribieron las instrucciones que se ejecutan una única vez, es decir las preconfiguraciones iniciales que se realizan al encender o resetear la placa Arduino, y se declaran los pines utilizados de la placa Arduino tanto de entrada como de salida.

- ✓ **La sección llamada "void loop()":** se ejecuta después de la sección llamada "void setup()" infinitas veces hasta el apagado o reiniciado de la placa. Se implementó el programa que se ejecutará para posibilitar la comunicación de la placa tanto por Bluetooth como por infrarrojo para la activación y desactivación de la propuesta de solución y monitoreo de los sensores.

Por otra parte, la creación de la aplicación Android encargada de activar y desactivar el prototipo de sistema de alarma, proporciona el beneficio de habilitar y deshabilitar el mando infrarrojo en caso de una posible pérdida. Para su desarrollo se utilizó una programación orientada a objetos y se empleó una arquitectura en capas. Este patrón define cómo organizar el modelo de diseño en capas, y cómo pueden estar físicamente distribuidas, lo que precisa que los componentes de una capa solo pueden hacer referencia a componentes en capas inmediatamente inferiores (Androideity, 2011).

Principales estilos del patrón arquitectónico:

- ✓ Arquitecturas de dos capas.

- ✓ Arquitecturas de tres capas.

- ✓ Arquitecturas de n capas.

Para el diseño de la aplicación móvil se utilizó específicamente el estilo arquitectónico de dos capas y se empleó la siguiente descomposición:

1. **Capa de presentación:** es la única que interactúa directamente con el usuario y se le presenta el control de la propuesta de solución. Contiene una interfaz gráfica que posibilita activar el Bluetooth del móvil, además de la activación y desactivación del

prototipo de sistema de alarma vía Bluetooth. Esta capa, solo interactúa con la capa de negocio, que es su inferior inmediata.

2. **Capa de negocio:** también conocida como lógica de negocio, es la que recibe las peticiones de la capa de presentación, realiza la comunicación con el microcontrolador y envía las peticiones del usuario. Esta capa interactúa con la capa de presentación para recibir sus solicitudes y presentarle los resultados.

2.8 Patrones de diseño

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en muchas situaciones a la hora de realizar una aplicación orientada a objetos. Además, son los encargados de identificar clases, instancias, roles, colaboraciones y la distribución de responsabilidades (Fernandez, 2009).

Estos ofrecen las siguientes ventajas:

- ✓ Proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Están basados en la recopilación del conocimiento de los expertos en desarrollo del software.
- ✓ Es una experiencia real, probada y funcional.
- ✓ Facilitan la localización de los objetos que forman el sistema, la determinación de la granularidad adecuada, el aprendizaje y la comunicación entre programadores.
- ✓ Especifican interfaces para las clases e implementaciones al menos parciales.

2.8.1 Patrones GRASP

Patrones de Asignación de Responsabilidad (GRASP, por sus siglas en inglés *General Responsibility Assignment Software Patterns*) representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Bajo acoplamiento: el acoplamiento es una medida de la fuerza en la que una clase está relacionada a otras clases. Una clase con bajo o débil acoplamiento no depende de muchas otras. Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma, que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, lo que potencia la reutilización, y disminuye la dependencia

entre las clases. La solución es asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible, sin comprometer la funcionalidad. En la aplicación se evidencia este patrón en la clase *DeviceList.java,* porque no tienen dependencia de ninguna otra clase y en *AlarmaControl.java* porque tiene la menor dependencia posible de otras clases.

Creador: para guiar la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El diseño bien asignado permitirá soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El propósito fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. En la aplicación se utiliza este patrón en la clase *DeviceList.java*.

2.8.2 Patrones GoF

Los patrones “Banda de los Cuatro” (GoF, por sus iniciales en inglés *Gang of Four*), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Existen tres tipos de patrones: los de creación que abstraen el proceso de creación de instancias estructurales que se ocupan de cómo las clases y objetos son utilizados para componer estructuras de mayor tamaño y de comportamiento que se refieren a los algoritmos y a la asignación de responsabilidades entre objetos.

A continuación, se identifican los patrones utilizados en el desarrollo de la aplicación, de acuerdo a su clasificación.

