Universidad Autónoma del Estado de México
Centro Universitario UAEM Zumpango
Ingeniería en Computación

Tesis

"Desarrollo de un sistema domótico y aplicación para dispositivos móviles Android para control de luces"

Que para obtener el Título de
Ingeniero en Computación

Presenta:
Fredi Cedeño Enriquez

Asesor:
Dr. En C.C. Asdrúbal López Chau
Coasesor:
M. en C. Valentín Trujillo Mora

Septiembre 2018

RESUMEN

En esta tesis se presenta el desarrollo de un sistema domótico y una aplicación para dispositivos móviles Android. El propósito de haber desarrollado este sistema es controlar el alumbrado de una casa y la apertura / cierre de la puerta del garaje. El hardware es compuesto por una tarjeta Arduino, un módulo Bluetooth para la comunicación y un servomotor para controlar la puerta. La aplicación permite encender/apagar las luces y abrir/cerrar la puerta presionando botones que se muestran en la interfaz gráfica del usuario. A lo largo del documento se presenta una descripción completa de cómo se diseñó y configuró el sistema (tanto hardware como software). Las pruebas al sistema y los trabajos futuros se muestran en la última parte de este documento.

ABSTRACT

In this thesis we developed a domotic system and an application for Android mobile devices. The purpose of having developed this system is to control the lighting of a house, and the opening / closing of the garage door. The hardware is composed of an Arduino board, a Bluetooth module for communications and a servomotor for controlling the door. The application allows to turn the lights on/off and open/close a door by pressing buttons shown on the graphical user interface. Throug this document, a complete description of how the system (both hardware and software) was designed and configured is presented. Tests to the system and future works are show in the last part of this document.

AGRADECIMIENTOS

He llegado al final de mi trabajo de tesis y quiero agradecer con cariño y respeto a mis padres por haberme dado la vida, quienes me han dedicado la mayor parte de su tiempo, motivándome día con día para forjar mi camino de formación personal como profesional, y así llegar ser una gran persona que sirva de bien a la sociedad. Gracias a mis hermanos y sobrinos por su apoyo moral y sincero que me han brindado de forma incondicional.

Quiero agradecer enormemente a mi asesor Dr. Asdrúbal López Chau y coasesor M. en C. Valentín Trujillo Mora, quienes me brindaron su ayuda y tiempo para lograr llegar a la meta de finalizar correctamente mi trabajo de tesis. También a mis revisores, el M. en C. Manuel Almeida Vázquez y el Ing. Diego Armando Ramírez Avelino, por sus comentarios y puntos de vista que enriquecieron este trabajo.

Agradezco a la Universidad Autónoma del Estado de México por haberme abierto sus puertas y brindado la oportunidad de estudiar, para alcanzar mi formación profesional. Finalmente agradezco a mi familia, amigos, compañeros y profesores por haberlos encontrado en esta etapa de mi vida y ser parte de ella.

MUCHAS GRACIAS

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO I

INTRODUCCIÓN

La integración de diversas tecnologías tales como dispositivos móviles, electrónicos, las telecomunicaciones y sistemas embebidos, han impulsado la incorporación de sistemas domóticos en una gran cantidad de hogares alrededor del mundo.

La domótica (en español) o domotique (en francés) es un término usado para hacer referencia a “casa inteligente”, el cual tiene origen en latín (“Domo” con casa) y (“tica” con informática, robótica y automática). La domótica es definida como la integración de tecnologías para automatizar viviendas.

En esta tesis se presenta un sistema domótico básico para el control del alumbrado y apertura-cierre de la puerta de acceso de la cochera, así como una revisión general del estado del arte de la domótica. La tesis está conformada por cuatro capítulos, los cuales se enlistan a continuación:

El Capítulo I presenta una descripción introductoria de este proyecto, se presenta el planteamiento del problema que dio paso al desarrollo a un sistema domótico para una casa habitación, los objetivos generales y específicos, para la creación del sistema presentado. También se incluye la justificación que permite mostrar el por qué fue pensado este proyecto, así como los alcances.

El Capítulo II presenta el marco teórico de las tecnologías de hardware y software empleadas en el sistema domótico desarrollado, como la tarjeta Arduino, el microservomotor SG90 9g y el módulo Bluetooth HC-05. Se destaca los antecedentes de estos dispositivos, de igual manera se exponen las características técnicas para el manejo correcto de estos mecanismos. Para la parte del software, se abarca una descripción del sistema operativo (SO) Android para dispositivos móviles, donde se desarrolló la aplicación que controla el sistema domótico. Esta parte de la investigación retoma la historia, tipo de arquitectura, versiones desde sus inicios hasta la actualidad de Android, entre otros.

El Capítulo III presenta el desarrollo del sistema domótico conformado por cada una de las partes del hardware y software que lo integran, se expone la lista de materiales utilizados para su desarrollo, asimismo, la arquitectura en la que fue basado. Por otra parte

se muestran imágenes del código de programación de la App y partes del Sketch (programa) para grabar la tarjeta de Arduino. En otra sección de este capítulo se destaca la forma de conectar el módulo Bluetooth a la unidad de control, y la configuración de control del servo, también es importante destacar que se muestra el diseño de la App en Android conforme a su diagrama de clases y la programación del Sketch de la tarjeta Arduino, conforme al diagrama de flujo de su funcionamiento.

El Capítulo IV presenta los resultados que demuestran la integración de la aplicación móvil con el hardware del sistema domótico integrado a una maqueta de una casa habitación, donde se realizaron pruebas. Tanto a la App controlando el encendido-apagado de las luces de la vivienda, al igual que la apertura-cierre de la pluma que resguarda la entrada y salida de la cochera. También, en este capítulo se muestra una conclusión que describe en general el logro obtenido al realizar la investigación para el desarrollo de este sistema domótico propuesto. Finalmente, el trabajo a futuro de este sistema inteligente presenta ideas claras para dar seguimiento a lo que se desea realizar para lograr transformar de un sistema domótico básico a un gran sistema domótico mucho más complejo que integre cualquier tipo de mecanismos tecnológicos y hasta el uso de Internet.

PLANTEAMIENTO DEL PROBLEMA

La domótica ha estado presente años atrás desde los primeros edificios automatizados en Estados Unidos con la tecnología X-10, el cual consistía en el control de la climatización y apertura- cierre de persianas en oficinas a través de un control.

En la actualidad, la domótica va en un constante crecimiento debido al gran desarrollo de las nuevas tecnologías, como los dispositivos móviles y el Internet de las cosas que en conjunto facilitan la realización de las actividades cotidianas.

El desarrollo del sistema domótico permite a las personas de tercera edad o aquellas que tenga alguna discapacidad motora mejorar su calidad de vida, también impacta en las que no cuentan con suficiente tiempo o están en constante presión, y en ocasiones olvidan realizar actividades cotidianas cómo apagar las luces, cerrar la puerta, entre otras, las cuales no realizan por diferentes motivos, un sistema domótico les permite controlar diversas actividades a distancia, o realizarlas desde el lugar que se encuentran sin tener que desplazarse, por lo cual impacta en tiempo y su economía.

Es por ello que este sistema domótico propuesto es de bajo costo y fácil de implementar en una casa habitación, si se piensa en el mantenimiento es muy económico al igual que de vida duradera. La domótica permite integrar sensores y cámaras controladas a través de aplicaciones desarrolladas para dispositivos móviles y a la vez hacer del uso de Internet.

En este trabajo se desarrolló un sistema básico de domótica para el encendido y apagado de las luces de una casa habitación, además de controlar apertura y cierre de la puerta de la cochera, con ello se apoya la calidad de vida de las personas de la tercera edad y personas que presentan un tipo de discapacidad motora además de personas que por sus actividades o condiciones físicas se les dificulta realizarlas.

OBJETIVOS

GENERAL

Diseñar e implementar una aplicación para teléfono inteligente con sistema operativo Android para controlar un prototipo de sistema domótico de desarrollo propio, con la finalidad de automatizar el encendido-apagado de luces, así como la apertura-cierre de la puerta de la cochera de una casa habitación a escala (maqueta).

ESPECÍFICOS

1. Realizar una investigación documental sobre domótica, así como de los componentes principales de este tipo de sistemas.
2. Diseñar e implementar el hardware para poder controlar el alumbrado de una casa habitación y la puerta de la cochera, con conectividad inalámbrica a través de Bluetooth.
3. Diseñar una aplicación para dispositivos inteligentes con sistema operativo Android, para controlar el hardware del sistema propuesto.
4. Realizar pruebas de funcionamiento al sistema desarrollado, tanto al hardware como a la aplicación que lo controla.
5. Describir las tecnologías involucradas en el desarrollo del hardware y de la aplicación implementada.
6. Integrar la aplicación junto con el hardware del sistema de automatización para conformar el sistema a la casa habitación (maqueta).
7. Realizar las pruebas de funcionalidad del sistema integrado.

JUSTIFICACIÓN

Las nuevas tecnologías en conjunto con las herramientas de electrónica hacen más fácil realizar actividades de la vida cotidiana, en lugares como casas, edificios, oficinas, entre otros, a razón de aumentar la calidad de vida de los usuarios, esto lo conocemos como domótica.

Gran parte de la gente desconoce el tema de la domótica (casas inteligentes) o tecnologías que integran un sistema de automatización, esta investigación aborda información relacionada a las casas inteligentes para trasmitirla a los interesados en el tema.

El uso y aplicación de la domótica impacta directamente al estilo de vida de las personas que presentan alguna discapacidad motora, y gente de la tercera edad.

El implementar un sistema domótico en una vivienda puede ayudar a reducir el consumo de energía eléctrica y con este tipo de sistemas puede ejecutar actividades a distancia.

El presente trabajo aporta la integración de dispositivos móviles, herramientas de control y electrónica, las cuales fusionadas logran apoyar a mejorar la calidad de vida y resolver algunos problemas de las personas que necesitan de apoyo para realizar sus actividades cotidianas.

Se pretende que este trabajo sirva como base para el desarrollo de la domótica en la región y logre un impacto social en su implementación.

ALCANCE

1. Con esta investigación se abarca el desarrollo de un sistema domótico, el cual pretende aumentar la comodidad de las personas al controlar a distancia el apagado-encendido de las lámparas de una casa habitación.
2. Al controlar de manera remota la apertura y cierre de la pluma de acceso a la cochera mejora la comodidad en comparación a las casas comunes.
3. Al emplear un sistema domótico en un hogar aumenta el valor de la vivienda.
4. Los sistemas domóticos se pueden integrar fácil con el Internet de las cosas.
5. Ahorro de energía eléctrica, al controlar mejor el alumbrado.

CAPÍTULO II TECNOLOGÍAS RELACIONADAS

Domótica

Desde los inicios de la humanidad el hombre siempre ha tenido como objetivo mejorar la comodidad y el confort de su hogar, para hacerlo además, un lugar donde pueda sentirse seguro, sin correr ningún riesgo. Por ello, siempre ha buscado la manera de crear mecanismos y nuevas tecnologías para satisfacer sus necesidades, un ejemplo de ello es la domótica.

La domótica surge aproximadamente en la década de 1970, cuando en Estados Unidos empiezan a automatizar los primeros edificios con tecnología X-10, que era un protocolo de comunicación de manejo a control remoto de dispositivos eléctricos, esta tecnología fue creada en Escocia por la empresa "Pico Electronics of Glenrothes" para manejar electrodomésticos, manipular la calefacción y climatización, con el fin de obtener ahorro de energía eléctrica y dinero [1][2].

En el año de 1998, en Francia se empieza a emplear la palabra **domotique** a razón de su traducción **Domótica** en España, ya que su significado tiene origen en latín "domus" (relación de domo con casa) y la palabra griega "tica" (relacionada con "automática" que funciona de forma autónoma) [2][3].

La domótica en términos generales es la forma de emplear tecnologías de automatización en una vivienda, cómo la electrónica, robótica, informática, telecomunicaciones y electricidad. Beneficiando a usuarios en seguridad, confort y ahorro en el uso de energía eléctrica [4].

Componentes de un sistema domótico

Un sistema domótico está conformado por tres elementos indispensables, como los sensores (dispositivos de entrada), unidad de control o (nodos) y los actuadores, ver Figura 1. A continuación se da una descripción de cada uno de estos componentes [2][4][5][6][7]:

- **Sensores:** son dispositivos que interactúan con el entorno del mundo real, cuya función es recolectar información que por consiguiente es transformada en señales digitales o analógicas que posteriormente son retomadas por la unidad de control para procesar y realizar la toma de decisiones.
- **Unidad de control o nodos:** Es la parte encargada del sistema domótico que realiza el trabajo de almacenar la información recolectada por los sensores, para ser

procesada y enviada a través del bus de comunicación a hacia los actuadores quienes la retoman para ejecutar acciones.

Las unidades de control integran un microcontrolador o microprocesador en su hardware de este modo desempeñan la función de compilar le secuencia de líneas de código o programa que el usuario ha grabado en su memoria flash.

- **Actuadores:** Los actuadores son mecanismos que hacen interactuar la unidad de control con el mundo real. Posteriormente estos dispositivos ejecutan órdenes del controlador para desempeñar funciones, por ejemplo; el accionamiento de un relevador, el giro de un servomotor, la activación de un pistón, o simplemente manipular el regulador de iluminación de una casa.

Figura 1. Componentes de un sistema domótico [8].

Ventajas de un sistema domótico

Entre las ventajas de los sistemas domóticos, con respecto a las casas sin algún tipo de automatización, se pueden mencionar entre las siguientes [7][9][10]**:**

- **Aumento de Seguridad:** es gracias a la integración de sensores a un sistema inteligente que puede detectar intrusos por las noches y en el caso de las tuberías es más eficaz localizar donde está una fuga de agua, gas, o humo de un incendio.
- **Confort:** un sistema domótico ofrece la ventaja de tener una mayor comodidad, a causa de realizar el control de actividades rutinarias de forma automática y a distancia, como el caso típico de prender o apagar el foco del pasillo, lámparas de las habitaciones o de otros electrodomésticos del resto de la vivienda, haciéndolas fáciles y placenteras al usuario.
- **Ahorro de energía eléctrica:** al emplear un sistema domótico a un hogar o edificio, no solo se benefician las personas, también se favorece el medio ambiente, puesto que un sistema inteligente gestiona el uso de la energía eléctrica, ayudando a combatir el calentamiento global.
- **La vida de un sistema domótico es duradera y una buena inversión:** Los sistemas domóticos tienen varios años de vida útil, y requieren poco mantenimiento.
- **Mayor valor monetario del hogar con un sistema domótico:** al integrar un sistema de automatización a una vivienda el precio crece en el mercado inmobiliario, otorgando la facilidad de ser vendida más rápido gracias a que el inmueble es atractivo y futurista teniendo características que se diferencian de la competencia.

Arduino

Arduino es una plataforma de código libre de (open-source), basada en hardware y software que son fáciles de aprender a usar, para gente profesional o cualquier sujeto que se interese en desarrollar este tipo de prototipos de electrónica como un pasatiempo [11][12]. Esta plataforma puede acoplarse a su entorno mediante la recepción de entradas y salidas, es capaz de influir a su alrededor con el control de luces, motores u otros módulos compatibles. El microcontrolador que está integrado en la tarjeta Arduino véase la Figura 2, se programa usando los IDE Arduino Programming Language (desarrollado en Wiring) y Development Environment (basado en Processing).

Los diseños de hardware de Arduino están disponibles para ser usados libremente ya que están a cargo de la licencia de **Open Source**. Estas tarjetas pueden ser ensambladas a mano o comprarse preensambladas [11][12]. Esta última opción es la más usada por la

mayoría de los usuarios. El software se puede descargar gratuitamente desde la dirección de Internet https://www.arduino.cc/ .

Figura 2. Tarjeta Arduino UNO.

Fuente: Imagen de elaboración propia.

Origen de Arduino

En el año 2005, el instituto de Diseño Interactivo (Ivrea) de Italia desarrolló Arduino. Por la necesidad de contar con una herramienta libre de utilizar y de bajo costo, puesto que la idea original era únicamente de uso interno para la institución, debido a que la institución se dedicaba a la experimentación de prototipos basados en microcontroladores de igual manera buscaban que estos dispositivos contaran con compatibilidad multiplataforma, optando que el usuario aprendiera fácil a trabajar con este tipo de artefacto.

Durante este mismo periodo el Instituto se vio forzado a cerrar sus puertas al proyecto, pero personas involucradas al no querer perder el avance del desarrollo de Arduino, acuerdan liberarlo y compartir a todo individuo que deseara implementar o proponer nuevas características, dando paso a la corrección de fallas para posteriormente perfeccionar lo que hoy en día es Arduino[12].

El equipo que creó Arduino estuvo conformado por cinco personas: Massimo Banzi (ex profesor de Ivrea) principal impulsor de la idea y diseño de Arduino; David Cuartielles (profesor universitario de Suecia); David Mellis (ex estudiante de Ivrea) actual miembro del instituto de investigación (High-Low Tech del Mit Media Lab), Tom lgoe (Escuela de Arte

Tisch de Nueva York), y el sujeto encargado de la fabricación de los prototipos de las placas, Gianluca Martino ,cuyo sitio oficial en la Web es http://www.smartprojects.it [12].

Modelos de Arduino

Hay diferentes modelos de tarjetas Arduino, sus similitudes y diferencias de características se muestran en la Tabla 1 [13]:

Tabla 1. Tipos de tarjetas Arduino

TIPO DE ARDUINO	CARACTERISTICAS
Arduino Mega 2560	• Microcontrolador: ATmega2560. • Voltaje de funcionamiento: 5 V. • Terminales I/O digitales: 54 (de los cuales 15 proveen salida PWM). • Terminales de entradas análogas: 16. • Corriente DC por cada terminal I/O: 40 mA. • Corriente DC en el terminal de 3.3 V: 50 mA. • Memoria Flash: 256 KB de los cuales 8 KB son utilizados por el bootloader. • SRAM: 8 KB (ATmega328). • EEPROM: 4 KB (ATmega328). • Velocidad del reloj: 16 MHz.
Arduino Mega	• Microcontrolador: ATmega2560. • Voltaje de funcionamiento: 5 V. • Terminales I/O digitales: 54 (de los cuales 15 proveen salida PWM). • Terminales de entradas análogas: 16. • Corriente DC por cada terminal I/O: 40 mA. • Corriente DC en el terminal de 3.3 V: 50 mA. • Memoria Flash: 256 KB de los cuales 8 KB son utilizados por el bootloader.

	• SRAM: 8 KB. • EEPROM: 4 KB. • Velocidad de reloj: 16 MHz.
Arduino Leonardo	• Microcontrolador: ATmega32u4. • Voltaje de funcionamiento: 5 V. • Terminales I/O digitales: 20. • Canales PWM: 7. • Terminales de entradas análogas: 12. • Corriente DC por cada terminal I/O: 40 mA. • Corriente DC en el terminal de 3.3 V: 50 mA. • Memoria Flash: 32 KB (ATmega32u4) de los cuales 4 KB son utilizados por el bootloader. • SRAM: 2 KB (ATmega32u4). • EEPROM: 1 KB (ATmega32u4). • Velocidad de reloj: 16 MHz.
Arduino Ethernet	• Microcontrolador: ATmega328. • Voltaje de funcionamiento: 5 V. • Terminales I/O digitales: 14 (de los cuales 4 proveen salida PWM). • Terminales de entradas análogas: 6. • Corriente DC por cada terminal I/O: 40 mA. • Corriente DC en el terminal de 3.3 V: 50 mA. • Memoria Flash: 32 KB (ATmega328) de los cuales 0.5 KB son utilizados por el bootloader. • SRAM: 2 KB (ATmega328). • EEPROM: 1 KB (ATmega328). • Velocidad de reloj: 16 MHz. • Controlador embebido EthernetW5100 TCP/IP. • Tarjeta MicroSD, con adaptadores activos de voltaje.
	• Microcontrolador: ATmega168.

Arduino Nano	• Voltaje de funcionamiento: 5 V. • Terminales I/O digitales: 14 (de los cuales 6 proveen salida PWM). • Terminales de entradas análogas: 8. • Corriente DC por cada terminal I/O: 40 mA. • Memoria Flash: 16 KB de los cuales 2 KB son utilizados por el bootloader. • SRAM: 1 KB. • EEPROM: 512 bytes. • Velocidad de reloj: 16 MHz.
Arduino Pro	• Microcontrolador: ATmega168. • Voltaje de funcionamiento: 3.3 V. • Terminales I/O digitales: 14 (de los cuales 6 proveen salida PWM). • Terminales de entradas análogas: 8. • Corriente DC por cada terminal I/O: 40 mA. • Memoria Flash: 16 KB de los cuales 2 KB son utilizados por el bootloader. • SRAM: 1 KB. • EEPROM: 512 bytes. • Velocidad de reloj: 8 MHz.
Arduino Pro Mini	• Microcontrolador: ATmega168. • Voltaje de funcionamiento: 3.3 V. • Terminales I/O digitales: 14 (de los cuales 6 proveen salida PWM). • Terminales de entradas análogas: 8. • Corriente DC por cada terminal I/O: 40 mA. • Memoria Flash: 16 KB de los cuales 2 KB son utilizados por el bootloader. • SRAM: 1 KB.

	• EEPROM: 512 bytes. • Velocidad de reloj: 8 MHz.

En este trabajo se utilizó la tarjeta Arduino Uno, por lo que se le dedica una sección para mencionar sus características también véase la Figura 3.

Arduino UNO

La tarjeta Arduino UNO tiene las siguientes características técnicas [12] [14][15]:

- Cuenta con un microcontrolador Atmega328P.
- Voltaje de entrada de 7 a 12v (recomendado).
- Voltaje de entrada de 8 a 20v (limite).
- Terminales de E / S digitales 14 (de las cuales 6 proporcionan salida PWM) “Pulse Width Modulation” (Modulación de Ancho de Pulsos).
- Terminales de E / S digitales de PWM 6.
- Terminales de entrada analógica 6.
- Corriente DC por Terminales E / S 20 mA.
- Corriente DC para 3.3V Terminal 50 mA.
- Memoria flash 32 KB (ATmega328P) de los cuales 0,5 KB utilizados por el gestor de arranque.
- SRAM 2 KB (ATmega328P).
- EEPROM 1 KB (ATmega328P).
- Velocidad de reloj 16 MHz.
- Led en terminal 13.
- Longitud 68.6 mm.
- Anchura 53.4 mm.
- Peso 25 g.
- RX (Recepción de Datos) y TX (Trasmisión de Datos): Se usan para transmisiones serie de señales TTL (Time To Live).
- Interrupciones externas: Las Terminales 2 y 3 están configurados para generar una interrupción en el atmega. Las interrupciones pueden dispararse cuando se encuentra un valor bajo en estas entradas y con flancos de subida o bajada de la entrada.

- PWM: Arduino dispone de 6 salidas destinadas a la generación de señales PWM de hasta 8 bits.
- SPI: Los terminales 10, 11, 12 y 13 pueden utilizarse para llevar a cabo comunicaciones SPI, que permiten trasladar información full dúplex en un entorno Maestro/Esclavo.

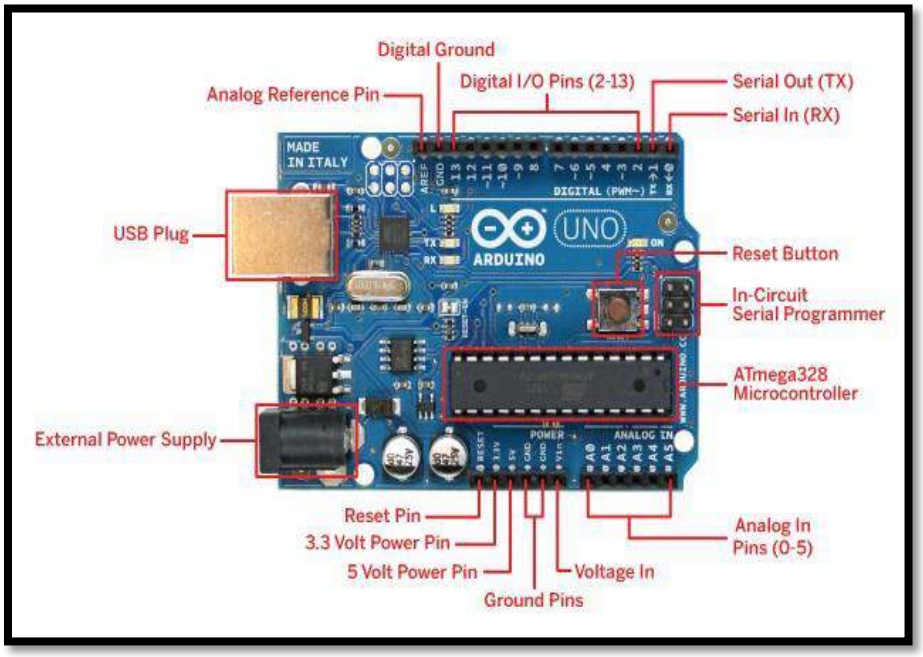


Figura 3. Partes de una Tarjeta Arduino UNO [16].

Entornos de desarrollo para Arduino

Entorno de Desarrollo Integrado (IDE por sus siglas en inglés Integrated Development Environment), es un software que agrupa ciertas herramientas que ayudan y permiten a desarrolladores a escribir, editar, simular y detectar errores en un programa ver Figura 4 [12].

Al describir un programa (Sketch) se dice que esta integrado por una serie de instrucciones ordenadas en grupos adecuadamente sin tener algún tipo de confusión. Un Sketch es usado para ser grabado en la memoria flash del microcontrolador de Arduino gracias al uso de un IDE.

Figura 4. IDE oficial de Arduino.

Fuente: Imagen de Elaboración propia

Si se desea iniciar a trabajar con Arduino, crear o probar algún Sketch se necesitará del IDE de desarrollo oficial, que puede ser descargado sin ningún costo en la página de Internet https://www.arduino.cc/ , por otra parte, no es la única opción a utilizar, existen otras plataformas. Por diversas razones, muchos usuarios no quedan satisfechos con la versión oficial, a continuación se enlistan algunas opciones [12]:

CodeBloks: plataforma libre para desarrollar aplicaciones en lenguaje C/C++, pero también se pueden crear y grabar Sketchs orientados a Arduino, para más información ingrese a http://www.arduinodev.com/codeblocks.

Gnoduino: es un IDE de uso único para Linux, este es una réplica del original de Arduino, pero desarrollado en Python lo cual lo hace una versión independiente de Java, cuyas características destacan el ser ligero, rápido y de uso en el escritorio GNOME, puede ser descargado de la página http://gnome.eu.org .

Codebender: este IDE es una página de Internet y a la vez cuenta con una versión instalable, el programador decide cual usara eso de pende de él, la forma online contiene la ventaja de dar espacio de almacenamiento en la nube y registro gratis, ya que esta plataforma integra un editor y un compilador para poder grabar el sketch en el microcontrolador de Arduino, si se desea más información puede ser consultada en http://www.codebender.cc .

Visualmicro: es un (plugin) usado en la plataforma de desarrollo de Visual Studio para desarrollar código Arduino, es una herramienta de uso privado, si se desea consultar más información visite el siguiente enlace http://visualmicro.codeplex.com .

Atom: editor de texto moderno, accesible y multiplataforma, aprovechado para programar en diversos lenguajes, además es simple configurar el uso de un plugin llamado **platomformio** entorno libre que permite desarrollar en Arduino. Si se desea consultar más información entre al siguiente enlace https://atom.io/packages/platomformio .

SublimeText: es un IDE privado que permite desarrollar en distintos lenguajes y además de ser multiplataforma, integra un plugin llamado **Stino** para programar en lenguaje Arduino. Si se desea más información consulte https://www.sublimetext.com/3dev.

Estructura general de un programa Arduino

En el lenguaje de programación de Arduino, la estructura básica de un “sketch” es muy sencilla (ver Figura 5), se conforma de tres partes principales, el primer apartado lo conforman las variables globales, y las otras dos son funciones esenciales que están declaradas por bloques que agrupan las instrucciones del programa [12][17].

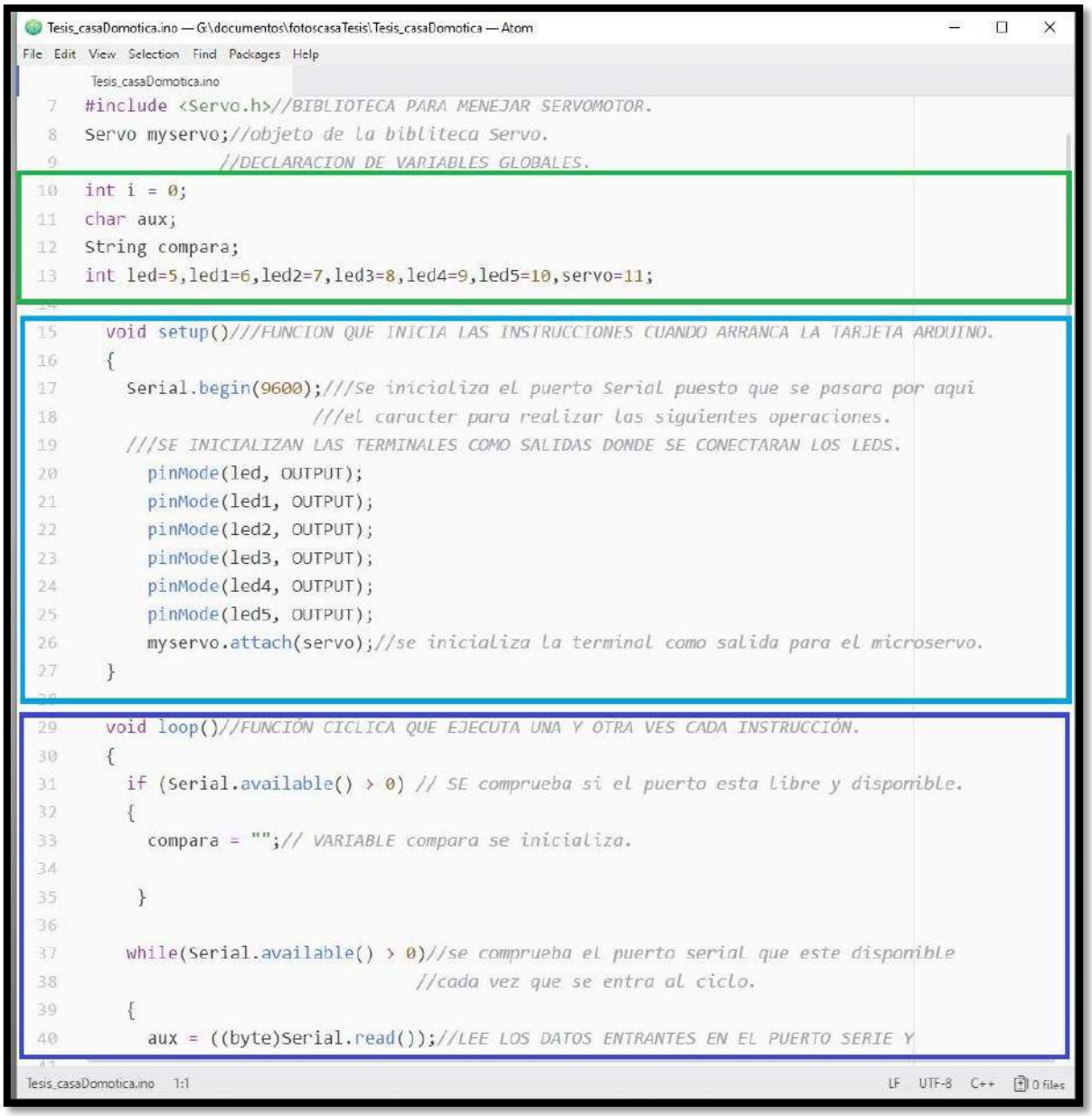


Figura 5. Estructura básica de un programa Arduino.

Fuente: imagen de elaboración propia.

Declaración de variables globales: se declara al principio de programa como la configuración de terminales de entrada y salida, constantes con un valor predeterminado o lo que se desee declarar ya que no tiene una delimitación, a diferencia de las funciones setup y loop. La Figura 6 muestra un ejemplo de la declaración de variables globales.

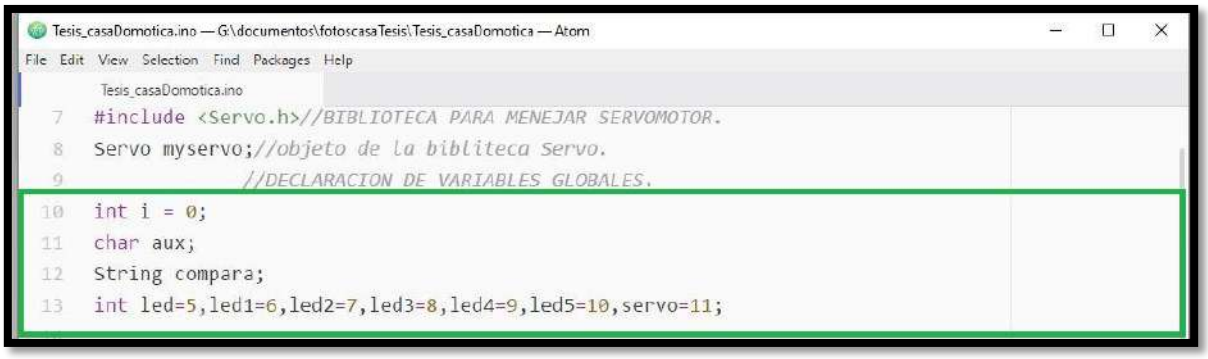


Figura 6. Declaración de variables globales.

Fuente: imagen de elaboración propia.

void setup (): es la parte encargada de recoger la configuración, se delimita por inicio y cierre de llaves. En este apartado las instrucciones solo son ejecutadas una vez al iniciar la tarjeta Arduino como se muestra en la Figura 7.

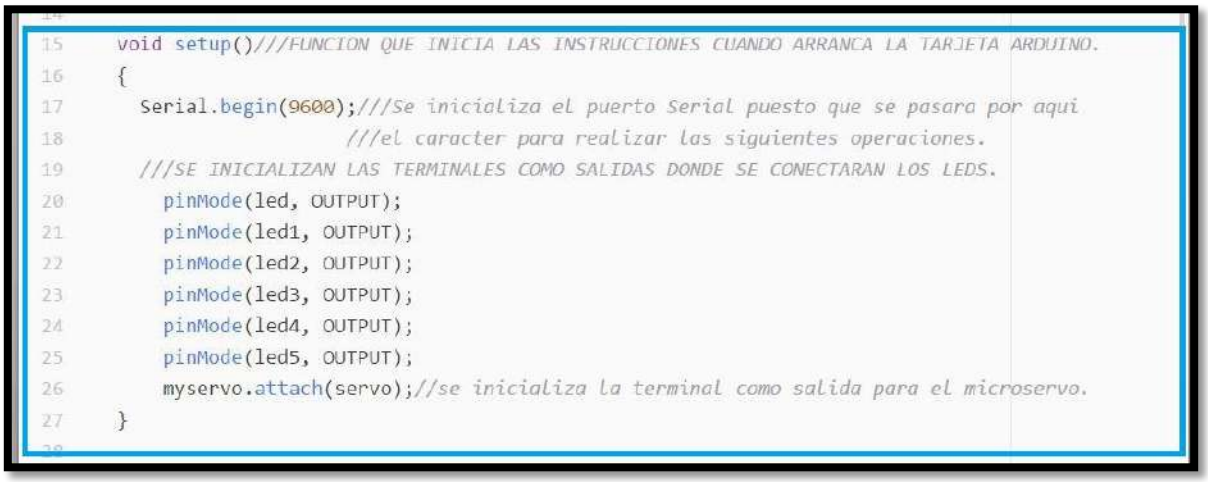
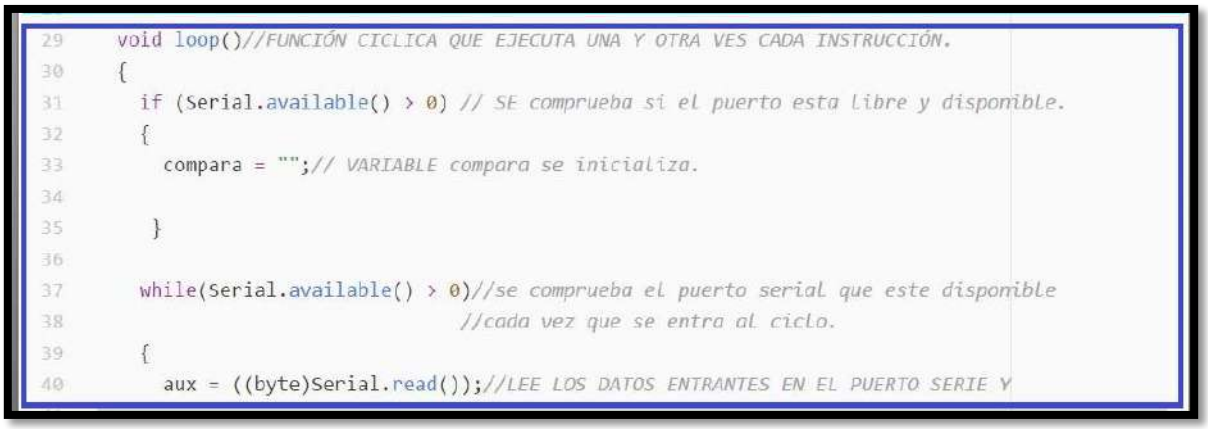


Figura 7. Función que inicializa las instrucciones al iniciar la tarjeta Arduino.

Fuente: imagen de elaboración propia.

void loop (): es la función del programa que se ejecutará cíclicamente (de ahí el termino loop –bucle), al igual que el setup es limitado por llaves de inicio y cierre. En esta parte del Sketch, las instrucciones del bloque son repetidas una y otra vez hasta que se ponga un punto de paro o una instrucción que detenga el ciclo en un lapso de tiempo. La Figura 8, presenta un ejemplo de esta función.



```
29  void loop()//FUNCIÓN CICLICA QUE EJECUTA UNA Y OTRA VES CADA INSTRUCCIÓN.
30  {
31    if (Serial.available() > 0) // SE comprueba si el puerto esta libre y disponible.
32    {
33      compara = "";// VARIABLE compara se inicializa.
34
35     }
36
37    while(Serial.available() > 0)//se comprueba el puerto serial que este disponible
38                                 //cada vez que se entra al ciclo.
39    {
40      aux = ((byte)Serial.read());//LEE LOS DATOS ENTRANTES EN EL PUERTO SERIE Y
```

Figura 8. Función que ejecuta las instrucciones cíclicamente.

Fuente: imagen de elaboración propia.

Bibliotecas para Arduino

El uso de bibliotecas en el entorno de programación Arduino es común al igual que otras plataformas de desarrollo [18]. Se emplean con objetivo de proporcionar ayuda extra en los sketchs, manejo de hardware y datos, si se desea integrar una biblioteca a un programa se realiza como se muestra en la Figura 9.

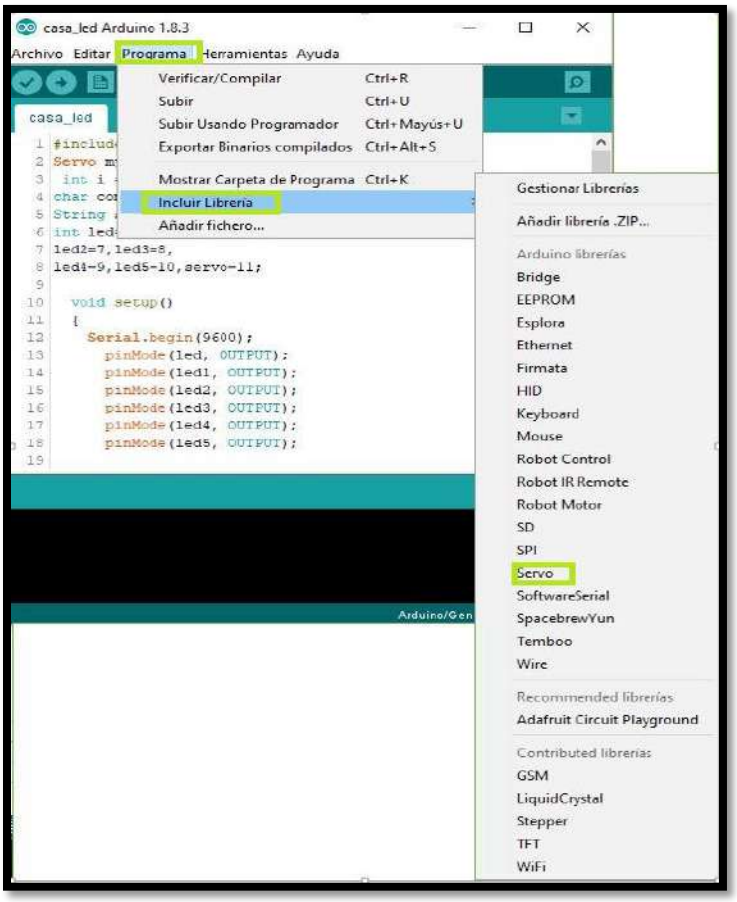


Figura 9. Ejemplo de inclusión de biblioteca para trabajar con servo.

Fuente: Imagen de elaboración propia.

La mayoría de bibliotecas con las que trabaja Arduino ya vienen integradas al IDE, pero también se pueden descargar o crear nuevas según la Guía de estilo API (**API Style Guide**), si se desea consultar más información ingrese a https://www.arduino.cc/en/Reference/APIStyleGuide .

Bibliotecas estándar para Arduino

Las bibliotecas más comunes para Arduino son las que se muestran en la Tabla 2.

Tabla 2. Algunas de las bibliotecas más comunes para Arduino.

Nombre	**Función**
EEPROM	Lectura y escritura
Ethernet/ ethernet2	Conexión a internet

Firmata	Comunicación de aplicaciones con protocolo serie estándar.
GSM	Conexión a la rede GSM/GRPS
LiquidCrystal	Control de pantallas LCD.
SD	Lectura y escritura de tarjetas SD.
Servo	Control de servos motores.
SPI	Comunicación de dispositivos que usan el bus Interfaz Periférica Serial (SPI).
SoftwareSerial	Comunicación en cualquier terminal digital.
Stepper	Control de motores a pasos.
TFT	Para dibujar texto, imágenes entre otros.
WiFi	Uso del escudo Arduino para conectarse a Internet.
Wire	Enviar y recibir datos a través de una red de dispositivos y sensores.

Servo library

En el desarrollo de este trabajo, se utilizaron servos para la implementación del sistema domótico. Es valioso destacar que la biblioteca de control de servomotores en la tarjeta Arduino "servo library", es la encargada de manejar con gran precisión los engranes y eje que integran a los servos comunes, permitiendo manipular el eje en diferentes ángulos entre 0 y 180 grados, ya que a diferencia de los motores de rotación continua puede ser ajustada la velocidad de giro del eje como se desee [19].

En cuanto a la biblioteca Servo library, reconoce 48 servomotores en la versión Mega de Arduino, y para el resto de las tarjetas Arduino sólo harán el reconocimiento de 12 motores, desactivando automáticamente el analogWrite () de Modulación de Ancho de Pulsos (PWM) en las entradas 9 y 10, mientras tanto en unidades de control Mega el manejo de esta biblioteca anula la función (PWM) en las terminales 11 y 12 si se emplean de 12 a 23 servos.

Shields de Arduino

Una shield (“Escudo” en español) es un circuito modular impreso en una placa para ser apilada sobre la parte superior de Arduino o de otra shield, lo cual ayuda a expandir la funcionalidad de la unidad de control. Logrando la comunicación entre ellas mediante conexión USB o simplemente interactuando con sus terminales de entrada y salida [12] [20][21]. Se debe tomar en cuenta que al montar una shield sobre una placa Arduino debe de tener la misma forma para poder integrarse a ella.

Existen dos grupos de shields, las oficiales y las no oficiales. Algunas de ellas se describen a continuación [12]:

1. **Shields Oficiales:** son placas armadas y distribuidas especialmente por Arduino, brevemente se enlistan algunas;
    - **Arduino Ethernet Shield:** es una tarjeta que se apila a la placa Arduino UNO con la característica de ser conectado a una red cableada TCP/IP, además presenta similitud de contener el mismo microcontrolador W5100 de Arduino Ethernet.
    - **Arduino Wireless SD Shield:** es compatible con Arduino UNO permitiendo la comunicación inalámbrica mediante dispositivos XBee acoplándose con otros mecanismos XBee en interiores a 100 m y en forma exterior hasta 300 m, además integra una ranura para colocar y usar una Micro SD que se configura a través de la biblioteca estándar “SD”.
    - **Arduino Wi-Fi Shield:** este escudo aumenta la capacidad de la placa Arduino UNO si se desea conectar a redes TCP/IP, puesto que incorpora un microcontrolador HDG104 y contiene una antena que concede conectarse a redes Wi-Fi versión 802.11b y 802.11g.
    - **Arduino Motor Shield:** es una shield que permite el control de servomotores, motores a pasos, relevadores entre otros dispositivos, en Arduino UNO. Su desempeño de este escudo es gracias a que integra un microcontrolador L298P ayudando a controlar la velocidad y dirección de mecanismo.
2. **Shield no Oficiales (prototipos):** son placas preensambladas y construidas por gran parte de individuos aficionados a Arduino, de otro modo estos escudos pueden ser llamados prototipos.