State (Comportamiento): este patrón consiste en permitir que un objeto modifique su comportamiento cada vez que cambie su estado interno. Esto se puede observar en los diferentes métodos asíncronos (ver figura 11) que realizan operaciones en la aplicación. Las mismas tienen varios comportamientos en dependencia del estado en que se encuentra el hilo de ejecución.

```
@Override
protected void onPostExecute(Void result
{
    super.onPostExecute(result);

    if (!ConnectSuccess)
    {
        msg("Falla de conexión.");
        finish();
    }
```

Figura 11. Ejemplo del patrón de comportamiento aplicado en un método asíncrono.

Estructurales: los patrones estructurales describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos. De los patrones definidos como estructurales se utiliza el adaptador.

- ✓ Adaptador (*Adapter*): convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permite que cooperen clases que de otra manera no podrían por tener interfaces incompatibles. Dentro de la plataforma móvil, el adaptador puede emplearse para generar los elementos de componentes visuales como las listas expandible, que requieren de un adaptador para crear los grupos padres y los hijos, que han de ser mostrados al usuario. Este patrón se utilizó en la clase *DeviceList.java*.

2.9 Conclusiones parciales

Con la realización del presente capítulo se define la línea base para el desarrollo del sistema de alarma para automóvil. Se identificaron las funcionalidades del sistema con detalles específicos de su desarrollo y se definieron los patrones arquitectónicos y de diseño usados con el objetivo de lograr una mayor organización en los elementos que conforman la propuesta de solución. Además, se realizó el plan de entregas donde se indicaron las funcionalidades creadas para cada versión del sistema de alarma para automóvil y las fechas en las que se publicaron estas versiones. Se obtuvo una planificación del tiempo de desarrollo de las iteraciones y el orden en que se implementaron las funcionalidades de acuerdo con la prioridad que le fue asignada.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo se describen las dos últimas fases de la metodología aplicada, en la que se analizan tanto los resultados del proyecto como su ejecución, y se realizan las actividades formales de cierre del proyecto. Además se describen la selección de las pruebas, así como su realización para la validación de la solución.

3.1 Pruebas

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación (Pressman, 2010). La metodología aplicada define que no debe existir ninguna funcionalidad en el programa que no haya sido probada. El equipo de desarrollo estará siempre acompañado por el cliente para convenir los detalles de los requerimientos y así poder implementarlos, probarlos y validarlos. De esta forma se brinda un resultado más completo para un producto final de manera creciente. AUP-UCI propone tres tipos de pruebas que se exponen a continuación (Sánchez, 2014):

- ✓ **Pruebas internas:** se verifica el resultado de la implementación al probar cada construcción, y al incluir tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como diseños de casos de prueba, listas de chequeo y de ser posibles, componentes de prueba ejecutables para automatizar las pruebas.
- ✓ **Pruebas de liberación:** pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
- ✓ **Pruebas de Aceptación:** es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las que el software fue construido.

3.1.1 Pruebas de aceptación

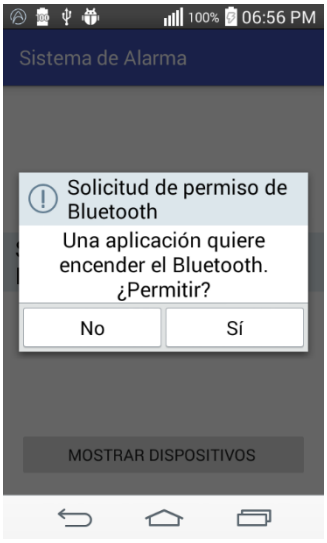
Las pruebas de aceptación son creadas sobre la base de las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. Las pruebas de aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos (Pujol Méndez, y otros, 2014).