Historia de los servomotores

El 13 de Junio de 1831 nace en la ciudad de Edimburgo Escocia el cinetífico James Clerk Maxwell, quien aporta a la humanidad la teoría de electromagnética clásica y con sus 4 ecuaciones de Maxwell logra simplificar las leyes de la electricidad, el magnetismo y la óptica.

Las escuaciones del científico Maxwell permiten deducir una ley de servomotores que determinan la relación entre el campo electromagnético y el torque aplicado.

En 1821 se crea el primer motor por el científico e inventor Michael Faraday de origen británico, quien realizo un experimento que colocaba un magneto sobre un plato cóncavo con mercurio y conectado a un polo del magneto y del plato con una batería, dando como resultado un campo magnético que produjo movimiento circular en este sencillo motor de un polo, para entonces ya en el año 1832 el científico e inventor William Sturgeon inventa un motor de cuatro polos para un asador, para dar paso a la cración de la primera cocina moderna.

El físico matemático Holandes Hendrik Antoon Lorentz establece que cuando se coloca un conductor que transporta la corriente en un campo magnético, se genera una fuerza ortogonal (perpendicular) al flujo de la corriente [22].

Servomotores

Resulta oportuno definir que un servomotor es un poderoso dispositivo con ciertas funciones especiales, es decir, puede ser controlado el movimiento de su eje en un rango de posición entre 0° a 180° y a la vez su velocidad de giro en un lapso de tiempo y mantenerlo fijo en un determinado ángulo. Si son modificados, estos motores sólo hay que remplazar el mecanismo de engranaje, transformando la rotación de 0° a 360°[23][24].

Si pensamos porqué un servomotor es fácil de controlar su posición y rapidez (ver Figura 10), es a causa de que está compuesto en su interior por un motor DC (corriente directa) que incluye un reductor de velocidad, caja de engranajes y un circuito de control de corriente.

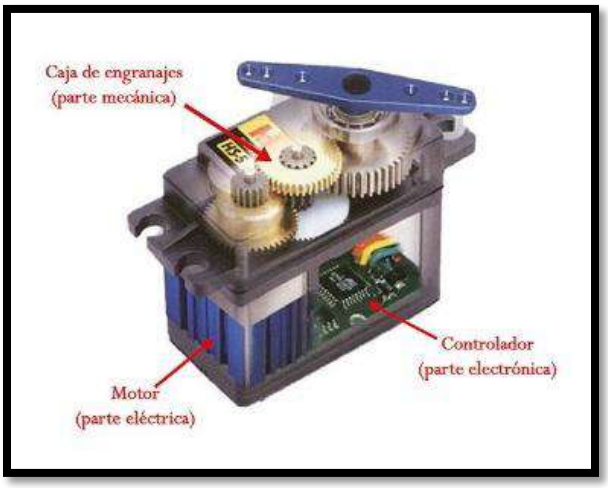


Figura 10. Componentes de un servomotor SG90 9g [25].

El manejo de un servomotor es realizado por tres cables; el primero realiza la función de alimentar con corriente eléctrica, en otras palabras, el positivo (Vcc 4.8 a 6 volts), el segundo es el negativo (tierra o GND) y el tercero es la señal de control de pulsos.
La Figura 11 muestra la configuración de los cables de algunos distribuidores [24].

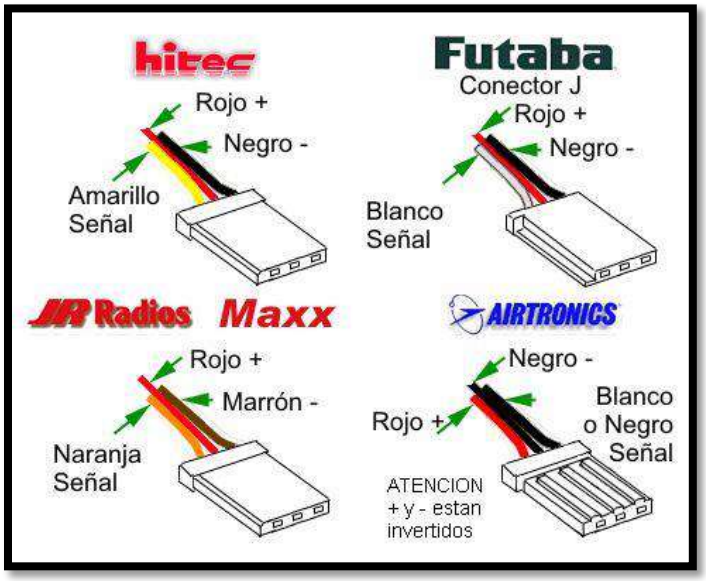


Figura 11. Configuración de cables conforme al fabricante [26].

Características generales de Servomotores

La Tabla 3 en lista la configuración y características de servomotores de acuerdo a los fabricantes que distribuyen este tipo de dispositivos [27]**.**

Tabla 3. Características generales de Servomotores de acuerdo a fabricantes.

Fabricante	Duración de pulsos (ms)			Frecuencia (Hz)	Color de cables		
	Mínima (0°)	Neutral (90°)	Máxima (180°)		Positivo	Negativo	Control
Futaba	0.9	1.5	2.1	50	Rojo	Negro	Blanco
Hitech	0.9	1.5	2.1	50	Rojo	Negro	Amarillo
Graupner/Jr.	0.8	1.5	2.2	50	Rojo	Negro	Naranja
Multiplex	1.05	1.6	2.15	40	Rojo	Negro	Amarillo
Robbe	0.65	1.3	1.95	50	Rojo	Negro	Blanco
Simprop	1.2	1.7	2.2	50	Rojo	Negro	Negro

Servomotor SG90 9 g Micro Servo

Este dispositivo pertenece a la familia de servomotores, como su nombre lo dice SG90 9 g Micro Servo (ver Figura 12), es pequeño y ligero, pero con la capacidad de generar torque de salida suficiente para muchas aplicaciones. Su rotación está en el rango de 0° a 180° (90° en cada dirección) su desempeño es parecido a los motores de tamaño estándar, pero a menor escala. Se puede emplear una variedad de servo código, hardware o distintas bibliotecas para echar andar un servomotor [23].

Figura 12. Micro Servo SG90 9g [23].

La Tabla 4 destaca características que integran al Micro Servo SG90 9g con respecto al datasheet del fabricante para que el usuario pueda manipularlo sin problemas.

Tabla 4. Características técnicas del Micro Servo SG90 9g.

Características	
• **Peso**	9 g
• **Dimensiones**	22.2 x 11.8 x 31 mm
• **Stall torque**	1.8 kgf *cm
• **Velocidad de Funcionamiento**	0.1 s/ 60 grados
• **Voltaje de funcionamiento**	4.8 v (~5 v)
• **Rango de temperatura**	0 °C - 55 °C
• **Ancho de banda muerta**	10 μs

Historia del Bluetooth

El termino Bluetooth se remonta al siglo diez a causa del nombre del rey danés Harald Blátand, quien fue considerado un pacifista de la guerra de Escandinavia, por otra parte el nombre de este emperador "Blátand" se puede traducir al inglés como Bluetooth [28].

El nacimiento de la tecnología inalámbrica Bluetooth se inicia aproximadamente en la década de los 90 cuando la empresa Ericsson buscaba desarrollar un protocolo de comunicación que le permitiera realizar una conexión a corto alcance con otros dispositivos, con la ventaja de que esta tecnología no consumiera bastante energía en dispositivos móviles.

Al paso de los años y el avance de la tecnología, bastantes empresas importantes empezaron a mostrar interés en la comunicación Bluetooth, para ese entonces se crea una asociación llamada SIG (Special Interest Group), conformada por Ericsson, Apple, Intel, Nokia, entre muchas otras.

Al paso de los años y después de varias versiones beta, en el año 1999 Bluetooth lanza su primera versión 1.0 quien permitía una velocidad de transferencia de 0.8 a 1Mbps con un alcance no mayor a los 10 m [29].

Módulo Bluetooth HC-05

Uno de los componentes principales del sistema domótico desarrollado es un módulo para la comunicación por medio de Bluetooth.

Un dispositivo Bluetooth HC-05 (ver Figura 13), es un módulo de comunicación serial inalámbrica, tiene la característica de tener una tasa de transferencia mejorada de 3 Mbps, modulación con transceptor de radio completo de 2.4 GHz y banda de base, además de ser configurado Master (maestro) – Slave (esclavo). Quiere decir que es capaz de vincularse con terminales móviles o computadoras, y a su vez genera conexiones a otros artefactos electrónicos, a diferencia de su sucesor HC-06, que sólo puede trabajar como Slave (esclavo). Después de lo anterior expuesto, la unidad HC-05 resalta su popularidad al ser usados en aplicaciones relacionadas con microcontroladores y microprocesadores [30][31][32].

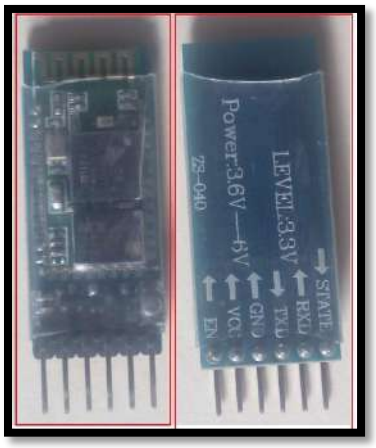


Figura 13. Módulo Bluetooth HC-05.

Fuente: imagen de elaboración propia.

Características del módulo Bluetooth HC-05

Para configurar en una tarjeta Arduino el módulo de comunicación Bluetooth HC-05, los usuarios deben de saber ciertas características, por ejemplo, a que voltaje trabaja y cuáles son sus terminales, entre otras. Para saber un poco más de este dispositivo vea la Tabla 5 [31].

Tabla 5. Características técnicas del módulo Bluetooth HC-05.

Características
• Bluetooth versión 2.0 + EDR (Enhanced Data Rate).
• Configuración maestra – esclavo, y esclavo con autoconexión (Loopback).
• Chip de radio: CSR BC417143.
• Frecuencia: 2.4 GHz, banda ISM.
• Modulación: GFSK (Gaussian Frequency Shift Keying).
• Antena PCB incorporada.
• Potencia de emisión: ≤ 4 dBm, Clase 2.
• Rango de alcance de 5m a 10m.
• Sensibilidad: ≤ 84 dBm a 0.1% BER.
• Velocidad Asincrónica: 2.1 Mbps(Max)/160 kbps, sincrónica: i Mbps/1 Mbps.
• Seguridad: autenticación y encriptación (Password por defecto: “1234” o “0000”).
• Perfiles: puerto serie Bluetooth.
• Modulo montado en una tarjeta con regulador de voltaje y 6 terminales suministran el acceso de VCC, GND, TXD, RXD y status LED (STATE). • VCC: alimentación de 3.6v mínimo a un máximo de 6v. • GND: negativo del módulo. • TXD: terminal de trasmisión de datos. • RXD: terminal de recepción de datos, trabaja con un voltaje de 3.3v, pero ya incluye un conversor de potencia y puede ser conectado directamente a los 5 v de la placa Arduino. • EN: terminal para entrar a la configuración del módulo HC-05 • STATE: terminal de salida para conectar un led y visualizar cuando se está comunicando el módulo HC-05.
• Consumo de corriente: 50 mA.
• El terminal RX del módulo requiere resistencia de pull up a 3.3v (4.7k a 10k). Si el microcontrolador no tiene resistencia pull up en el terminal TX se debe colocar externamente.

• Niveles lógicos: 3.3 v conectar a señales a más de 5v puede ocasionar daños en modulo.
• Voltaje de alimentación: 3.6v a 6v.
• Dimensiones totales del módulo: 1.7 cm x 4 cm.
• Rango de Temperatura de operación: 20 ºC a 75 ºC.

Módulos Bluetooth HC-05 y HC-06

El módulo Bluetooth HC-05 cuenta con 6 terminales para ser manipulado, pero su sucesor el HC-06, solo cuenta únicamente con 4 terminales como se puede ver en la Figura 14. La Tabla 6 muestra más ampliamente las terminales que integran estos módulos [32][33].

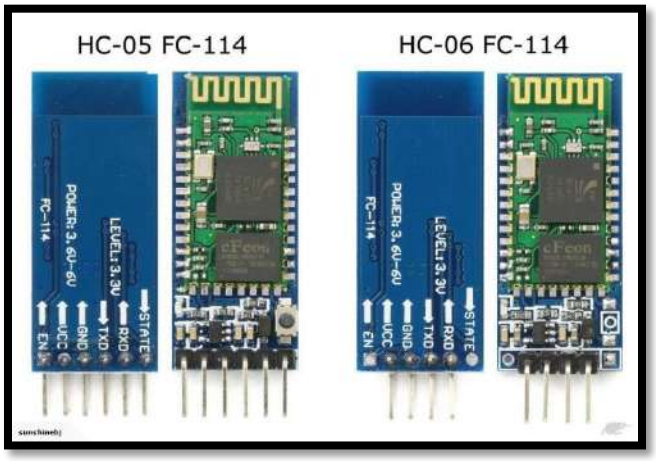


Figura 14. Bluetooth HC-05 y Bluetooth HC-06 [34].

Tabla 6. Terminales que integra el Bluetooth HC-05 Y HC-06.

MÓDULO HC-05	MÓDULO HC-06	Descripción
GND	GND	Tierra
VCC	VCC	Alimentación 3.3-6V
TX	TX	Transmisión.
RX	RX	Recepción
EN	-	Habilitar
STATE	-	Salida para conectar un led.

Android

El sistema operativo Android fue creado con la misma intención que los diversos sistemas operativos móviles que existen en la actualidad en el mercado de telefonía, es decir, facilitar al usuario la utilización de los dispositivos, así como permitir la ejecución de aplicaciones en múltiples y diferentes marcas de teléfonos inteligentes.

Hoy en día Android es un sistema operativo utilizado en distintos tipos de dispositivos, no únicamente para telefonía móvil, también ya es dirigida a cualquier objeto electrónico, por ejemplo: tablets, computadoras, TVs, electrodomésticos de línea blanca, automatización de casas, oficinas, entre muchos otros [35][36].

Android es un sistema operativo de código abierto, cuyo núcleo es basado en Linux 2.6, convirtiéndolo en una plataforma realmente libre, dando oportunidad a cualquier individuo que quiera desarrollar para móviles u otros terminales.

La facilidad de crear y diseñar aplicaciones en Android para múltiples dispositivos es gracias a que usa una Máquina Virtual, parecida a la máquina virtual de Java JVM (por sus siglas en inglés, Java Virtual Machine). La máquina virtual de Android se llama Dalvik, fue desarrollada para versiones menores a la 5.0, tiempo después se lanzó la máquina virtual Android Runtime (ART), mejorada y orientada a versiones superiores a la 5.0.

Historia de Android

En el año 2003, la empresa Android desarrolla su sistema operativo para dispositivos móviles [37][38]. La compañía Android Inc. fue adquirida por Google en julio de 2005, posteriormente se inicia los trabajos en la creación de una máquina virtual parecida a la de Java, se crea la Dalvik Virtual Machine, optimizada y adaptable para dispositivos móviles.
En 2007, después de empezar a trabajar sobre la maquina Dalvik, se funda la organización OHA (Open Handset Alliance), conformada por importantes empresas de alto prestigio en el mercado de dispositivos móviles, encabezadas por Google, HTC, LG, Motorola, Samsung y Sony Ericsson, entre otras. El motivo de formar una unión empresarial, fue para impulsar el desarrollo de estándares libres que hicieran crecer la telefonía móvil [37][38].

Por otra parte, en el mes de noviembre de 2007 es lanzada una versión actualizada de Android SDK. Para entonces, en el año 2008, la compañía Google saca al mercado el primer equipo móvil con SO Android: el modelo “T-Mobile G1”. Después, en el mes de octubre, la empresa Google decide liberar el código fuente de su Sistema Operativo bajo el cargo de

Open Source Apache, como consecuencia surge la tienda de apps “Google Market”, facilitando a los usuarios la descarga de aplicaciones a sus terminales. Tras el paso de los meses, en abril de 2009, la versión 1.5 se presenta con mejoras en el teclado y pantalla. Posteriormente, en el año 2010 sobresalen las versiones 2.1, 2.2 y 2.3, con similitudes y nuevas características, colocando a Android entre uno de los sistemas operativos más empleados en equipos móviles, dejando atrás por mucho a iOS de Apple.

Después de haber tenido éxito en 2010 con las versiones 2.x, en el año 2011 se lanzan dos actualizaciones destacables; la primera es la 3.x dirigida principalmente a tabletas electrónicas, y la segunda es la 4.x, enfocada a todos los equipos móviles. Posteriormente, en el año 2012, Google decide actualizar la tienda de apps “Google Market” a “Google Play Store”, presentando mejoras. Contando con una variedad de contenidos. No únicamente aplicaciones, ahora incluye música, videos, imágenes, libros, entre otras funciones. Android está posicionado como uno de los gigantes de la nueva era de la telefonía.

Arquitectura de Android Dalvik

El sistema operativo Android está conformado por cuatro capas esenciales que forman su funcionamiento (ver Figura 15), desarrolladas bajo la licencia de código abierto [37] .

Kernel o núcleo de Linux

El núcleo de Android está basado en el sistema operativo Linux, adaptado totalmente al hardware de dispositivos móviles, proporciona distintos servicios, en particular el manejo de la seguridad, la gestión de memoria, además de determinados protocolos al igual que el multiproceso y el funcionamiento de los controladores para periféricos (*drivers*).

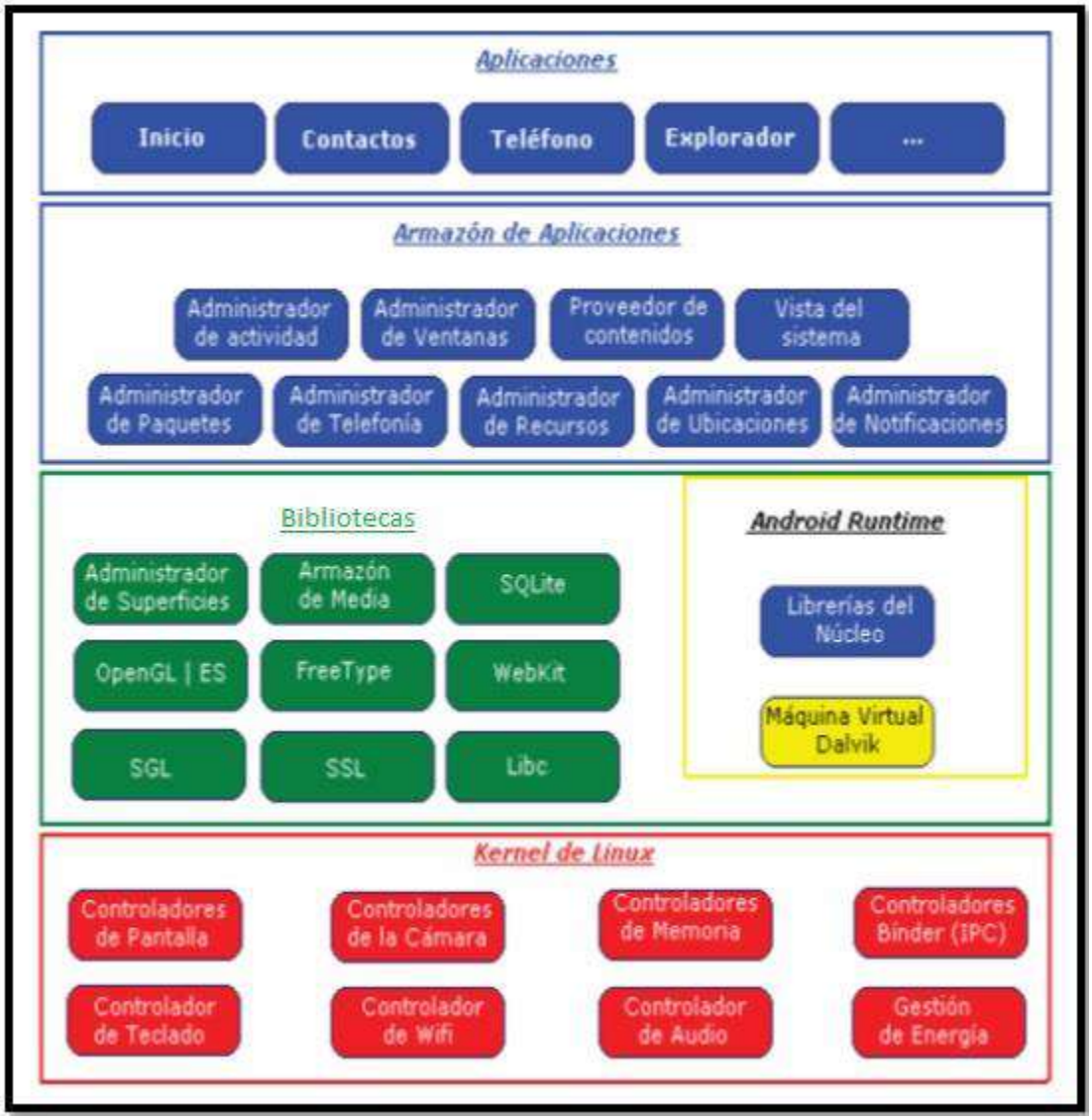


Figura 15. Arquitectura Dalvik [39].

Android Runtime o entorno de ejecución de Android

El entorno de ejecución es basado en la Máquina Virtual de Android (Dalvik) desarrollada por Google para dispositivos móviles, la cual se adapta a las limitaciones de memoria y procesador. En cuanto este fragmento de la arquitectura no es considerado una capa ya que está incorporada a Android Runtime. En otras palabras, el entorno de ejecución es el conjunto de herramientas conformadas por un grupo de bibliotecas basadas en el lenguaje de programación Java, permitiendo a programadores desarrollar aplicaciones Android.

Bibliotecas nativas

La mayoría de bibliotecas nativas que integran este sector de la arquitectura son desarrolladas en lenguaje de programación C/C++, empleadas en diversos componentes que conforman Android, están compiladas en código a nivel de procesador bajo la licencia de Open Source, cabe mencionar las siguientes [37][40]:

- **System C library:** su origen tiene relación con la Biblioteca estándar (libc) del lenguaje de programación C, la cual aporta un conjunto de funciones como el manejo de cadenas de caracteres, cálculos matemáticos, administra el uso correcto de memoria entre muchos otros servicios del sistema operativo.
- **Media Framework:** biblioteca basada en *Packet Video´s Open Core*, desarrollada para la grabación y reproducción de distintos tipos de formatos de imagen, audio y video.
- **Surface Manager:** responsable de cuidar el acceso de los gráficos 2D y 3D en la pantalla.
- **WebKit:** acepta navegador web moderno de Android basado en webView (navegador interno de aplicaciones de acceso a desarrolladores).
- **SGL:** se encarga de los gráficos 2D.
- **LIBRARYS 3D:** biblioteca desarrollada en la API OpenGL ES 1.0, hace uso de acelerador de hardware 3D y software altamente optimizado por proyección 3D.
- **FreeType:** transformación de estilo de imágenes.
- **SQLite:** base de datos relacionales SQLite, rápidas y ligeras.
- **SSL:** integra servicios de comunicación encriptada entre cliente y servidor a través de Secure Socket Layer.

Entorno de aplicación o marco de trabajo de aplicaciones

El entorno de trabajo de aplicaciones fue diseñado para establecer uso simplificado de componentes, puesto que una aplicación puede compartir ciertas capacidades que después otra hará uso de ellas, teniendo en cuenta el cumplimiento correcto de las normas de seguridad.

Gran parte de los elementos que conforman esta capa son bibliotecas del lenguaje Java tienen acceso a ciertos recursos derivados de la ejecución de Dalvik Virtual Machine. Algunas de las más importantes se mencionan a continuación:

- **Activity Manager (Administrador de Actividades):** tiene el control en el ciclo de vida de las actividades
- **Views (vistas):** son empleadas en el diseño de vistas para pantallas de dispositivos Android.
- **Resource Manager (Administrador de recursos):** concede permisos de acceso a gran parte de los elementos que conforman una aplicación que no está directamente dentro del código, como contenido de audio, imágenes, cadenas de caracteres traducidas a otros lenguajes, entre otros.
- **Windows Manager (Administrador de ventanas):** realiza la tarea de organizar el contenido que aparece en pantalla y crea superficies que son rellenadas por otras actividades.
- **Content Provider (Proveedor de Contenidos):** junta un grupo de datos que posteriormente comparte con las demás aplicaciones, un ejemplo de ello es la agenda telefónica que permite el acceso a los contactos alojados en el almacenamiento interno del teléfono.
- **Notification Manager (Administrador de Notificaciones):** administra el acceso de notificaciones personalizadas de las aplicaciones.
- **Sensor Manager (Administrador de Sensores):** administra la mayoría de los sensores que están integrados a los equipos Android, por ejemplo: el sensor de luz, presión, temperatura, distancia, proximidad, brújula, entre muchos otros.

Aplicaciones

El propósito de este nivel es la interacción de las aplicaciones y el usuario, por ello, es compuesta por cualquier aplicación que viene instalada en un teléfono con S.O (Sistema Operativo) Android o que el usuario puede instalar, cabe mencionar algunas de ellas; agenda de números telefónicos, navegadores web, cámara, calculadora, entre otras.

Arquitectura ART

La arquitectura ART (ver Figura 16) es la parte clave del funcionamiento del sistema Operativo Android en las últimas versiones. A diferencia a la arquitectura anterior ahora se incorpora HAL (capa de abstracción de hardware) haciendo ART superior a Dalvik [36][37] [41].

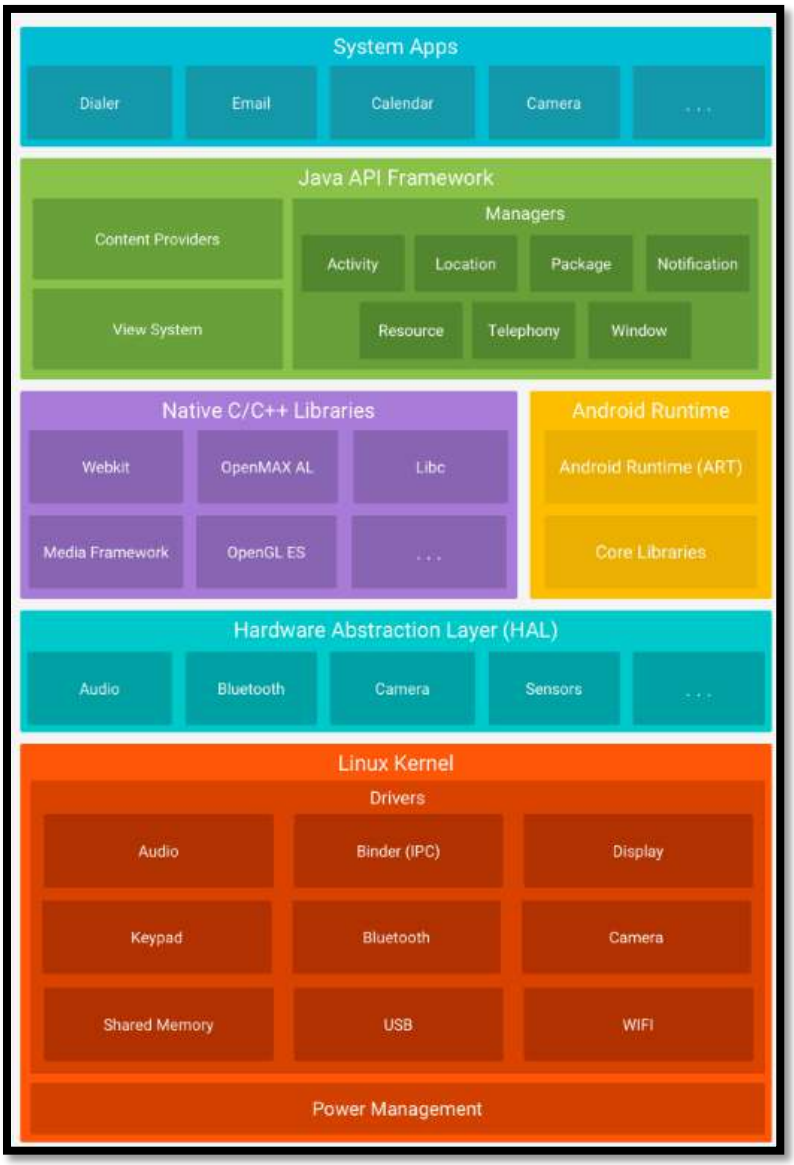


Figura 16. Arquitectura ART [42].

Kernel (núcleo) de Linux de ART

El núcleo de Linux es el cimiento principal del sistema operativo Android más reciente. ART utiliza este Kernel para controlar el uso de memoria de bajo nivel y la creación de subprocesos. Al emplear el núcleo de Linux permite a fabricantes de dispositivos móviles crear controladores y a la vez aprovechar servicios de seguridad.

Capa de abstracción de hardware (HAL, Hardware Abstraction Layer)

La función principal de esta capa es aportar interfaces estándares que le permiten al sistema operativo ejecutar el uso de los *drivers*, consiste en la implementación de una interfaz

para cada módulo del hardware, como el controlador de la cámara, Wi-Fi o del Bluetooth. En otras palabras, el sistema operativo llama la capa de los controladores del equipo móvil.

Entorno de ejecución Android ART

ART fue desarrollada para ejecutar varias máquinas virtuales, asimismo cada aplicación se ejecuta individualmente en su propio proceso instanciado al tiempo de ejecución de Android, puesto que realiza su compilación AOT (*Ahead-Of-Time*, antes de tiempo), explicado de otra forma, compila el archivo bytecode en código maquina previamente de ser ejecutado en el programa.

Cabe mencionar algunas funciones destacadas de ART:

- Mejoras en rendimiento del (GC) recolector de basura.
- Consumo reducido de energía.
- Compila archivos ". dex" en forma AOT (Ahead-Of-Time, Antes de Tiempo) y JIT (Just-In-Time, solo a tiempo).
- Optimiza depuración de aplicaciones.
- Aumento en el rendimiento del sistema operativo.
- Ahorro de tiempo en la ejecución de bytecodes.
- Manejo optimizado de memoria RAM y ROM.
- Compatibilidad de ejecución de aplicaciones entre ART O Dalvik.

Esta parte de Android es también conformada por un grupo de bibliotecas orientadas a destacar el trabajo del lenguaje java 8, integrando funciones que el framework de la API Java maneja.

Bibliotecas C/C++ nativas

La mayoría de los servicios y componentes que integran la última versión del sistema operativo Android son desarrolladas con base en las bibliotecas originarias del lenguaje de programación C/C++. Algunas de las bibliotecas nativas son las siguientes:

- **OpenGL ES**, se tiene acceso a través de la API de java OpenGL lo cual otorga compatibilidad en las aplicaciones Android para manejar dibujos y gráficos en 2D y 3D.
- **Manejo de NDK** (Kit de desarrollo nativo), para desarrollar aplicaciones en lenguaje C/C++, teniendo acceso a bibliotecas nativas para realizar compilaciones a partir de código fuente.

Framework de Java de API

La mayoría de las tareas que desempeña el sistema operativo son disponibles por la API desarrollada en lenguaje de programación Java, la cual permite desarrollar Apps (aplicaciones) de forma simplificada, al reutilizar ciertos servicios centrales y modulares al igual que otros componentes del sistema, como los que se enlistan a continuación:

- **Sistema de vista:** es conformada por listas, cuadriculas de texto, botones, navegador Web.
- **Administrador de recursos (Resource Manager):** conceden permisos a recursos sin código como archivos de diseño, gráficos y strings traducidas a otros lenguajes.
- **Administrador de notificaciones (Notification Manager):** permite a las aplicaciones mostrar notificaciones personalizada en la barra de estado.
- **Administrador de actividad (Activity Manager):** maneja el ciclo de vida de las aplicaciones móviles.
- **Proveedores de contenido (Content Provider):** permite que las aplicaciones compartan datos entre ellas.

Apps (aplicaciones) del sistema

Esta capa de la arquitectura ART es conformada por una serie de aplicaciones del sistema como; navegador de Internet, mensajería SMS, calendario, correo electrónico o simplemente la agenda de contactos, funcionando como medio de interacción con los usuarios.

Características del sistema operativo Android

Algunas de las principales características de este sistema se muestran a continuación [37]:

- **Plataforma realmente abierta.** Brinda oportunidad a cualquier individuo de programar Aplicaciones móviles de forma gratuita ya que es una plataforma basada en el sistema operativo Linux, convirtiendo a Android un entorno de desarrollo libre.
- **Portabilidad asegurada.** Las aplicaciones Android son multiplataforma porque son desarrolladas en lenguaje de programación Java, lo que permite ejecutarlas en cualquier teléfono móvil viejo o moderno gracias a la compatibilidad de la máquina virtual instalada en su sistema operativo.

- **Arquitectura basada en componentes inspirados en Internet de dispositivo siempre conectado a Internet.**

  Esto es a causa de emplear XML en el diseño de Interfaces de usuario, lo cual permite que las aplicaciones se ejecuten en cualquier terminal con diferentes tipos de tamaño de pantalla, ya sea; un teléfono inteligente (*Smartphone*), una tableta electrónica (*Tablet*), una computadora o cualquier otro dispositivo.
- **Gran cantidad de servicios incorporados.** Enlistando alguno de ellos; localización basada en GPS, bases de datos SQLite, sintetizador de voz, entre muchos otros.
- **Seguridad.** Es una característica muy importante basada en Linux que limita los permisos de ejecución de cada aplicación.
- **Optimizado para baja potencia y poca memoria.** Esto se debe a la Máquina virtual de Android que es parecida a la de Java. La cual permite adaptarse a dispositivos con limitados recursos de procesador y memoria.
- **Alta calidad de gráficos y sonido**. Manejo de gráficos 2D y 3D desarrollados con la biblioteca OpenGL ES, de igual forma soporta distintos formatos de audio, imagen y video.

Las versiones de Android y niveles de API

Android desde que es parte de Google otorga nombres de postres a sus versiones de sistema operativo, un resumen de ellas se presenta la Tabla 7 [37][43][44]. Al momento de desarrollar una App, es necesario considerar el nivel de la API para poder acceder a ciertos controles disponibles en algunas versiones.

Tabla 7. Versiones de Android y nivel de API.

Nombre de sistema operativo	Versión de Plataforma Android	Nivel de API
Apple Pie	1.0	1
Banana Bread	1.1	2
Cupcake	1.5	3
Donut	1.6	4
Éclair, Éclair_0_1, Éclair_MR1	2.0, 2.0.1 y 2.1x	5, 6 y 7

Froyo	2.2.x	8
Gingerbread	2.3, 2.3.1 y 2.3.2	9
Gingerbread_MR1	2.3.3 y 2.3.4	10
Honeycomb	3.0.x	11
Honeycomb_MR1, Honeycomb_MR2	3.1.x, 3.2	12 y 13
Ice Cream Sandwich	4.0, 4.0.1 y 4.0.2	14
Ice Cream Sandwich	4.0.3 y 4.0.4	15
Jelly Bean, Jelly Bean_MR1, Jelly Bean_MR2	4.1, 4.1.1, 4.2, 4.2.2 y 4.3	16, 17 y 18
KitKat	4.4	19
KitKat_WATCH	4.4W	20
Lollipop	5.0	21
Lollipop_MR1	5.1	22
Marshmallow	6.0	23
Nougat	7.0 y 7.1	24 y 25
Oreo	8.0 y 8.1	26 y 27

Fuente: información retomada de https://developer.android.com/guide/topics/manifest/uses-sdk-element.html?hl=es-419 .

IDE Android Studio

De acuerdo con el sitio oficial Android Studio, Android Studio es un entorno de desarrollo Integrado (IDE) (ver Figura 17) que proporciona herramientas eficientes para crear Apps enfocadas a cualquier dispositivo Android. Está basada en IntelliJ IDEA y cuenta con un poderoso editor de código [45].

Android Studio cuenta con una variedad de funciones para optimizar la compilación y desarrollo de Apps. Algunas de las cuales se enlistan a continuación [45]:

- Sistema de compilación basado en Gradle flexible.
- Emulador rápido con varias funciones.
- Entorno extendido para desarrollar para toda clase de dispositivos Android.

- Función Instant Run para modificar las apps sin volver a compilar el Apk (Android Application Package).
- Integra funciones para importar código y tareas comunes de las apps conformadas por plantillas y GiHub para compilar.
- Una vasta cantidad de herramientas y framework de prueba.
- Uso de herramientas Lint para localizar detalles en rendimiento, compatibilidad entre versiones Android y usabilidad.
- Compatibilidad del lenguaje de programación C++ y el NDK (Native Development Kit).

Figura 17. IDE Oficial de Android [46].

Estructura de un proyecto

Los proyectos en Android Studio son integrados por módulos que contienen archivos de código fuente y archivos de recursos. Entre los que destacan [47]:

- Módulos de aplicaciones para Android
- Módulos que contienen las bibliotecas
- Módulos de Google App Engine

Android Studio muestra los módulos de forma organizada como se muestra en la Figura 18.

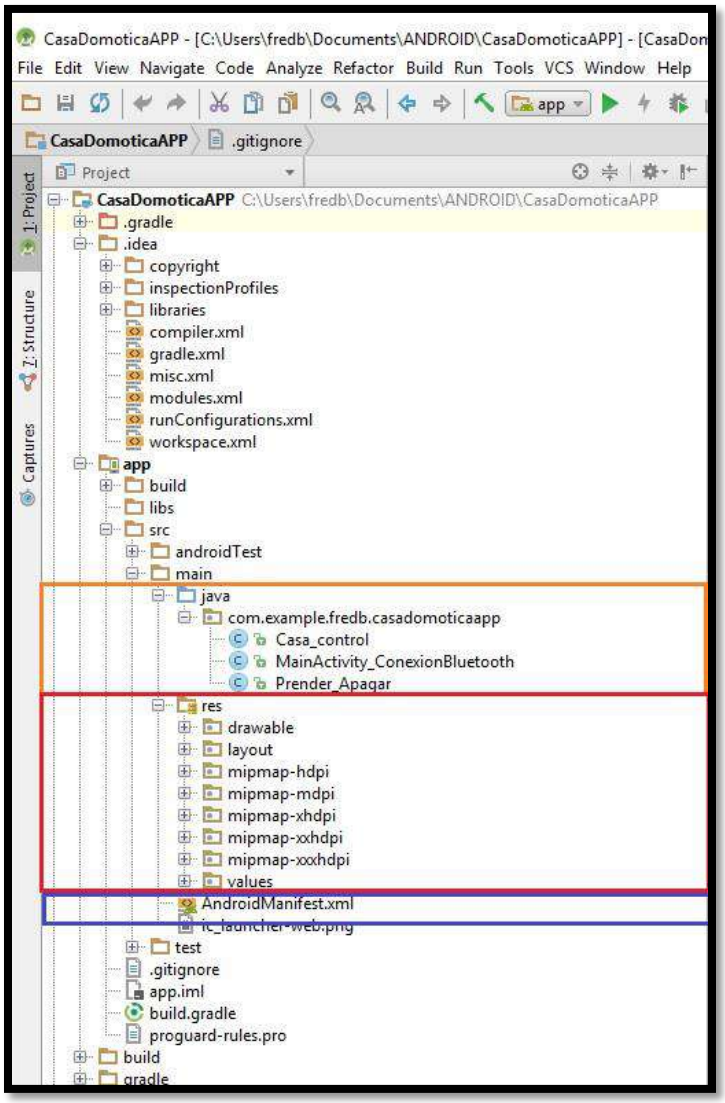


Figura 18. Estructura de un proyecto en Android Studio.

Fuente: imagen de elaboración propia.

Los directorios o carpetas más importantes son:

- **La carpeta manifest:** contiene los archivos de tipo AndroidManifest.xml, en otras palabras, este archivo contiene información general de nuestra App, como el nombre y permisos.
- **El módulo Java:** guarda archivos fuente desarrollados en Java incluyendo la prueba JUnit.
- **Res:** es el módulo que almacena repertorios de diseño XML, imágenes y cadenas de Interfaz de Usuario UI. Esta carpeta contiene todos los archivos indexados por el

compilador para generar archivos de recursos **R**, que posteriormente son usados de manera eficiente.

Versiones de plataformas para desarrollar en Android

Además de Android Studio, existen otras plataformas que permiten desarrollar Apps para Android, entre ellas se encuentran las siguientes [48]:

1. **Xamarin:** es una plataforma de desarrollo privada de Microsoft, orientada a iOS y Android. Puesto que sus méritos lo han llevado a posicionarse entre las favoritas de los programadores por su eficacia en el crear apps nativas centralizadas a sistemas operativos móviles, para más información consulte la página de Internet https://www.xamarin.com/.
2. **Adobe PhoneGap**: plataforma libre y distribuida por Open Source Apache Cordova, para desarrollar WebApps hibridas con HTML, CSS Y Java Script. Puede consultar más información en el sitio Web oficial https://phonegap.com/ .
3. **Appery.ip:** es un entorno de desarrollo dirigido a Apps Android, basado en la nube evitando ser instalado localmente en el disco duro, es una herramienta de uso privado creada por compañías de telefonía como Samsung y AT&T. Más información consulte el siguiente enlace https://appery.io/ .
4. **Appcelerator:** es una herramienta de desarrollo libre para Apps Android, puesto que es basada en Eclipse. Por ello Appcelerator es un entorno de tecnología Web compatible con JavaScript. Consulte más información en http://www.appcelerator.com/ .
5. **AppMachine:** un entorno de desarrollo de aplicaciones de uso privado, orientado a sistemas móviles como iOS, Android y Windows Phone. A demás de ser dirigido a programadores principiantes, el siguiente enlace de Internet es para más información http://www.appmachine.com/.
6. **MIT App Inventor:** es un entorno de desarrollo de aplicaciones para Android de uso libre desarrollado por Google. Si se desea consultar este sitio ingrese en la página de Internet http://appinventor.mit.edu/explore/ .

CAPÍTULO III DESARROLLO DEL SISTEMA DOMÓTICO

Las personas siempre han tenido necesidades que las casas sin algún tipo de tecnología integrada no pueden satisfacer, ejemplo de estas necesidades son confort, seguridad o incluso ahorro de energía. La innovación de las viviendas a través de la incorporación de sistemas electrónicos (domótica) logra que las casas habitación se conviertan en un lugar más placentero para vivir. Por lo anterior, en este trabajo se desarrolló un sistema domótico para una casa habitación integrada por 2 dormitorios, 1 baño, 1 sala, 1 comedor y 1 cochera para un automóvil. El sistema desarrollado tiene la finalidad de controlar las luces y puertas de una casa habitación, a través de una aplicación para teléfonos inteligentes (*Smartphone*) con sistema operativo Android. Este tipo de sistemas son apropiados para cualquier persona, pero brindan un gran apoyo a personas de la tercera edad y a personas con algún tipo de incapacidad motora.

Diagrama a bloques del sistema

El sistema domótico está compuesto por cuatro bloques principales (ver Figura 19):

- Aplicación: Se ejecuta en un Smartphone con sistema operativo Android, sirviendo como interfaz entre el sistema y el usuario.
- Módulo de Bluetooth: Permite la comunicación inalámbrica entre el Smartphone y la unidad de control.
- Unidad de control (Arduino UNO): Se encuentra integrada por un microcontrolador, quien se encarga de interpretar comandos recibidos y activar los circuitos para las luces y servomotores.
- Actuadores: Integrado por servomotor y leds que simulan las luces.

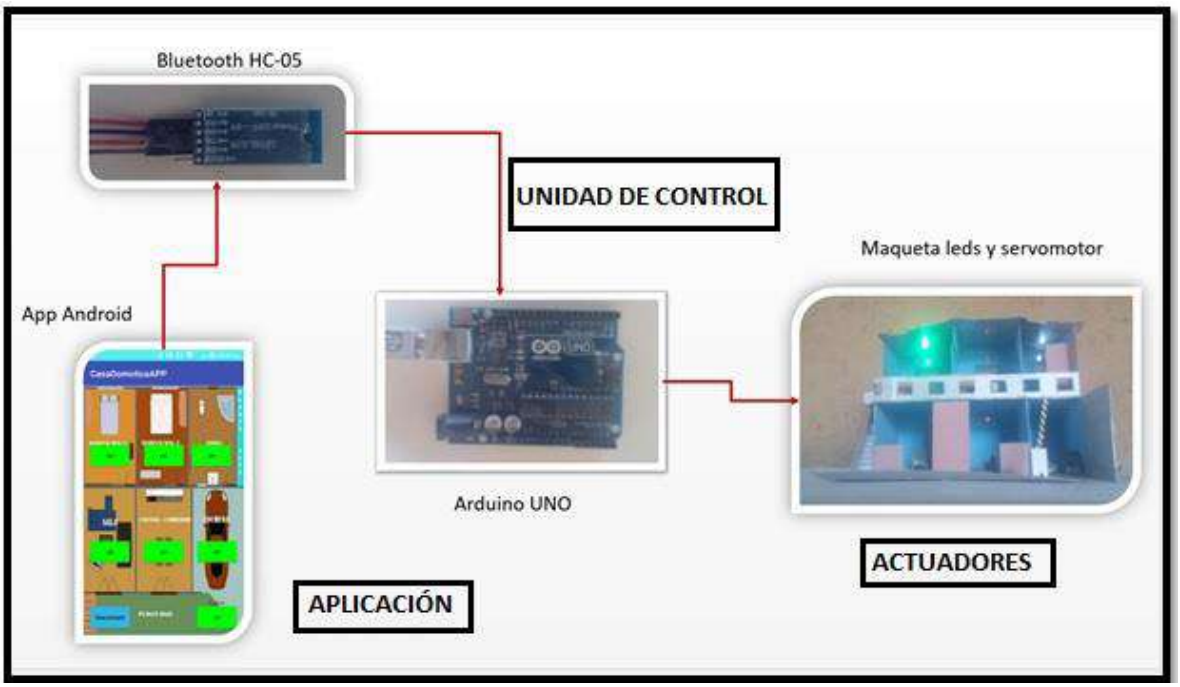


Figura 19. Diagrama de bloques del Sistema Domótico.