Las pruebas de aceptación significan la satisfacción del cliente con el producto desarrollado; es por esto, que el cliente es la persona adecuada para diseñarlas. Se tomó como criterio de aprobación de cada iteración el 100% de los casos de prueba exitosos para pasar de iteración. El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. Para la representación de las pruebas de aceptación se definieron los siguientes elementos:

- ✓ **Código de la prueba:** representa al caso de prueba, incluye el número de HU y de la prueba.
- ✓ **Nombre de Historia de Usuario:** nombre de la historia de usuario.
- ✓ **Descripción de la prueba:** acción que debe realizar el sistema.
- ✓ **Condiciones de ejecución:** describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- ✓ **Entrada /Pasos de ejecución:** incluye las entradas necesarias para realizar el sistema, además de los pasos para realizar el caso de prueba.
- ✓ **Resultado Esperado:** descripción de la respuesta del sistema ante el caso de prueba.
- ✓ **Resultado Obtenido:** respuesta visual del sistema después de realizar el caso de prueba.
- ✓ **Evaluación de la Prueba:** clasificación de la prueba en satisfactoria o insatisfactoria.

A continuación, se muestran las pruebas de aceptación realizadas a las 11 HU.

Tabla 38: Prueba 1 de la historia de usuario Activar alarma por aplicación.

Código de la Prueba: HU1_P1	**Nombre de Historia de Usuario:** Activar alarma por aplicación.
Descripción de la Prueba: La prueba consiste en conectarse al dispositivo de Bluetooth HC-06 sin estar habilitado el Bluetooth en el móvil. El objetivo de esta prueba es ver la reacción de la aplicación móvil cuando no esté habilitado el Bluetooth en el móvil.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado para la realización de esta prueba.	
Entrada / Pasos de ejecución: • Desactivar el Bluetooth en el móvil • Abrir la aplicación móvil	
Resultado Esperado: La aplicación móvil lanza una solicitud de permiso para activar el Bluetooth del móvil.	
Resultado Obtenido: 	

Evaluación de la Prueba: Satisfactoria

Tabla 39: Prueba 2 de la historia de usuario Activar alarma por aplicación.

Código de la Prueba: HU1_P2	**Nombre de Historia de Usuario:** Activar alarma por aplicación.
Descripción de la Prueba: La prueba consiste en activar la propuesta de solución desde la aplicación móvil sin estar encendido el prototipo de sistema de alarma. El objetivo de esta prueba es ver la reacción de la aplicación móvil cuando no puede establecer la conexión con el dispositivo de Bluetooth conectado al microcontrolador.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar desconectado para la realización de esta prueba.	
Entrada / Pasos de ejecución: • Abrir la aplicación móvil • Aceptar la solicitud de permiso de Bluetooth • Listar los dispositivos vinculados • Seleccionar el HC-06 • Presionar el botón ON/OFF	
Resultado Esperado: La aplicación móvil lanza un mensaje con el que indica la falla de conexión.	
Resultado Obtenido:	

Evaluación de la Prueba: Satisfactoria

Tabla 40: Prueba 1 de la historia de usuario Desactivar alarma por aplicación.

Código de la Prueba: HU2_P1	**Nombre de Historia de Usuario:** Desactivar alarma por aplicación.
Descripción de la Prueba: La prueba consiste en desactivar la propuesta de solución desde la aplicación móvil sin estar encendido el prototipo de sistema de alarma. El objetivo de la prueba es ver la reacción de la aplicación móvil cuando no puede establecer la conexión con el dispositivo de Bluetooth conectado al microcontrolador.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar desconectado para la realización de la prueba.	
Entrada / Pasos de ejecución: • Abrir la aplicación móvil • Aceptar la solicitud de permiso de Bluetooth • Listar los dispositivos vinculados • Seleccionar el HC-06	

• Presionar el botón ON/OFF
Resultado Esperado: La aplicación móvil lanza un mensaje con el que indica la falla de conexión.
Resultado Obtenido: 
Evaluación de la Prueba: Satisfactoria

Tabla 41: Prueba 1 de la historia de usuario Activar alarma por mando infrarrojo.

Código de la Prueba: HU3_P1	**Nombre de Historia de Usuario:** Activar alarma por mando infrarrojo.
Descripción de la Prueba: La prueba consiste en activar la propuesta de solución desde el mando infrarrojo presionando otro botón y enviando un código diferente. El objetivo de la prueba es ver la reacción del prototipo de sistema de alarma cuando se le envíe otro comando por infrarrojo.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado para la realización de la prueba.	