Fuente: imagen de elaboración propia.

Los componentes que integran el sistema domótico propuesto, se pueden ver en la Tabla 8, en la cual se lista el nombre y número de elementos utilizados.

Tabla 8. Componentes del sistema domótico.

Nombre del componente	**Cantidad**
Tarjeta Arduino Uno.	1
Módulo Bluetooth HC-05.	1
Leds.	6
Maqueta de casa habitación.	1
Micro Servo SG90 9g.	1
Resistencias de 220 ohms.	6
Resistencias de 330 ohms.	1
Teléfono móvil con sistema operativo Android.	1

Arquitectura del sistema

En la literatura especializada, puede encontrarse que existen 4 tipos de arquitecturas para implementar un sistema domótico [7], estas son las siguientes:

- **Arquitectura centralizada:** La arquitectura centralizada es la que tiene una unidad de control de forma central, en otras palabras, la unidad de control es el cerebro del sistema, puesto que no permite la interacción directa entre los sensores y los actuadores, ya que la unidad de control es la que recibe la información de los sensores para posteriormente gestionar y procesar la información captada para ser trasmitida a los actuadores, por lo tanto, si la unidad de control se dañara entonces todo el sistema dejaría de funcionar debido que esta permite la interacción de todos los elementos del sistema domótico.
- **Arquitectura descentralizada:** Este tipo de arquitectura cuenta con más de una unidad de control a diferencia de la anterior, además sus unidades de control están ligadas entre ellas a través de un bus que envía información que capta cada unidad de control para después actuar de forma individual como una unidad centralizada.
- **Arquitectura distribuida:** Esta arquitectura es más sencilla que las anteriores debido a que no usan una unidad de control, ya que la arquitectura distribuida hace interactuar directamente los sensores y los actuadores conectados mediante un bus central desempeñando un trabajo parecido a una unidad de control, en otras palabras, los sensores envían la información directamente a los actuadores y estos a su vez captan la información para realizar sus funciones.
- **Arquitectura hibrida o mixta:** Es una arquitectura donde se pueden combinar las tres arquitecturas antes mencionadas, para ser empleadas en un sistema domótico.

De estas arquitecturas, el sistema propuesto fue diseñado conforme a la arquitectura centralizada, misma que se muestra en la Figura 20. Está conformada por un microcontrolador central que recibe la información del módulo Bluetooth HC-05; un Smartphone que sirve de interfaz para el usuario; actuadores para el encendido de lámparas y un servomotor para la apertura de puerta de la cochera.

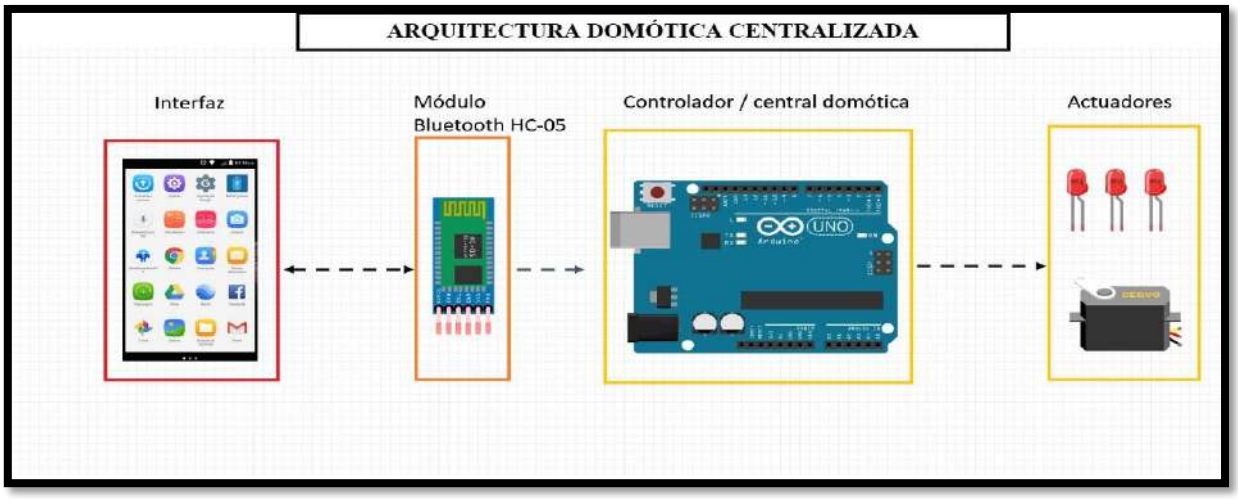


Figura 20. Arquitectura centralizada.

Fuente: Imagen de elaboración propia, basada en el diagrama de http://repositorio.upct.es/bitstream/handle/10317/2793/pfc4381.pdf

Configuración del Arduino UNO

Para el sistema desarrollado, se configuraron 6 salidas digitales del microcontrolador, en las terminales 5, 6, 7, 8, 9 y 10, las cuales se conectaron LEDs con una resistencia de 220 ohms para proteger la tarjeta Arduino y a los LEDs, ya que esta parte del sistema domótico simula la activación de lámparas eléctricas. El ánodo de cada LED se conecta a 5 V a través de la resistencia, y el cátodo se conecta a la terminal gnd del microcontrolador.

La Figura 21 muestra el diagrama electrónico de esta parte del sistema. Con la finalidad de poder presentar una forma más realista y clara de las conexiones físicas.

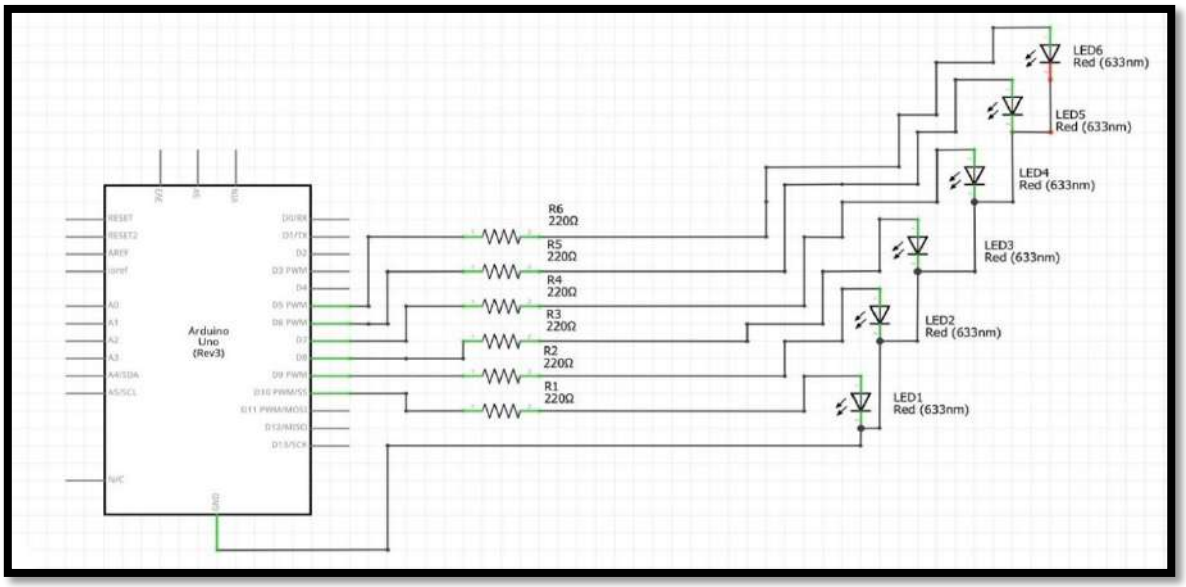


Figura 21. Diagrama electrónico para las salidas digitales de Arduino UNO.

Para tener una idea un poco más clara de cómo se realizó la conexión de los leds en un entorno real del ensamblaje del circuito en protoboard. La Figura 22 muestra cómo se realizó la forma de conectar cada led a su determinada terminal, teniendo en cuenta la protección de la tarjeta Arduino y el componente haciendo el uso adecuado de una resistencia de 220 ohms para cada conexión.

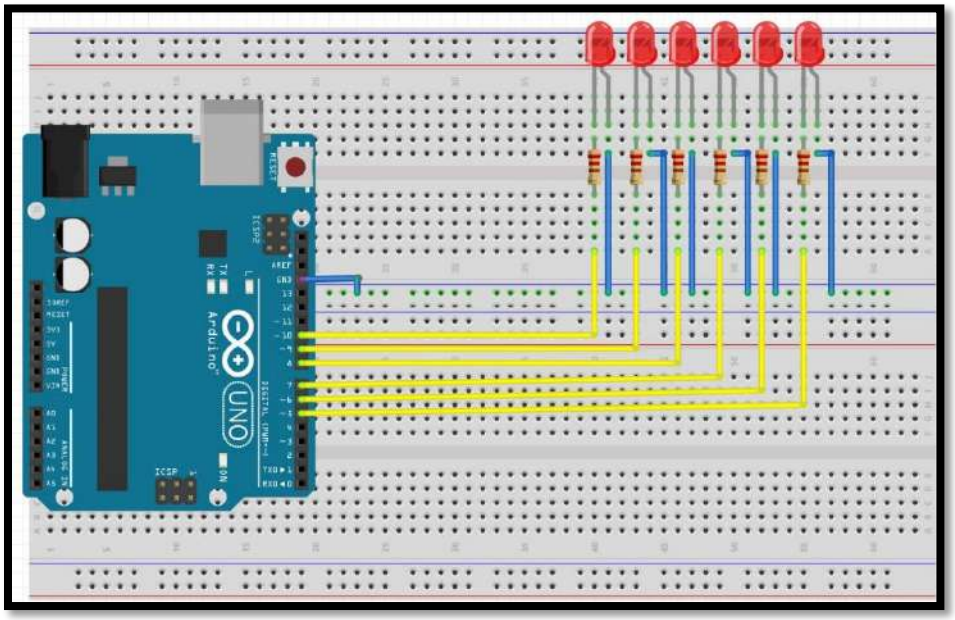


Figura 22. Esquema de conexión de LEDs en protoboard.

En el código para el microcontrolador se declaran variables de tipo entero asociadas a las terminales digitales 5, 6, 7, 8, 9 y 10 de Arduino donde se conectan los LEDs, posteriormente en la función “void setup” se inicializan las terminales como salidas al iniciar la tarjeta de Arduino UNO se puede ver en la Figura 23.

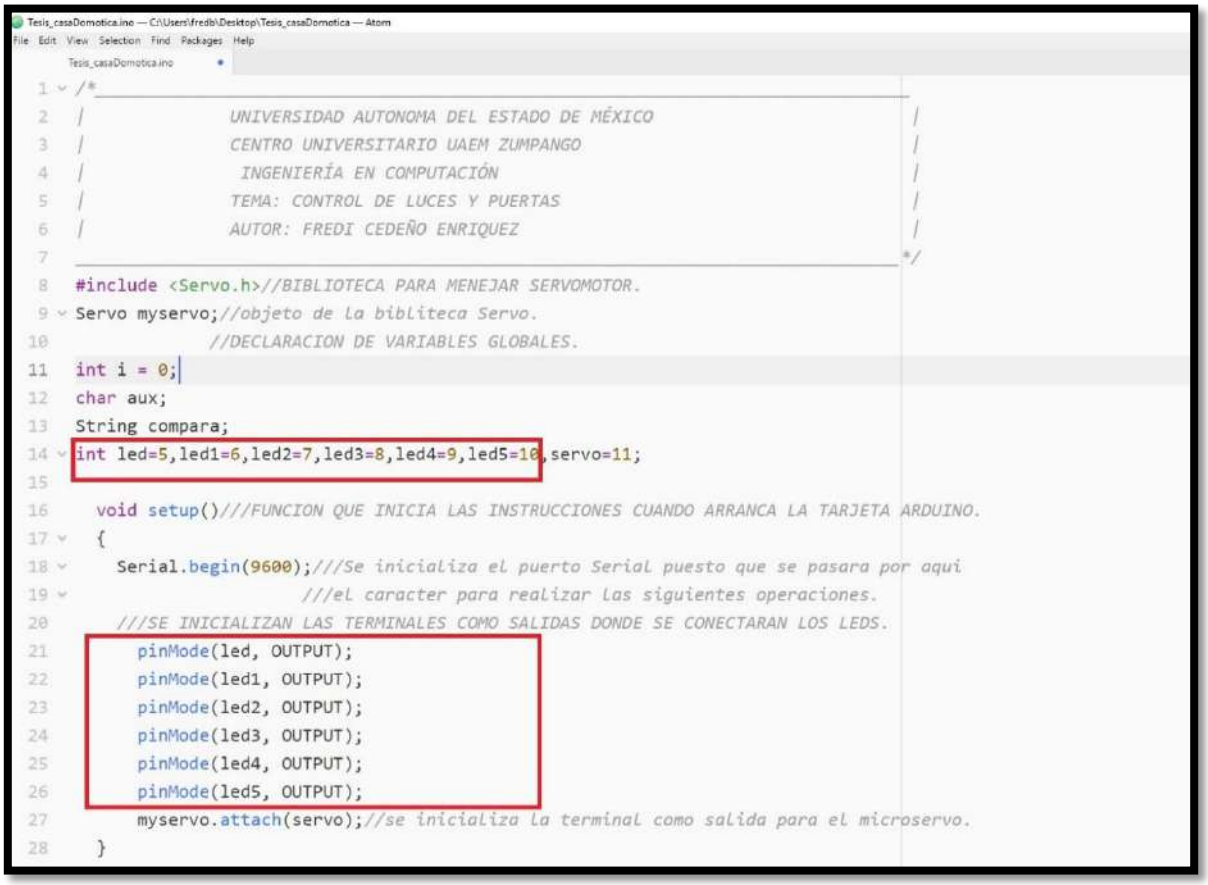


Figura 23. Declaración e inicialización de terminales digitales como salidas.

Bluetooth

Para la comunicación inalámbrica entre la tarjeta Arduino y la aplicación móvil, fue necesario usar como intermediario a un módulo Bluetooth HC-05 que recibe información enviada por la aplicación Android hacia la unidad de control, y así puedan activarse los actuadores. En la Figura 25 muestra el diagrama electrónico para poder configurar el Bluetooth HC-05 con la tarjeta Arduino, y así poder llevar a cabo la comunicación.

Es muy importante destacar que al realizar la conexión de las terminales TX y RX del módulo bluetooth HC-05 con las terminales TX y RX de la tarjeta Arduino. La configuración de estas terminales debe ser realizada de la siguiente forma para que se pueda llevar a cabo correctamente la comunicación entre estos dispositivos, como se muestra en la Figura 24, en otras palabras, la terminal TX de Arduino debe conectarse con RX del módulo HC-05, de igual manera el TX de Bluetooth HC-05 con el RX de Arduino, intercalando las terminales.

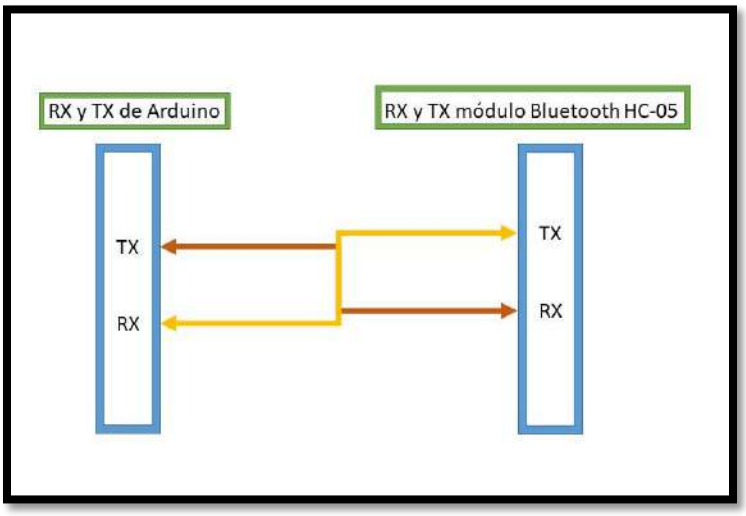


Figura 24. Configuración de terminales de Arduino y Bluetooth HC-05.

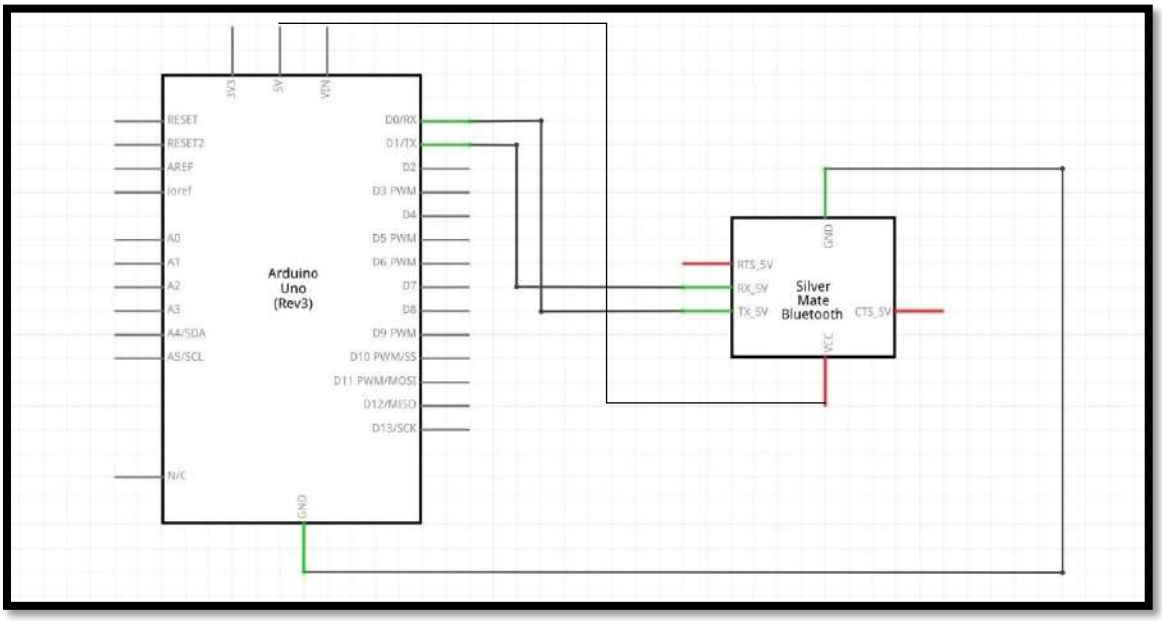


Figura 25. Diagrama electrónico para el módulo Bluetooth HC-05.

Para tener una idea más amplia de como configurar la conexión de Vcc, gnd, TX y RX del módulo Bluetooth con Arduino en un entorno más real, se observa en la Figura 26 el circuito ensamblado en protoboard.

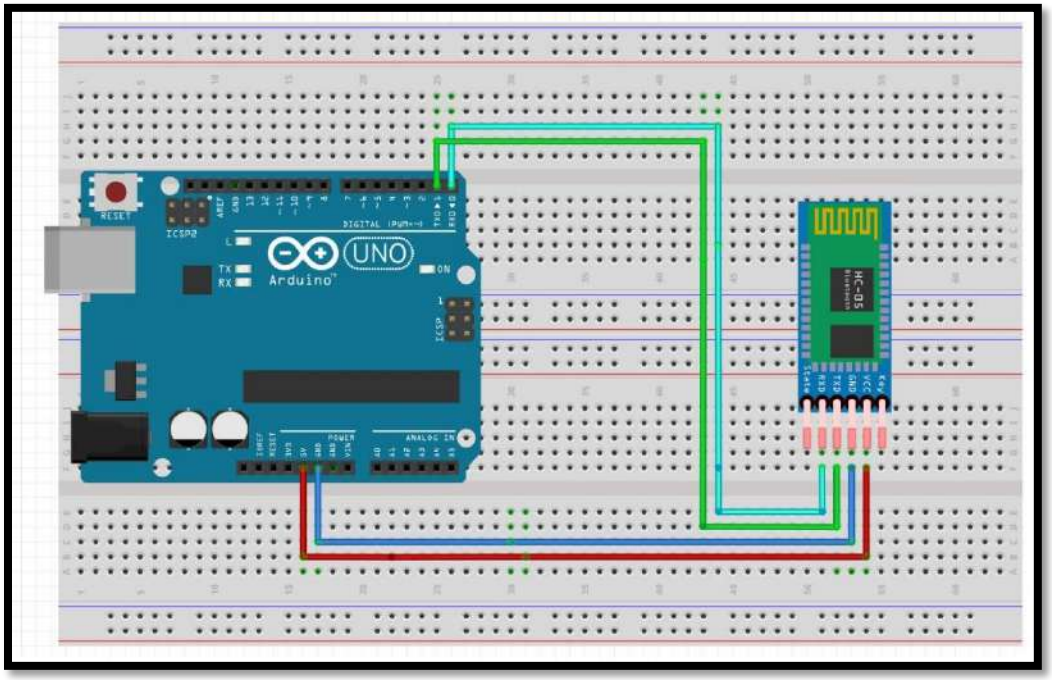


Figura 26. Conexión y alimentación eléctrica del módulo Bluetooth HC-05 con Arduino UNO.

Para la comunicación con el módulo Bluetooth, se configuró el software del microcontrolador para habilitar el puerto serial de Arduino UNO. En código se configuró el puerto serial de Arduino UNO con una velocidad de 9600 de transferencia de datos, para realizar la comunicación entre el módulo bluetooth HC-05 y la tarjeta Arduino como se muestra en la Figura 27.

```
14
15  void setup()///FUNCION QUE INICIA LAS INSTRUCCIONES CUANDO ARRANCA LA TARJETA ARDUINO.
16  {
17    Serial.begin(9600);//Se inicializa el puerto Serial puesto que se pasara por aqui
18                       ///el caracter para realizar las siguientes operaciones.
19    ///SE INICIALIZAN LAS TERMINALES COMO SALIDAS DONDE SE CONECTARAN LOS LEDS.
20      pinMode(led, OUTPUT);
21      pinMode(led1, OUTPUT);
22      pinMode(led2, OUTPUT);
23      pinMode(led3, OUTPUT);
24      pinMode(led4, OUTPUT);
25      pinMode(led5, OUTPUT);
26      myservo.attach(servo);//se inicializa la terminal como salida para el microservo.
27  }
28
29  void loop()//FUNCIÓN CICLICA QUE EJECUTA UNA Y OTRA VES CADA INSTRUCCIÓN.
30  {
31    if (Serial.available() > 0) // SE comprueba si el puerto esta libre y disponible.
32     {
33       compara = "";// VARIABLE compara se inicializa.
34
35      }
```

Figura 27. Inicialización y comprobación del puerto serial.

Servomotor

Este bloque del sistema domótico se configuró la terminal digital 11 de Arduino como salida, conectando el cable del servo con una resistencia de 330 ohms para proteger los mecanismos, como se muestra en la Figura 28. Además, se conectó a Vcc y tierra las otras dos terminales del servo, para alimentarlo con corriente eléctrica. Esta sección del sistema tiene como objetivo simular la apertura y cierre de la puerta de la cochera.

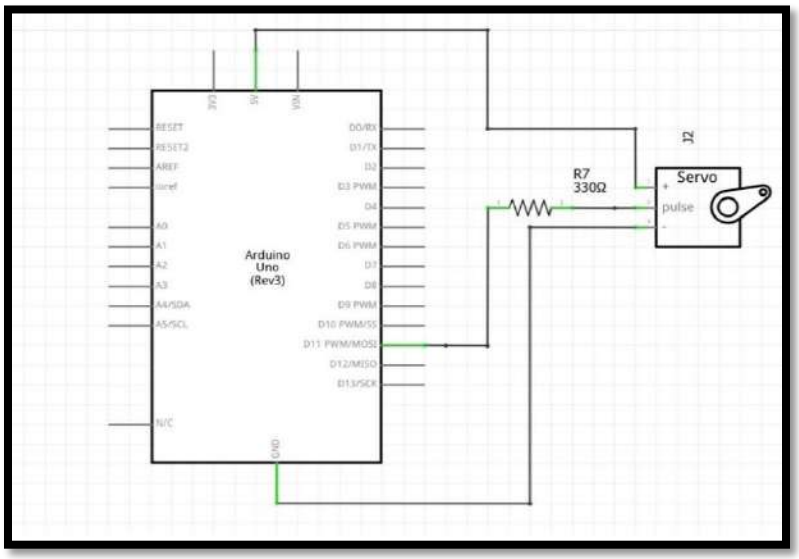


Figura 28. Diagrama de conexiones de Micro Servo Motor SG90 9g con Arduino UNO.

Para hacer más descriptiva la explicación, se presenta el circuito del servomotor en la Figura 29. Se puede observar que el cable de color amarillo (o de control) del servo es conectado con una resistencia de 330 ohms a la terminal digital 11 de Arduino; de igual forma, el cable de color rojo se conecta a 5v, y el cable de color negro a tierra de la tarjeta Arduino.

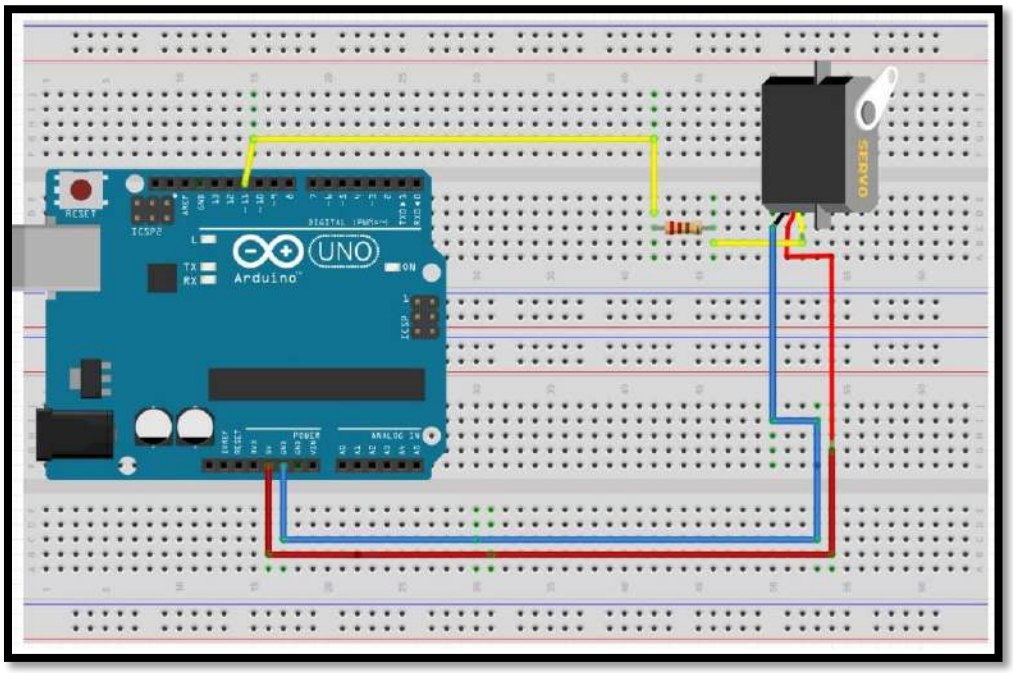


Figura 29. Ensamblado de circuito de Arduino y Servomotor en protoboard.

Para manejar el micro servomotor SG90 9g con Arduino, es necesario incluir en el código de programación la biblioteca <Servo.h>, permitiendo manipular a la gran mayoría de los servomotores, en este caso se declara un objeto de tipo Servo para posteriormente inicializar la terminal 11 de Arduino como salida. Dicha terminal 11 es declarada previamente en la sección de variables como (servo = 11) de tipo entero. Estos pasos se presentan en la Figura 30.

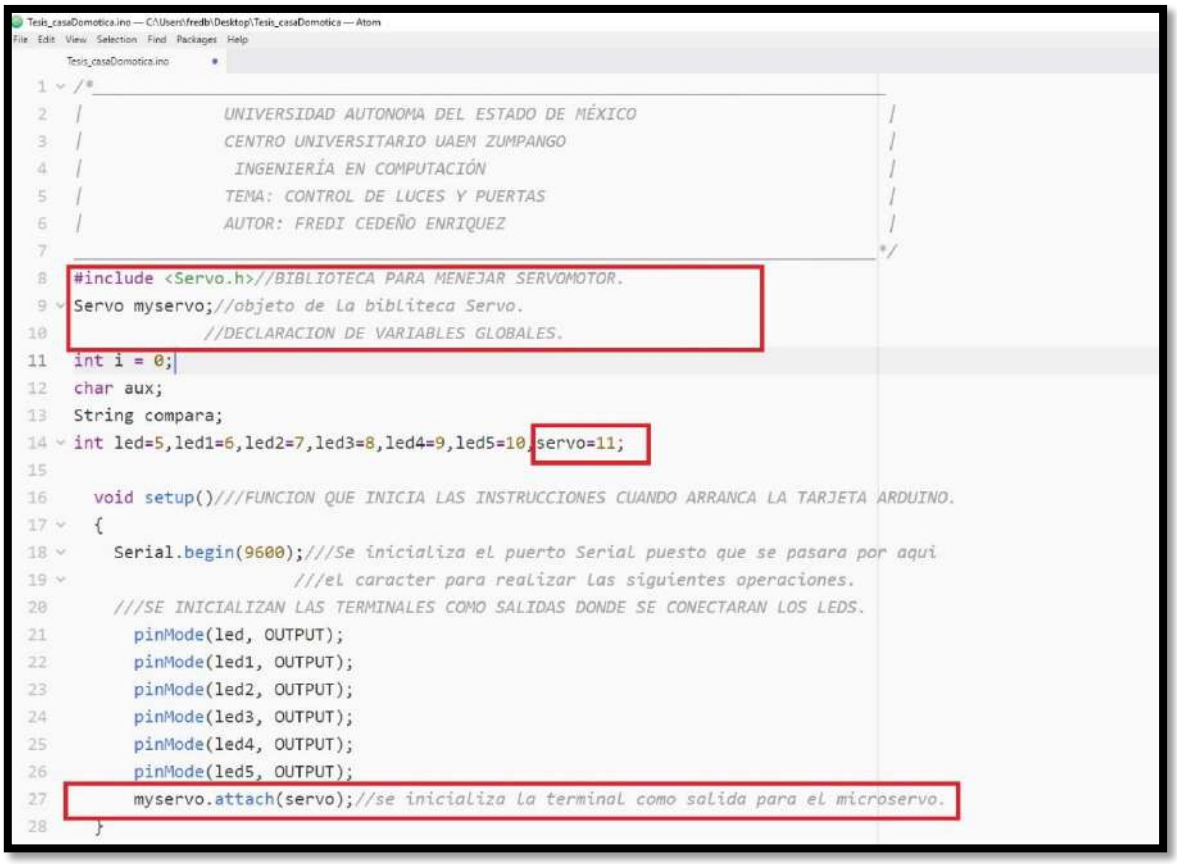


Figura 30. Configuración de servomotor en código Arduino.

Vinculación del teléfono inteligente con el módulo Bluetooth HC05

Para hacer uso de la aplicación móvil, se tiene que realizar una serie de pasos para configurar el modulo Bluetooth HC-05, como los que se muestran a continuación en cada figura. Primero, se debe de seleccionar la función Bluetooth en el teléfono móvil (ver Figura 31), para desplegar la lista de dispositivos a vincular. EL Shield de Arduino se identifica con el nombre “HC-05”, y se agrega a la lista de vinculados del Smartphone.

Figura 31. Módulo Bluetooth HC-05 a vincular con dispositivo móvil.

En el caso de ser la primera vez que se vincula el módulo HC-05 con el Smartphone, se nos pedirá un *pin* de seguridad de cuatro dígitos, como se muestra en la Figura 32. El *pin* predeterminado del módulo puede ser “1234” o “0000”.

Figura 32. Ingreso de pin de seguridad.

La Figura 33 muestra la validación del *pin* de seguridad del módulo Bluetooth HC-05, para posteriormente ser parte de la lista de dispositivos vinculados en el teléfono móvil.

Figura 33. Validación del pin de seguridad.

Una vez realizado todo el proceso anterior y que ha sido vinculado al teléfono móvil correctamente, en el Smartphone debe mostrarse algo similar a la Figura 34.

Figura 34. Módulo bluetooth HC-05 ya es parte de la lista de vinculados del teléfono móvil.

Aplicación Android para el sistema domótico

Funcionamiento general

El icono de la app se puede observar en la Figura 35, de forma instalada en el menú de aplicaciones del teléfono móvil la cual lleva por nombre “CasaDomoticaAPP” lista para ser usada por el usuario.

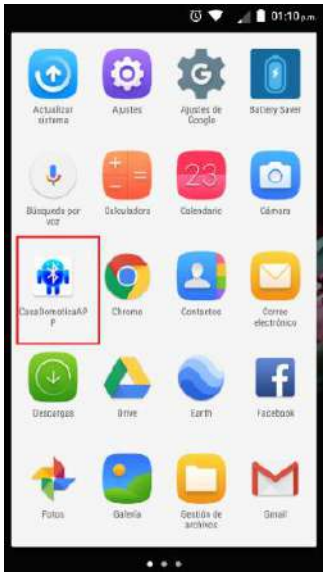


Figura 35. Icono de la App instalada en teléfono móvil Android.

Cada vez que se ejecuta la app, esta comprueba que el Bluetooth del teléfono móvil está activo, en caso contrario, esta mostrará una notificación (mediante un control Toast) indicando que el bluetooth del teléfono móvil no está activo, y pedirá permisos al usuario para activarlo, como se muestra en la siguiente Figura 36.

Figura 36. Activar bluetooth del equipo móvil en caso de estar desactivado.

La Figura 37 muestra la pantalla inicial de la App, la cual está compuesta por dos botones, el primero con la etiqueta “lista vinculados” y de color azul, mientras el segundo botón es de color amarillo y una etiqueta con el texto “Salir”.

Figura 37. Vista inicial de la App integrada por 2 botones iniciales.

Al seleccionar el botón de “lista vinculados” se despliega la lista de dispositivos emparejados al teléfono móvil, para posteriormente seleccionar el módulo bluetooth con el nombre HC-05, como lo muestra la Figura 38.

Figura 38. Selección del botón "lista vinculados" y despliegue de lista de dispositivos vinculados al Smartphone.

Una vez seleccionado el módulo HC-05 de la Lista de Dispositivos Vinculados, se permitirá su uso en la App. Luego se presenta la vista principal (ver Figura 39) que controla las luces de la vivienda y la puerta de la cochera. El usuario tiene el control para encender o apagar las lámparas de cada sección de la casa, puesto que la aplicación integra un croquis arquitectónico, donde es fácil identificar el botón correspondiente.

Figura 39. Botones en modo espera.

Cuando el usuario quiere activar la lámpara de una sección de la casa, solamente presiona el botón de un lugar específico, por ejemplo, una habitación o la cochera. El usuario puede percatarse que las lámparas de la vivienda están encendidas o el caso de la cochera, la puerta está abierta, debido a que el botón seleccionado cambia a color **verde** indicando que los botones están activos (ver Figura 40).

Figura 40. Botones en modo Activo.

El usuario puede percatarse que las lámparas de cada sección de la casa han sido apagadas, debido a que cada botón cambia de color **verde a rojo** indicando que está en modo desactivado, como se muestra en la Figura 41.

Figura 41. Botones en modo Desactivado.

Desarrollo de la aplicación Android

Al diseñar una aplicación en Android Studio, esta plataforma permite utilizar un lenguaje de desarrollo de interfaz de usuario o (UI por sus siglas en inglés Interface User Interface), siendo XML (Extensible Markup Language) que es un grupo de normas para codificar, además de ser un formato importante para compartir información en Internet, haciendo fácil la estructuración de la forma visual de una aplicación en Android [49].

La Figura 42 muestra el diseño de la vista principal de la App desarrollada en XML, integrada por 2 botones; el primero es conformado con un fondo azul y una etiqueta con el texto “LISTA VINCULADOS”, el segundo botón es integrado con un fondo amarillo y una etiqueta con la palabra “SALIR”, estos dos componentes mencionados están dentro de un gestor *linearLayout* (vertical) para mantener la alineación y distancia, además de contar con un control *listView* (lista) que contiene los nombres de los dispositivos vinculados y un control *textView* (etiqueta). Estos otros elementos mencionados están dentro de otro gestor *linearLayout*, haciendo que los componentes se mantengan organizados correctamente.

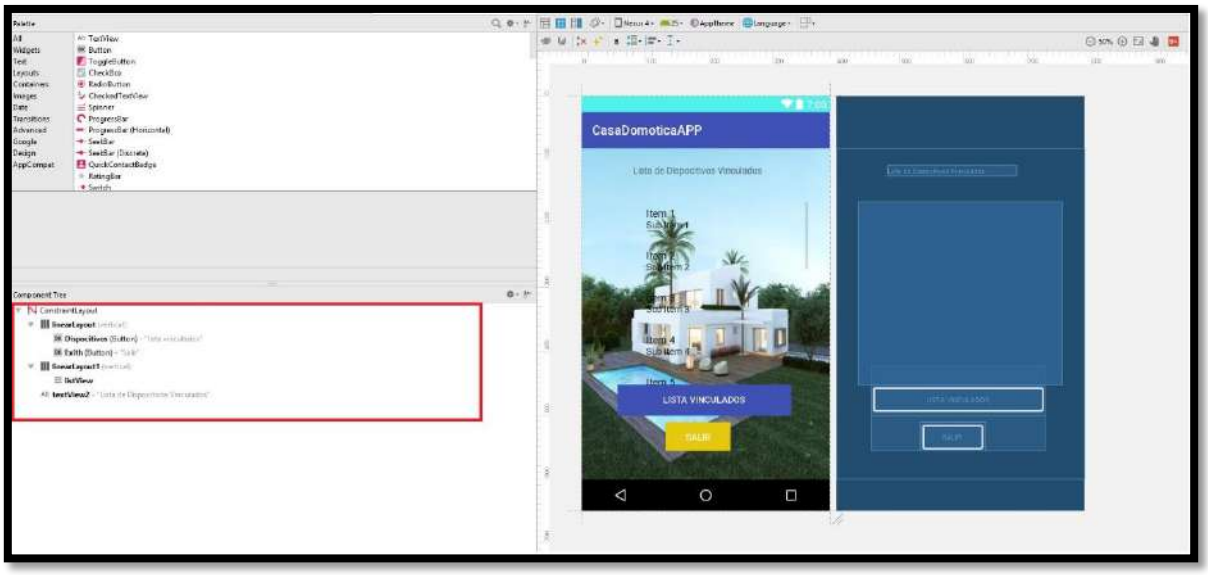


Figura 42. Diseño en XML de la vista principal de la App del sistema domótico en Android Studio.

La codificación en XML del diseño de los dos botones iniciales de la aplicación antes expuesta, se muestra en la Figura 43. El usar XML permite desarrollar vistas bastante intuitivas para el usuario.

```
<LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="269dp"
    android:layout_height="133dp"
    android:layout_marginStart="55dp"
    android:orientation="vertical"
    android:weightSum="1"
    tools:layout_constraintBottom_creator="1"
    app:layout_constraintBottom_toBottomOf="parent"
    tools:layout_constraintLeft_creator="1"
    android:layout_marginBottom="44dp"
    app:layout_constraintLeft_toLeftOf="parent">

    <Button
        android:id="@+id/Dispocitivos"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:background="@color/colorPrimary"
        android:text="lista vinculados"
        android:textColor="@android:color/background_light" />

    <Button
        android:id="@+id/Exith"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="75dp"
        android:layout_marginTop="10dp"
        android:layout_weight="0.03"
        android:background="@color/colorAccent"
        android:text="Salir"
        android:textColor="@android:color/background_light" />
</LinearLayout>
```

Figura 43. Código XML de los botones "SALIR" y "LISTA VINCULADOS".

Esta parte del código XML realiza el trabajo de ordenar de forma visual la lista empleada que contiene los nombres de los dispositivos vinculados al teléfono móvil y la separación con una etiqueta que muestra el nombre de esta lista, en la Figura 44 se puede observar que algunos elementos que conforman el diseño como el "id" que es utilizado en las clases java para poder codificar el comportamiento del componente ya sea un botón, una lista o entre otros elementos.

```
        <ListView
            android:id="@+id/listView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_alignParentLeft="false"
            android:layout_alignParentStart="true"
            android:layout_below="@+id/textView"
            android:layout_marginLeft="0dp"
            android:paddingStart="50dp"
            tools:layout_editor_absoluteX="8dp"
            tools:layout_editor_absoluteY="8dp" />
    </LinearLayout>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="45dp"
        android:text="Lista de Dispocitivos Vinculados"
        tools:layout_constraintTop_creator="1"
        tools:layout_constraintRight_creator="1"
        android:layout_marginEnd="45dp"
        app:layout_constraintRight_toRightOf="@+id/linearLayout"
        android:layout_marginTop="26dp"
        tools:layout_constraintLeft_creator="1"
        app:layout_constraintLeft_toLeftOf="@+id/linearLayout1"
        app:layout_constraintTop_toTopOf="parent" />
```

Figura 44. Código XML de la lista y su etiqueta.

Para el diseño de esta parte de la aplicación del control domótico se implementó un fondo de pantalla de un croquis que simula cada sección de una casa habitación, posteriormente se colocaron 7 componentes. Se utilizó un control **ToggleButton** que permite alternar la configuración de dos estados él encender o apagar, además de un botón con fondo azul que permite desconectar la aplicación y regresar a la vista anterior, haciendo más fácil la interacción del usuario con la interfaz.

La Figura 45 muestra la forma en que los controles **ToggleButton** fueron distribuidos. Uno para cada una las siguientes partes de la casa: primera habitación, segunda habitación, sala, cocina-comedor, baño. Dos para en la cochera. Un botón realiza el encendido y apagado de las luces como los demás distribuidos en cada sección del hogar, y otro activa o desactiva el servomotor que controla la puerta de acceso a la cochera.

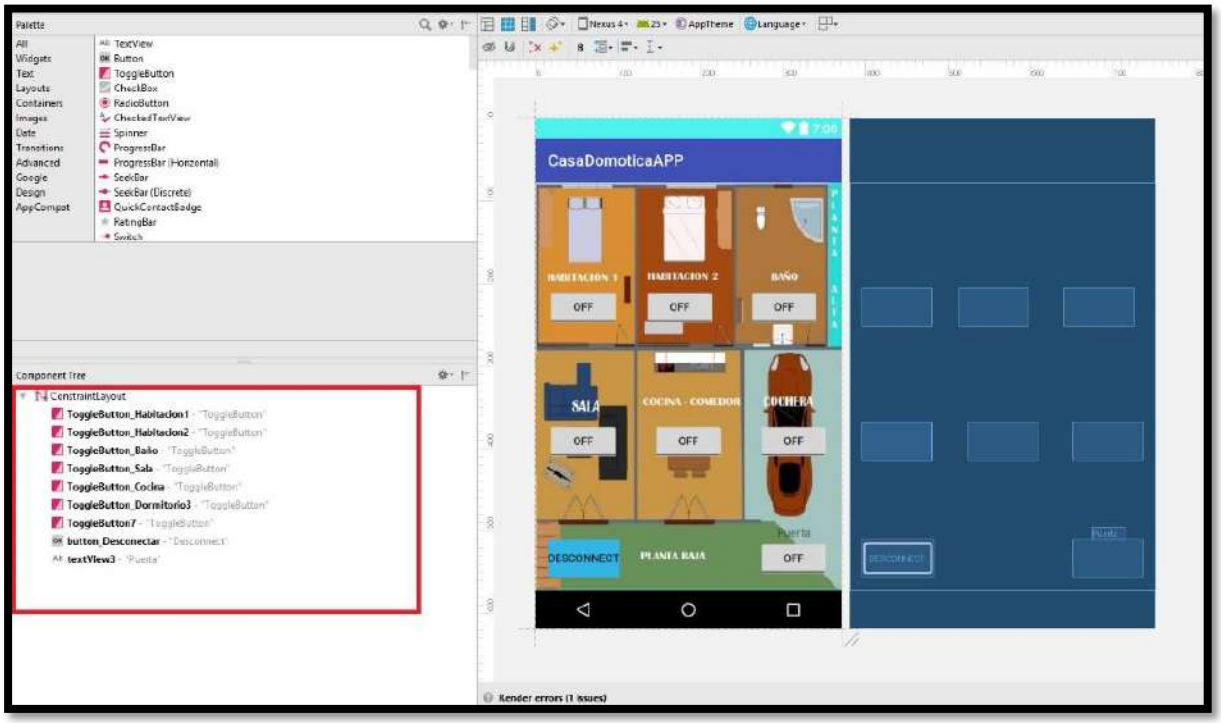


Figura 45. Diseño en XML, control de botones de la vista secundaria de la App.