Entrada / Pasos de ejecución: Presionar otro botón del mando infrarrojo
Resultado Esperado: El prototipo de sistema de alarma no reacciona cuando se le envía el otro código por infrarrojo.
Resultado Obtenido:
Evaluación de la Prueba: Satisfactoria

Tabla 42: Prueba 1 de la historia de usuario Desactivar alarma por mando infrarrojo.

Código de la Prueba: HU4_P1	**Nombre de Historia de Usuario:** Desactivar alarma por mando infrarrojo.
Descripción de la Prueba: La prueba consiste en desactivar la propuesta de solución desde el mando infrarrojo al presionar otro botón y al enviar un código diferente. El objetivo de la prueba es ver la reacción del prototipo de sistema de alarma cuando se le envíe otro comando por infrarrojo.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado para la realización de la prueba.	
Entrada / Pasos de ejecución: Presionar otro botón del mando infrarrojo	
Resultado Esperado: El prototipo de sistema de alarma no reacciona cuando se le envía el otro código por infrarrojo.	
Resultado Obtenido:	
Evaluación de la Prueba: Satisfactoria	

Tabla 43: Prueba 1 de la historia de usuario Detectar acceso no autorizado.

Código de la Prueba: HU5_P1	**Nombre de Historia de Usuario:** Detectar acceso no autorizado.
Descripción de la Prueba: La prueba consiste en aproximarle un imán al sensor magnético cuando la propuesta de solución esté desactivada. El objetivo de la prueba es ver la reacción del prototipo de sistema de alarma cuando se le aproxime el imán al estar desactivado.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado para la realización de la prueba.	
Entrada / Pasos de ejecución: • Desactivar la propuesta de solución • Aproximarle el imán al sensor magnético	
Resultado Esperado: El prototipo de sistema de alarma no reacciona cuando se le aproxima el imán al estar desactivado.	
Resultado Obtenido:	
Evaluación de la Prueba: Satisfactoria	

Tabla 44: Prueba 1 de la historia de usuario Detectar inclinación del vehículo.

Código de la Prueba: HU6_P1	**Nombre de Historia de Usuario:** Detectar inclinación del vehículo.
Descripción de la Prueba: La prueba consiste en inclinar la propuesta de solución cuando esté desactivada. El objetivo de esta prueba es ver la reacción del prototipo de sistema de alarma cuando se inclina al estar desactivado.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado para la realización de la prueba.	
Entrada / Pasos de ejecución: • Desactivar la propuesta de solución • Inclinar el prototipo de sistema de alarma	
Resultado Esperado: El prototipo de sistema de alarma no reacciona cuando se le inclina al estar desactivado.	
Resultado Obtenido:	
Evaluación de la Prueba: Satisfactoria	

Tabla 45: Prueba 1 de la historia de usuario Habilitar mando infrarrojo.

Código de la Prueba: HU7_P1	**Nombre de Historia de Usuario:** Habilitar mando infrarrojo.
Descripción de la Prueba: La prueba consiste en habilitar el mando infrarrojo desde la aplicación móvil. El objetivo de la prueba es ver la reacción del prototipo de sistema de alarma.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado para la realización de esta prueba.	
Entrada / Pasos de ejecución: • Abrir la aplicación móvil • Aceptar la solicitud de permiso de Bluetooth • Listar los dispositivos vinculados • Seleccionar el HC-06 • Presionar el botón Deshabilitar mando • Presionar el botón Habilitar mando	
Resultado Esperado: El prototipo de sistema de alarma habilita el mando infrarrojo.	
Resultado Obtenido:	

Evaluación de la Prueba: Satisfactoria

Tabla 46: Prueba 1 de la historia de usuario Deshabilitar mando infrarrojo.

Código de la Prueba: HU8_P1	**Nombre de Historia de Usuario:** Deshabilitar mando infrarrojo.
Descripción de la Prueba: La prueba consiste en deshabilitar el mando infrarrojo desde la aplicación móvil. El objetivo de esta prueba es ver la reacción del prototipo de sistema de alarma.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado para la realización de la prueba.	
Entrada / Pasos de ejecución: • Abrir la aplicación móvil • Aceptar la solicitud de permiso de Bluetooth • Listar los dispositivos vinculados • Seleccionar el HC-06 • Presionar el botón Deshabilitar mando	

Resultado Esperado: El prototipo de sistema de alarma deshabilita el mando infrarrojo.
Resultado Obtenido: 
Evaluación de la Prueba: Satisfactoria

Tabla 47: Prueba 1 de la historia de usuario Alertar mediante señal de luces.

Código de la Prueba: HU9_P1	**Nombre de Historia de Usuario:** Alertar mediante señal de luces.
Descripción de la Prueba: La prueba consiste en hacer que el prototipo de sistema de alarma se dispare mediante uno de sus sensores. El objetivo de esta prueba es ver si la propuesta de solución es capaz de encender las luces del auto en caso de un intento de acceso al auto.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado y activado para la realización de esta prueba.	
Entrada / Pasos de ejecución:	

• Activar la propuesta de solución • Inclinar el prototipo de sistema de alarma
Resultado Esperado: El prototipo de sistema de alarma enciende el relé encargado de encender las luces del auto.
Resultado Obtenido:
Evaluación de la Prueba: Satisfactoria

Tabla 48: Prueba 1 de la historia de usuario Alertar mediante señal del claxon.

Código de la Prueba: HU10_P1	**Nombre de Historia de Usuario:** Alertar mediante señal del claxon.
Descripción de la Prueba: La prueba consiste en hacer que el prototipo de sistema de alarma se dispare mediante uno de sus sensores. El objetivo de esta prueba es ver si la propuesta de solución es capaz de encender el claxon del auto en caso de un intento de acceso al auto.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado y activado para la realización de esta prueba.	
Entrada / Pasos de ejecución: • Activar la propuesta de solución • Inclinar el prototipo de sistema de alarma	
Resultado Esperado: El prototipo de sistema de alarma enciende el relé encargado de activar el claxon del auto.	

Resultado Obtenido:
Evaluación de la Prueba: Satisfactoria

Tabla 49: Prueba 1 de la historia de usuario Mostrar el estado del sistema de alarma.

Código de la Prueba: HU11_P1	**Nombre de Historia de Usuario:** Mostrar el estado del sistema de alarma.
Descripción de la Prueba: La prueba consiste en activar y desactivar la alarma por la aplicación móvil. El objetivo de esta prueba es ver si la propuesta de solución es capaz de indicar su estado a través de un led.	
Condiciones de Ejecución: El prototipo de sistema de alarma debe estar conectado.	
Entrada / Pasos de ejecución: • Activar la propuesta de solución mediante la aplicación móvil	
Resultado Esperado: El led indicador se enciende cuando la propuesta de solución se activa y se apaga en caso contrario.	
Resultado Obtenido:	
Evaluación de la Prueba: Satisfactoria	

En la primera iteración se realizó una etapa de pruebas, en ella se realizaron 6 casos de pruebas en los que se obtuvieron resultados satisfactorios. Como se cumplió con el criterio de aceptación, se decidió no realizar otra etapa de pruebas y, por tanto, pasar a la segunda iteración. Se realizaron dos etapas de pruebas con un total de 10 casos de pruebas por etapas. En la primera etapa se detectaron 2 no conformidades, lo que representa un 20% de resultados insatisfactorios. Después de la corrección de los mismos se determinó realizar la segunda etapa en los que no se encontraron errores, lo que representa un 100% de los resultados

satisfactorios. Estos resultados cumplen con el criterio de aceptación, por lo que se decide pasar a la tercera iteración. No se encontraron errores para un 100% de resultados satisfactorios. Los resultados se presentan en la figura 11.