El código de XML "id=@+id/ToggleButton_Habitacion1" es muy importante para cada control *ToggleButton*. Este atributo hace referencia con su nombre para poder ser usados en la programación de su comportamiento (gestor de eventos); además, cuenta con atributos indispensables que hacen que su estética de diseño sea más agradable al usuario como se ve en la Figura 46. También en esta parte de XML, el diseño de los ToggleButton nos permite distribuir de forma correcta cada botón tomando en cuenta cada espacio y posición, para poder ver cada componente ordenado y alineado.

```
<ToggleButton
    android:id="@+id/ToggleButton_Habitacion1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:text="ToggleButton"
    android:textOff="off"
    android:textOn="on"
    app:layout_constraintBaseline_toBaselineOf="@+id/ToggleButton_Habitacion2"
    app:layout_constraintLeft_toLeftOf="parent"
    tools:layout_constraintBaseline_creator="1"
    tools:layout_constraintLeft_creator="1" />

<ToggleButton
    android:id="@+id/ToggleButton_Habitacion2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="33dp"
    android:layout_marginTop="132dp"
    android:text="ToggleButton"
    android:textOff="off"
    android:textOn="on"
    app:layout_constraintLeft_toRightOf="@+id/ToggleButton_Habitacion1"
    app:layout_constraintTop_toTopOf="parent"
    tools:layout_constraintLeft_creator="1"
    tools:layout_constraintTop_creator="1" />

<ToggleButton
    android:id="@+id/ToggleButton_Baño"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="28dp"
    android:layout_marginRight="28dp"
    android:text="ToggleButton"
    android:textOff="off"
    android:textOn="on"
    app:layout_constraintBaseline_toBaselineOf="@+id/ToggleButton_Habitacion2"
```

Figura 46. Código XML de los ToggleButton que simular el control de lámparas.

La Figura 47 muestra el diseño del botón con la etiqueta “Desconnect”, la cual se puede observar los atributos que conforman el diseño de este botón, como el “id= @+id/button_Desconectar”. Este elemento es muy importante ya que es el que se usa para poder programar su comportamiento en la clase java orientada para este componente, además de permitir que acciones debe realizar este elemento de la App, el cual destaca su función de desconectarse y regresar a la vista anterior.

```
<ToggleButton
    android:id="@+id/ToggleButton7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginEnd="16dp"
    android:text="ToggleButton"
    android:textOff="off"
    android:textOn="on"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    tools:layout_constraintBottom_creator="1"
    tools:layout_constraintRight_creator="1" />

<Button
    android:id="@+id/button_Desconectar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginStart="16dp"
    android:background="@android:color/holo_blue_light"
    android:text="Desconnect"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    tools:layout_constraintBottom_creator="1"
    tools:layout_constraintLeft_creator="1" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Puerta"
    tools:layout_constraintBottom_creator="1"
    app:layout_constraintBottom_toTopOf="@+id/ToggleButton7"
    tools:layout_constraintLeft_creator="1"
    android:layout_marginBottom="0dp"
    app:layout_constraintLeft_toLeftOf="@+id/ToggleButton7"
```

Figura 47. Código XML del botón desconectar.

Los diagramas de clases son muy utilizados en el diseño de software ya que son útiles para describir la estructura de un sistema, como la visualización de sus clases, sus atributos (características) y métodos (comportamiento), además de mostrar la relación entre objetos [50].

La Figura 48 describe la estructura de la aplicación desarrollada en Android para el control del sistema domótico propuesto para una casa habitación, esta descripción se hace con un diagrama de clases desarrollado en UML (LENGUAJE UNIFICADO DE MODELADO).

Esta aplicación es integrada por la clase “MainActivity_ConexionBluetooth” que hereda de “AppCompatActivity” que es una súper clase de una Activity (Actividad) en Android el cual contiene métodos que puede heredar, además “MainActivity_ConexionBluetooth”, es dependiente de ella para poder funcionar.

La clase “Casa_control” es dependiente y hereda de la superclase “AppCompatActivity” ciertos métodos para lograr su funcionamiento. Por otra parte, también se desarrolló otra clase para poder llevar acabo el funcionamiento correcto de esta aplicación el cual lleva por nombre “Prender_Apagar” quien hereda de su clase padre “Casa_control” y está misma presenta una dependencia de la clase “Prender_Apagar” para poder funcionar.

La clase “ListView” es conjunto de vistas, y es una agregación de la clase “MainActivity_ConexionBluetooth” ya que es parte de esta clase, pero es dependiente de “ListView” para que funcione, permitiendo que sus componentes puedan ser compartidos a otras clases.

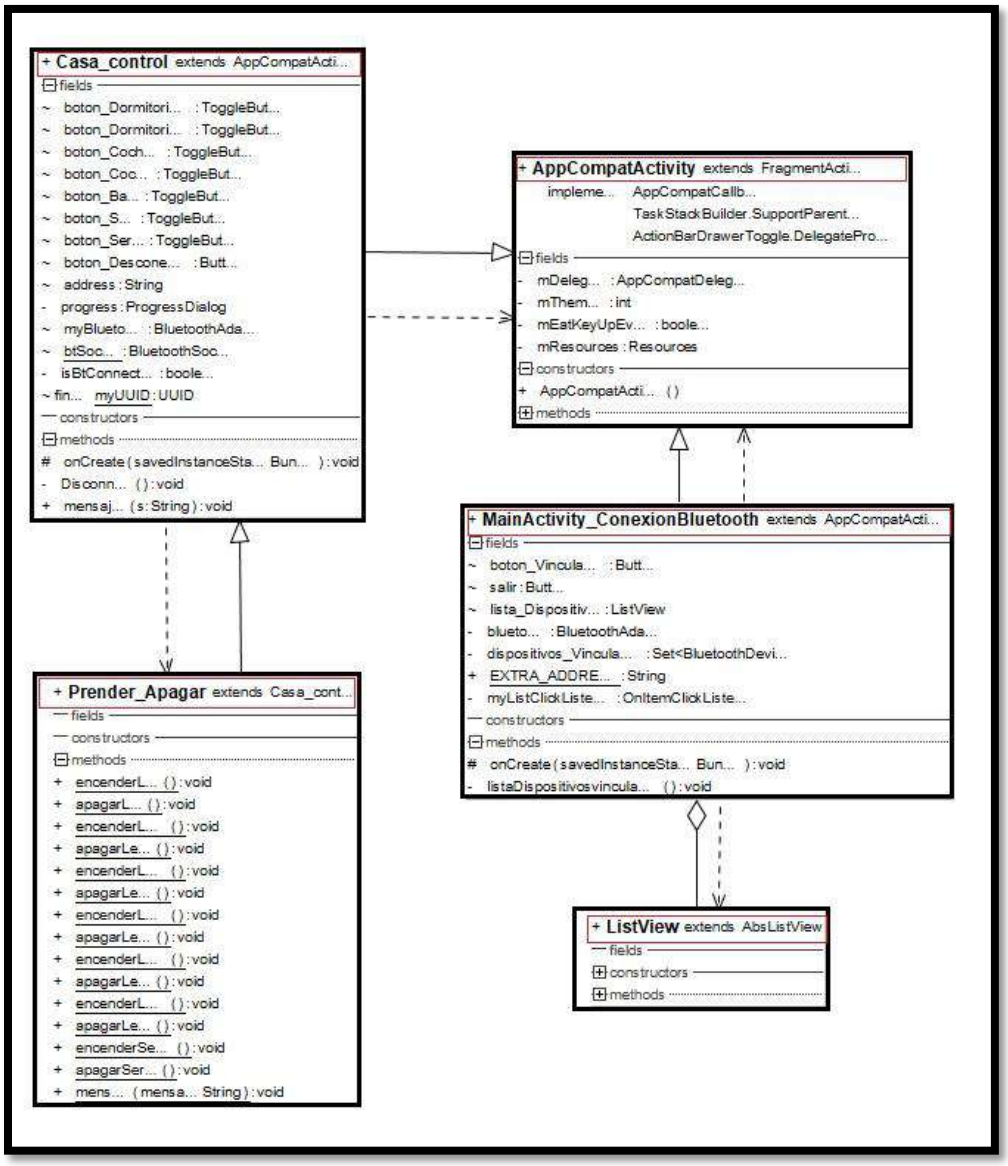
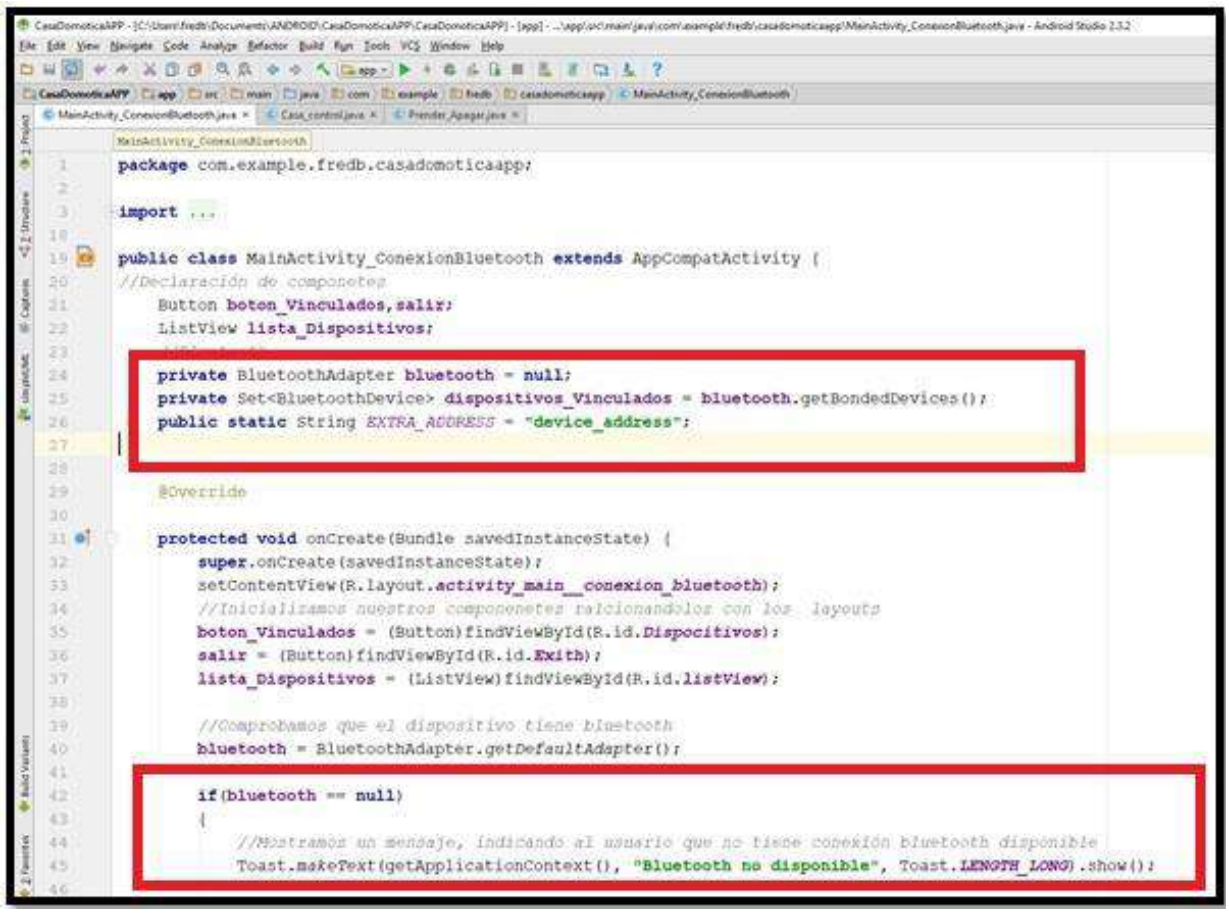


Figura 48. Diagrama de clases de la aplicación en Android para el control del sistema domótico.

La clase de Java "MainActivity_ConexionBluetooth", contiene la programación de la conexión Bluetooth, en la Figura 49 se puede observar en el primer recuadro la declaración de los objetos para llevar acabo la comprobación y acoplamiento de dispositivos al teléfono móvil, el segundo recuadro contiene el código que pregunta al usuario si el Bluetooth de su Smartphone esta desactivado, y posteriormente sea activado.



```
package com.example.fredb.casadomoticaapp;

import ...

public class MainActivity_ConexionBluetooth extends AppCompatActivity {
//Declaración de componetes
    Button boton_Vinculados,salir;
    ListView lista_Dispositivos;

    private BluetoothAdapter bluetooth = null;
    private Set<BluetoothDevice> dispositivos_Vinculados = bluetooth.getBondedDevices();
    public static String EXTRA_ADDRESS = "device_address";

    @Override

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main__conexion_bluetooth);
        //Inicializamos nuestros componentes relacionandolos con los  layouts
        boton_Vinculados = (Button)findViewById(R.id.Dispocitivos);
        salir = (Button)findViewById(R.id.Exith);
        lista_Dispositivos = (ListView)findViewById(R.id.listView);

        //Comprobamos que el dispositivo tiene bluetooth
        bluetooth = BluetoothAdapter.getDefaultAdapter();

        if(bluetooth == null)
        {
            //Mostramos un mensaje, indicando al usuario que no tiene conexión bluetooth disponible
            Toast.makeText(getApplicationContext(), "Bluetooth no disponible", Toast.LENGTH_LONG).show();
```

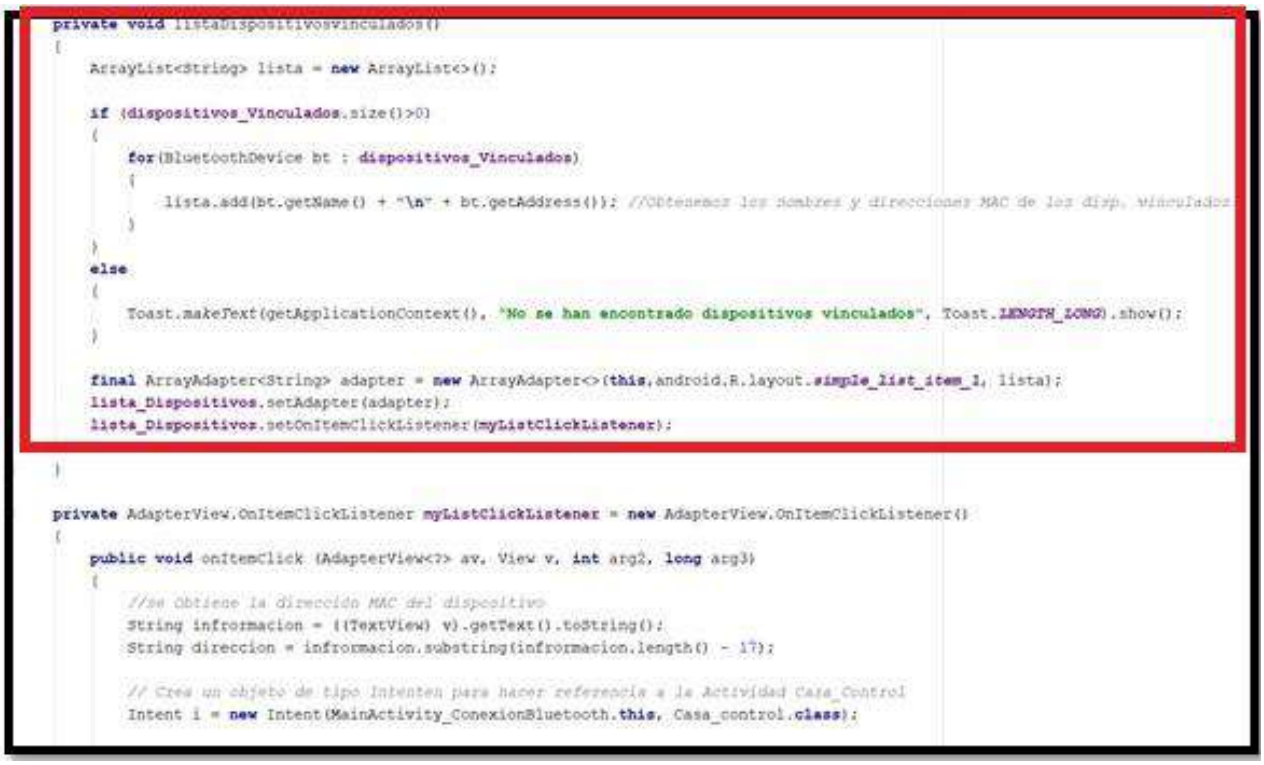
Figura 49. Código de la clase MainActivity_ConexionBluetooth en java.

La Figura 50 muestra las acciones que realiza el **"boton_Vinculados",** las cuales invocan la función **listaDispositivosvinculados()**, que despliega los dispositivos emparejados al teléfono móvil, y posteriormente seleccionar el módulo Bluetooth correspondiente para la comunicación del sistema domótico, también se puede ver el código del botón **salir**, este invoca la función (**exit(0))** para salir la aplicación.

Figura 50. Código de los botones Salir y lista vinculados.

Esta parte de código muestra el funcionamiento de "listView", este elemento enlista el nombre y la dirección MAC de los dispositivos vinculados al teléfono móvil, como se puede ver en la Figura 51.

Otro elemento importante se puede observar en la figura antes mencionada, es el uso del **(Intent)** con él se puede instanciar una nueva **Activity,** en este caso se crea un objeto de tipo **Intent** para poder ser enviado a través de la función **startActivity(),** y está a su vez permita iniciar la clase Activity Casa_control.



```
private void listaDispositivosVinculados()
{
    ArrayList<String> lista = new ArrayList<>();

    if (dispositivos_Vinculados.size()>0)
    {
        for(BluetoothDevice bt : dispositivos_Vinculados)
        {
            lista.add(bt.getName() + "\n" + bt.getAddress()); //Obtenemos los nombres y direcciones MAC de los disp. vinculados
        }
    }
    else
    {
        Toast.makeText(getApplicationContext(), "No se han encontrado dispositivos vinculados", Toast.LENGTH_LONG).show();
    }

    final ArrayAdapter<String> adapter = new ArrayAdapter<>(this,android.R.layout.simple_list_item_1, lista);
    lista_Dispositivos.setAdapter(adapter);
    lista_Dispositivos.setOnItemClickListener(myListClickListener);
}

private AdapterView.OnItemClickListener myListClickListener = new AdapterView.OnItemClickListener()
{
    public void onItemClick (AdapterView<?> av, View v, int arg2, long arg3)
    {
        //se Obtiene la direccion MAC del dispositivo
        String infromacion = ((TextView) v).getText().toString();
        String direccion = infromacion.substring(infromacion.length() - 17);

        // Crea un objeto de tipo Intenten para hacer referencia a la Actividad Casa_Control
        Intent i = new Intent(MainActivity_ConexionBluetooth.this, Casa_control.class);
```

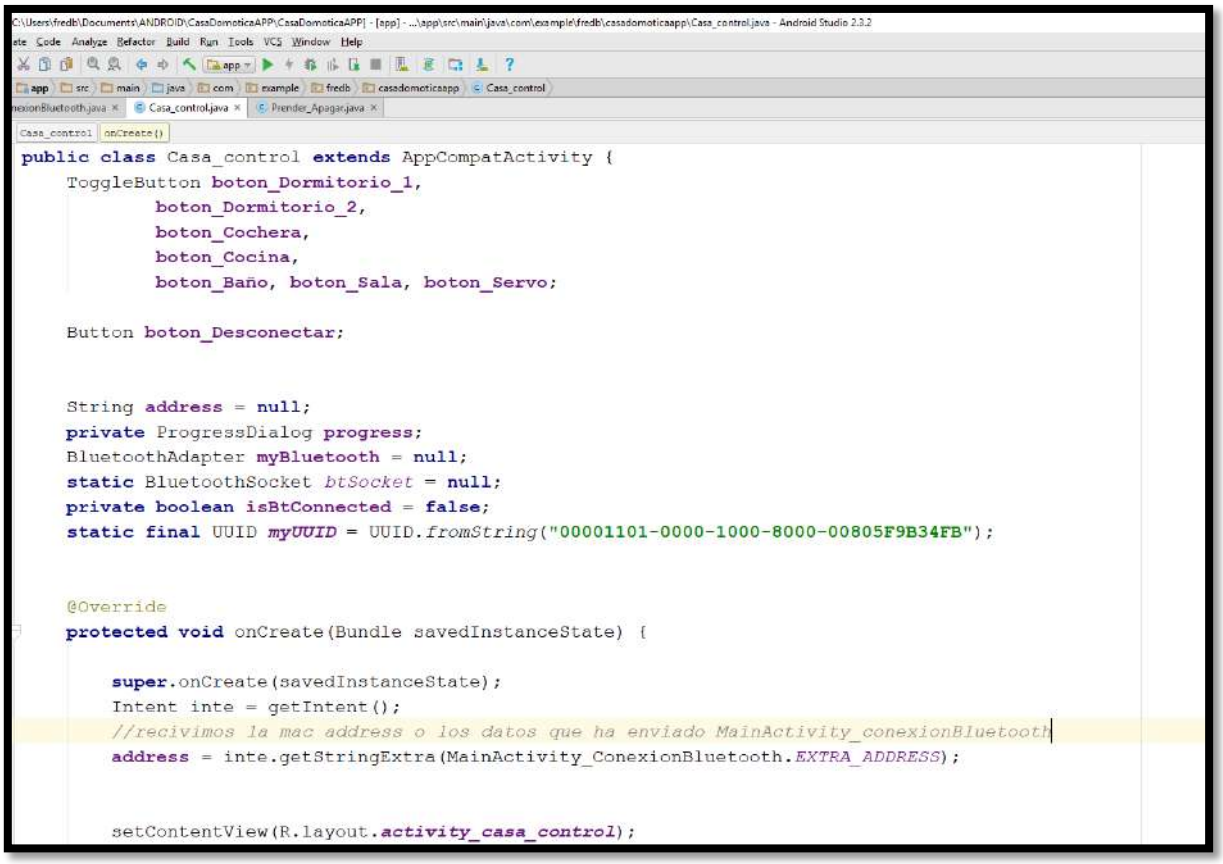
Figura 51. Código de la lista de dispositivos (listView).