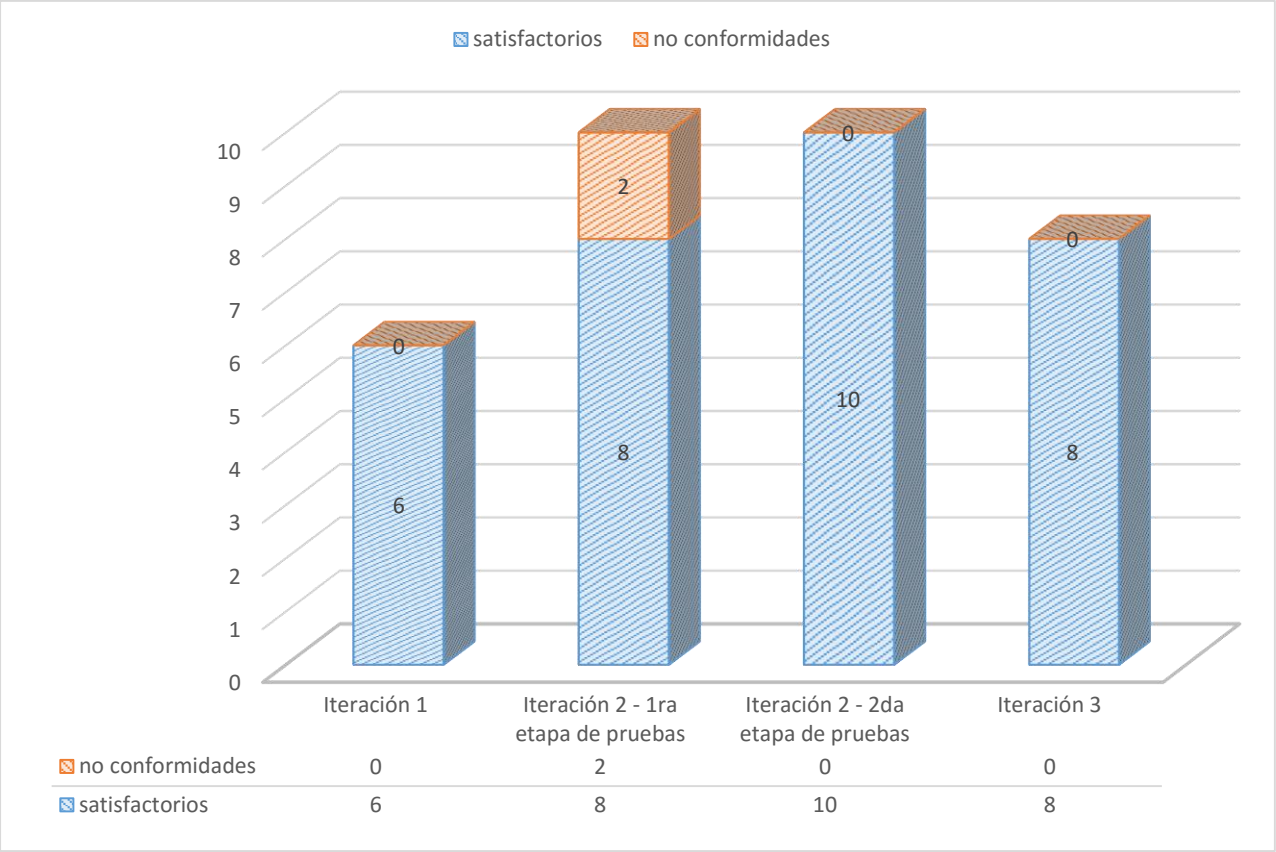


Figura 11: Resultados de las pruebas de aceptación por iteración.

3.2 Conclusiones parciales

Con la ejecución de las fases de implementación y prueba se elaboraron los diferentes casos de prueba, a los que se le realizaron pruebas de aceptación y se tomó como criterio de aprobación el 100% de los casos de prueba satisfactorios por iteración. Se corrigieron todas las no conformidades encontradas, además se determinó que la propuesta de solución desarrollada cumple con todos los requerimientos acordados con el cliente.

CONCLUSIONES

La investigación permitió arribar a las siguientes conclusiones:

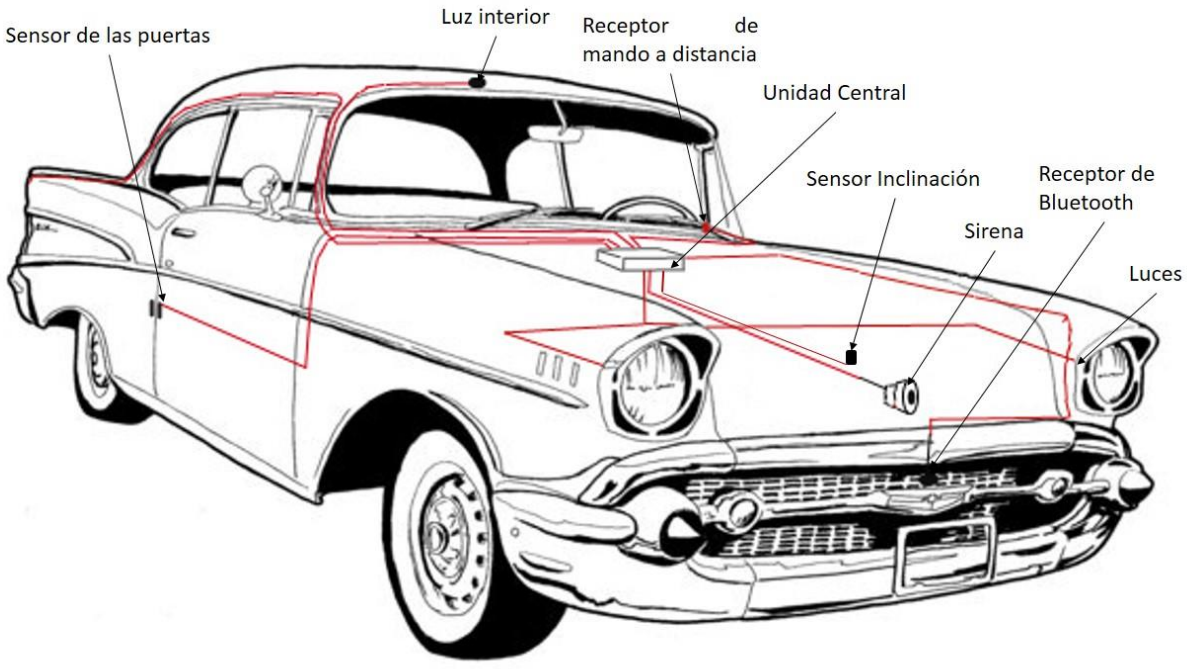
- ✓ El diagnóstico inicial apoyado en la aplicación de los métodos científicos permitió identificar las funcionalidades y las tecnologías libres a utilizar en el desarrollo de la propuesta de solución.
- ✓ Se obtuvo un sistema de alarma para automóvil basado en dispositivos Arduino que tiene como objetivo contribuir a la seguridad de los autos en Cuba, y que permite alertar mediante la señal de la sirena y de luces en caso de un intento de acceso no autorizado al interior del vehículo.
- ✓ Mediante la realización de las pruebas de aceptación se comprobó que la propuesta de solución cumple con los requerimientos pactados con el cliente y le proporciona conformidad. Además, permitió el cumplimiento del objetivo trazado inicialmente en la investigación.

RECOMENDACIONES

El objetivo de este trabajo fue alcanzado, pero durante su desarrollo surgieron nuevas ideas que serían recomendables tener en cuenta para su futuro perfeccionamiento como son:

- ✓ Desarrollar un mando infrarrojo. Para ello, se considera necesario reprogramar el microcontrolador al que se le debe modificar el código que se recibirá por el nuevo mando infrarrojo.
- ✓ Cambiar los sensores de inclinación de mercurio por acelerómetros. Para ello, se debe reprogramar el microcontrolador al que se le debe cambiar los valores obtenidos por los nuevos sensores.

Anexo 1. Esquema de componentes del sistema de alarma



Anexo 1: Esquema de componentes del sistema de alarma (Fuente: Elaboración propia).

Anexo 2: Comparación entre sistemas operativos para dispositivos móviles

Global Smartphone Operating System Shipments (Millions of Units)	Q2'13	Q2'14
Android	186.8	249.6
Apple iOs	31.2	35.2
Microsoft Windows Phone	8.9	8.0
BlackBerry OS	5.7	1.9
Others	0.5	0.5
Total	233.0	295.2

Global Smartphone Operating System Marketshare %	Q2'13	Q2'14
Android	80.2	84.6
Apple iOs	13.4	11.9
Microsoft Windows Phone	3.8	2.7
BlackBerry OS	2.4	0.6
Others	0.2	0.2
Total	100.0	100.2
Total Growth Year-over-Year %	48.9	26.7

Anexo 2: Comparación entre plataformas móviles (Hyers, 2014)

REFERENCIAS BIBLIOGRÁFICAS

Acosta, P .E. 2011. Tecnologías Móviles. *TICSMIELCA.* [Online] 2011. http://sites.google.com/site/ticsmielca/tecnoligias-moviles.

Android. 2014. Android Studio Overview | Android Developer. [Online] 2014. [Cited: 2 15, 2016.] http://developer.android.com/tools/studio/index.html.

Androideity. 2011. Androideity. *Androideity.* [Online] 2011. [Cited: 4 1, 2016.] http://www.androideity.com/.

Arduino. 2015. www.arduino.cc. *www.arduino.cc.* [Online] 2015. [Cited: 12 18, 2015.] http://www.arduino.cc.

Arduino.org. 2016. http://www.arduino.org. *arduino.org.* [Online] 1 9, 2016. [Cited: 1 9, 2016.] http://www.arduino.org.

Artengo, Óscar Torrente. 2013. *Arduino: Curso práctico de formación.* s.l. : RC LIBROS, 2013.

Banzi, Massimo. 2012. *Introducción a Arduino.* s.l. : ANAYA MULTIMEDIA, 2012.

Beck, K. 1999. *Extreme Programming Explained.* 1999.

CEASI, «Seguridad integral,» 2013. [En línea]. Available: http://www.nuestraseguridad.gob.ec/sites/default/files/comision_INFORME%20

OCTUBRE.pdf. 2013. Seguridad integral. *CEASI.* [Online] 2013. http://www.nuestraseguridad.gob.ec/sites/default/files/comision_INFORME%20 OCTUBRE.pdf.

DIYMakers. 2012. DIYMakers. *DIYMakers.* [Online] 2012. [Cited: 4 9, 2016.] http://www.diymakers.es/.

Edgardo Luna Melara, Dennis, Arnulfo Vasquez Flores, Oscar and Merlos, Yamileth Martinez. 2013. *Smart Security Car.* 2013.

ELECTRONICA ESTUDIO.COM. [Online] http://www.electronicaestudio.com/microcontrolador.htm.

Española, Real Academia. 2016. Real Academia Española. [Online] 1 19, 2016. [Cited: 1 19, 2016.] http://dle.rae.es.

Fernandez, Luis H. 2009. Patrones de diseno. [Online] 2009. [Cited: 4 5, 2016.] http://software.guisho.com/patronesde-diseno.

Firtman, M. 2013. Video2Brain. Curso Básico Android. [Online] Video2Brain, 2013. [Cited: 3 11, 2016.] http://www.video2brain.com..

Foundation, Free Software. 2015. Free Software Foundation. *Free Software Foundation.* [Online] 2015. [Cited: 12 17, 2015.] http://www.fsf.org.

Garcia, S. M y Contreras. 2014. *Tecnologías Móviles.* España , Universidad de Salamanca : s.n., 2014.

Graverán, Dariel Noa. 2012. *Control remoto de dispositivos en tiempo real usando el marco de trabajo jWebSocket y la plataforma de hardware Arduino .* Artemisa : s.n., 2012.

Groussard, Thierry. 2009. Recursos Informaticos: Java 6. Barcelona : Editions ENI, 2009.

Hyers, K. 2014. Strategy Analytics: Global Smartphone Shipments. [Online] 2014. http://www.strategicanalytics.com.

PCC. 2011. *"VI Congreso del Partido Comunista de Cuba. Lineamientos de la Política Económica y Social del Partido y la Revolución".* La Habana : s.n., 2011.

Pelaez, J. 2009. *Arquitectura basada en capas.* 2009.

Pérez, J. 2012. *Guía Comparativa de Metodologías Ágiles.* Segovia : Universidad de Valladolid, 2012.

Pressman, Roger S. 2010. *Ingeniería del Software.* Madrid : s.n., 2010. 7ma edición.

Pujol Méndez, Naivy and Echevarría Díaz, Jesús Miguel. 2014. *Sistema de Gestión para el Centro Internacional de Posgrado.* La Habana : s.n., 2014.

Rumbaugh, James y Jacobson, Ivar. 2007. *El Lenguaje Unificado de Modelado. Manual de Referencia.* 2007.

Salazar, A. 2014. *Tecnologías Móviles MIS.204.* 2014.

Sánchez, Tamara Rodríguez. 2014. *Metodología de desarrollo para la Actividad productiva de la UC.* La Habana : s.n., 2014. 1.2.