La clase **"Casa_control" (**ver Figura 52), contiene la programación del comportamiento de los botones que alternan el encendido y apagado de las luces de cada sección del hogar, de igual forma, el botón que controla la puerta de acceso a la cochera.

Para iniciar la (**Activity**) Casa_control esta recibe el objeto **Intent** que fue enviado desde la clase **MainActivity_ConexionBluetooth**, posteriormente recibe la dirección MAC y el nombre del módulo Bluetooth para realizar la comprobación correspondiente y pasar a realizar el manejo de la vista secundaria de control de la aplicación.

Esta sección se declararon siete botones de tipo **ToggleButton** empleados para el control de luces de la casa y puerta de la cochera**,** y un botón más de tipo ***Button*** para desconectar la vista **Casa_control.**

El uso de la variable **address** tipo String, es para almacenar la serie de caracteres de la dirección física del dispositivo Bluetooth vinculado al teléfono móvil.



```
public class Casa_control extends AppCompatActivity {
    ToggleButton boton_Dormitorio_1,
            boton_Dormitorio_2,
            boton_Cochera,
            boton_Cocina,
            boton_Baño, boton_Sala, boton_Servo;

    Button boton_Desconectar;


    String address = null;
    private ProgressDialog progress;
    BluetoothAdapter myBluetooth = null;
    static BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");


    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        Intent inte = getIntent();
        //recivimos la mac address o los datos que ha enviado MainActivity_conexionBluetooth
        address = inte.getStringExtra(MainActivity_ConexionBluetooth.EXTRA_ADDRESS);


        setContentView(R.layout.activity_casa_control);
```

Figura 52. Código de la clase Casa_control en java.

Esta parte de código se puede observar la inicialización de los botones que controlan las luces y puerta de la cochera, también se puede ver el código del comportamiento del botón "Descontentar", como se ve en la Figura 53.

```
boton_Desconectar = (Button) findViewById(R.id.button_Desconectar);
boton_Servo = (ToggleButton) findViewById(R.id.ToggleButton7);
boton_Dormitorio_1 = (ToggleButton) findViewById(R.id.ToggleButton_Habitacion1);
boton_Dormitorio_2 = (ToggleButton) findViewById(R.id.ToggleButton_Habitacion2);
boton_Cochera = (ToggleButton) findViewById(R.id.ToggleButton_Dormitorio3);
boton_Cocina = (ToggleButton) findViewById(R.id.ToggleButton_Cocina);
boton_Baño = (ToggleButton) findViewById(R.id.ToggleButton_Baño);
boton_Sala = (ToggleButton) findViewById(R.id.ToggleButton_Sala);
new ConnectBT().execute();

boton_Desconectar.setOnClickListener(
        new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Disconnect();
    }
});
```

Figura 53. Inicialización de cada botón del control domótico.

Cabe destacar que las líneas de código de la Figura 54 hacen referencia al comportamiento de los controles **ToogleButton** que controlan las luces de la casa, se puede notar que el alternado de prender y apagar, esto es controlado por una condición if-else; cuando el botón está en modo activo se llama a la función **encenderLed()**, y establece la propiedad ***Background*** (fondo) del botón a color Verde, cuando el botón está en modo desactivado, es llamada la función apagarLed() y la propiedad Background del botón se establece de color rojo.

```
boton_Dormitorio_1.setOnCheckedChangeListener((buttonView, isChecked) → {
        if (isChecked) {

            encenderLed();
            boton_Dormitorio_1.setBackgroundColor(Color.GREEN);

        } else {

            apagarLed();
            boton_Dormitorio_1.setBackgroundColor(Color.RED);

        }
});


boton_Dormitorio_2.setOnCheckedChangeListener((buttonView, isChecked) → {
        if (isChecked) {
            encenderLed1();
            boton_Dormitorio_2.setBackgroundColor(Color.GREEN);
        } else {
            apagarLed1();
            boton_Dormitorio_2.setBackgroundColor(Color.RED);

        }
});
```

Figura 54. Acciones de los botones del control domótico.

Esta la sección de código mostrada en la Figura 55, realiza el trabajo para la conexión entre la aplicación instalada en el teléfono móvil y el módulo Bluetooth, para poder establecer la comunicación entre estos dos elementos que integran para el funcionamiento del sistema domótico.

```
//////////////////////////
private class ConnectBT extends AsyncTask<Void, Void, Void>
{
    private boolean ConnectSuccess = true;

    @Override
    protected void onPreExecute() {
        progress = ProgressDialog.show(Casa_control.this, "Conectando...", "Espere Por favor!!!");
    }

    @Override
    protected Void doInBackground(Void... devices) {
        try {
            if (btSocket == null || !isBtConnected) {
                myBluetooth = BluetoothAdapter.getDefaultAdapter();
                //conectamos al dispositivo y checamos si esta disponible
                BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(address);
                btSocket = dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);
                BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                btSocket.connect();
            }
        } catch (IOException e) {
            ConnectSuccess = false;
        }
        return null;
    }
```

Figura 55. Emparejamiento de la conexión Bluetooth.

En la Figura 56 se presenta el código que comprueba si la conexión se ha dado de forma correcta entre la App y el modulo Bluetooth. Si la conexión se realizó exitosamente, entonces se lanza una notificación con el texto “conectado”, en caso contrario se lanza el mensaje “conexión fallida”.

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

            if (!ConnectSuccess) {
                mensaje1("Conexión Fallida");
                finish();
            } else {
                mensaje1("Conectado");
                isBtConnected = true;
            }
            progress.dismiss();
        }
    }


}//clase Casa_control
```

Figura 56. Comprobación de la conexión.

En la Figura 57 se puede ver que la funciones encenderLed() contienen el carácter que es enviado a través de un Socket, que establece la comunicación con el puerto serial de Arduino, este carácter es usado en código Arduino para comparar que terminal está siendo accionada para activar la lámpara de la sección seleccionada de la casa, estas líneas de código están encerradas entre un **try y catch** el cual son usadas para el manejo de excepciones.

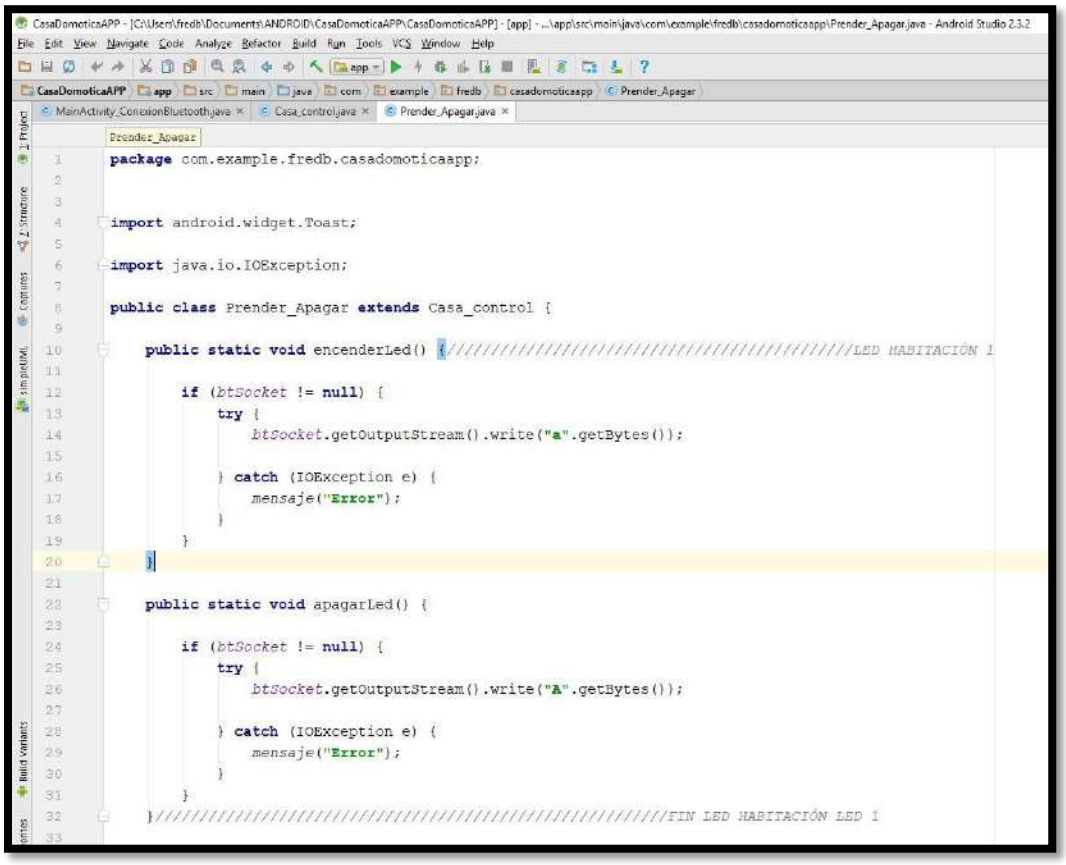


Figura 57. Código de la clase Prender_Apagar en java.

Diagrama de conexiones eléctricas generales del sistema domótico

El siguiente diagrama muestra las conexiones generales del sistema domótico presentado, este sistema es integrado por 6 leds, un módulo Bluetooth HC-05 y un Micro Servomotor SG90 9G.

La Figura 58 muestra que el ánodo de cada LED es conectado mediante una resistencia de 220 ohms a la terminal digital de la tarjeta Arduino, y el cátodo es conectado con el cátodo de los otros LED hacia GND o tierra. El servomotor es integrado por tres cables, dos hacen el trabajo de alimentar de corriente eléctrica, y otro cable es conectado a la terminal

de la unidad de control a través de una resistencia de 330 ohms. Finalmente poder realizar la comunicación entre la tarjeta Arduino y la App, se configura el módulo Bluetooth HC-05, que es conformado por dos terminales, estas son conectadas al suministro de energía de Arduino y otras dos son conectadas al TX y RX para realizar la comunicación.

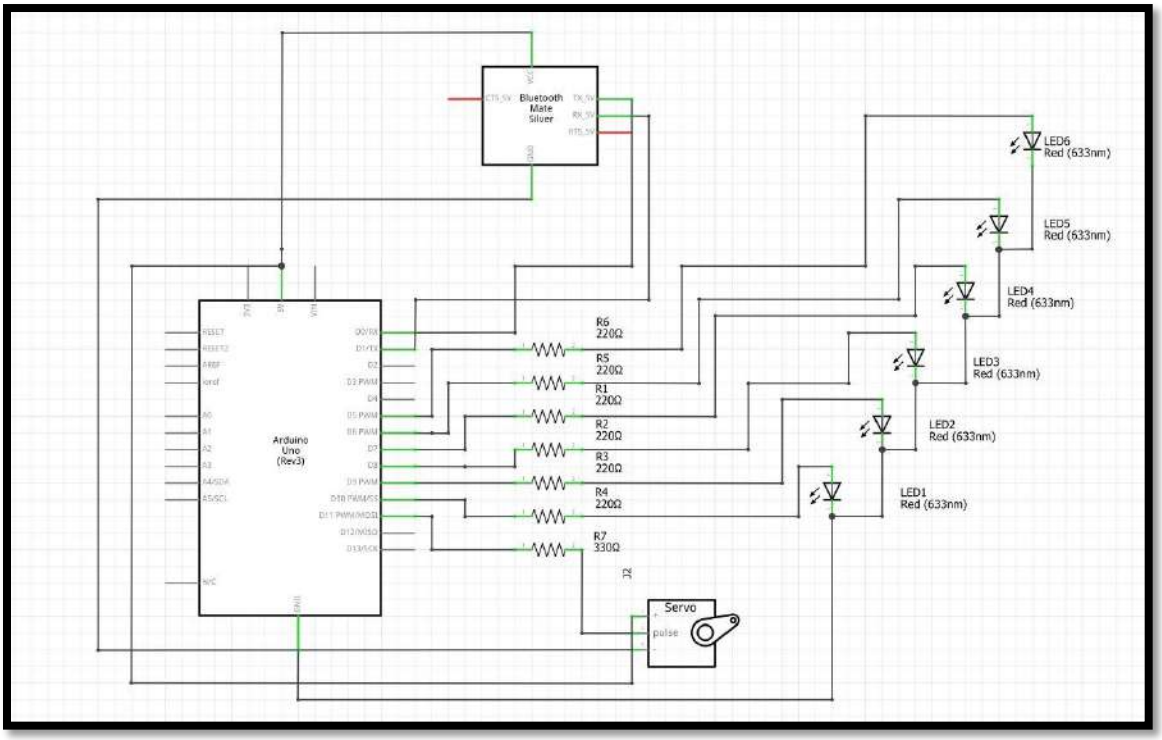


Figura 58. Diagrama de conexiones de todos los elementos que integran el sistema domótico.

El uso de diagramas de conexiones realista brinda al lector una idea más clara de cómo realizar el cableado correcto entre los dispositivos, y poder llegar a tener un buen funcionamiento de lo que se desee hacer. La Figura 59 muestra una vista de este tipo, sobre cómo conectar los componentes que integran este sistema domótico y la conexión de los LEDs con su resistencia correspondiente para su protección o realizar la conexión correcta entre el módulo Bluetooth HC-05 y Arduino, para que puedan comunicarse correctamente. Estos diagramas son muy fáciles de entender y muestran cómo llevar acabo dichas conexiones para prevenir y no causar algún corto circuito, y evitar dañar los componentes.

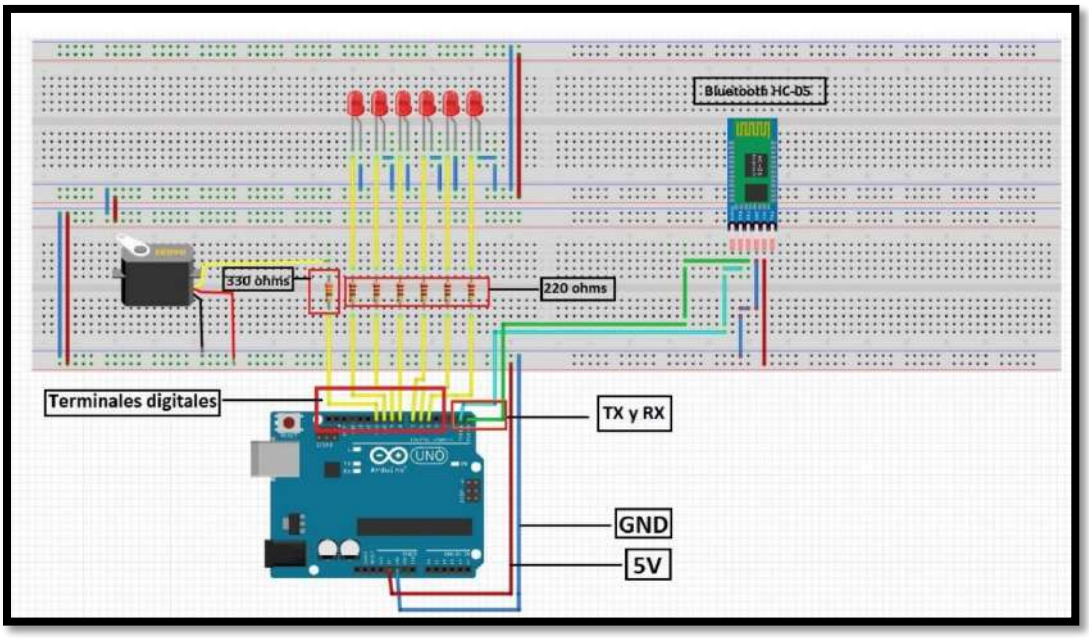


Figura 59. Configuración de conexiones de todos los elementos del Sistema propuesto.

La Figura 60 muestra la parte del circuito del sistema domótico donde se realizaron las conexiones de cada componente, en recuadros se presentan las características más importantes para que este sistema realice su funcionamiento correcto, como la alimentación de energía de la protoboard, el uso de las resistencia, además de la configuración del módulo Bluetooth HC-05.

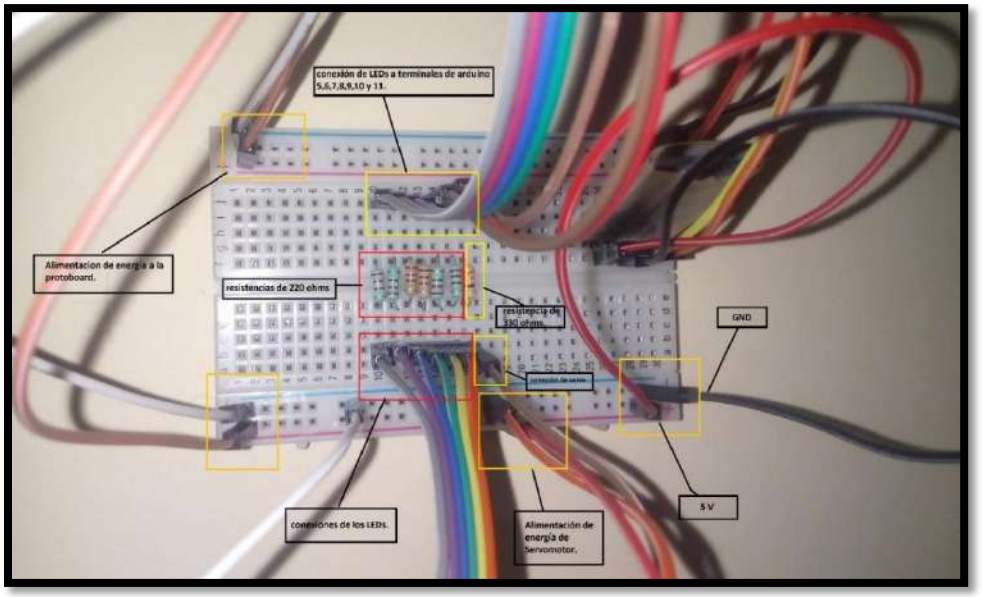


Figura 60. Montaje de forma real en protoboard de todos los componentes del sistema propuesto.

Para llevar a cabo la conexión entre los mecanismos montados en la protoboard y las terminales digitales de Arduino se utilizaron cables Jumpers (Macho-Macho) y (Macho-Hembra), además de un cable USB conectado a un cargador de 5v para poder alimentar la tarjeta, y esta pueda abastecer energía eléctrica a los dispositivos de la protoboard. La Figura 61 muestra cómo queda finalmente ensamblado el circuito del sistema domótico.

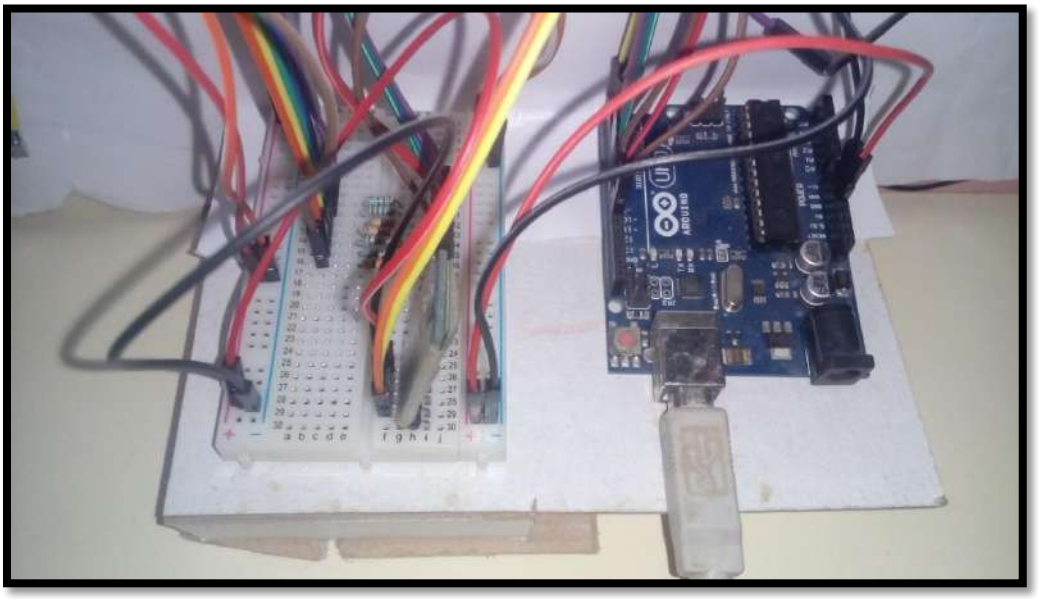


Figura 61. Cableado entre la protoboard y Arduino UNO.

Una vez realizado el cableado correcto entre el circuito de conexiones del sistema domótico y la tarjeta Arduino, se pasa a conectar cada terminal con una resistencia de 220 ohms a cada LED y una más para el terminal del servomotor. La Figura 62 muestra las conexiones de cada sección de la casa.

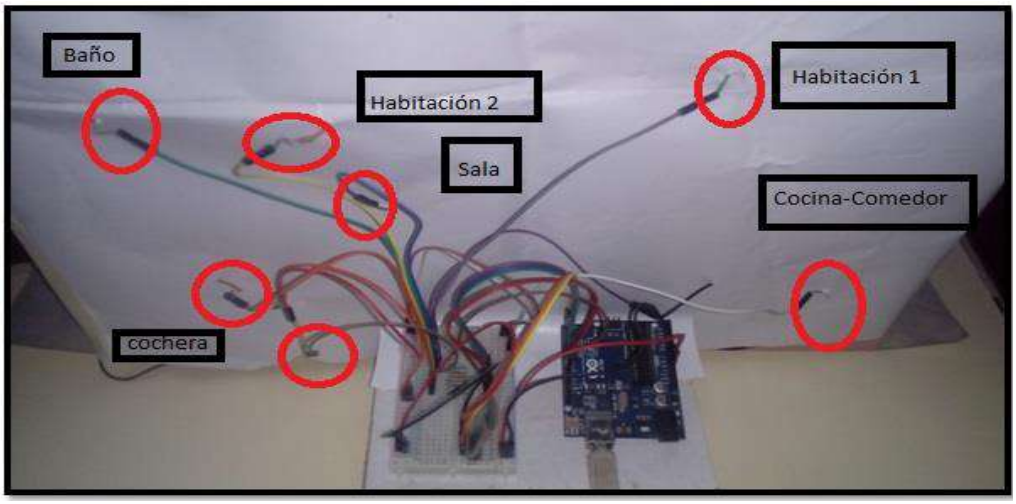


Figura 62. Conexión de LEDs en la parte trasera de la maqueta.

Lógica de control del microcontrolador

Los diagramas de flujo son usados para realizar la descripción de un proceso, algún sistema o simplemente un algoritmo informático. El uso de este tipo de diagramas sirve para el estudio, mejora y planificación de un programa o algoritmo, en otras palabras, estos diagramas son empleados para explicar la lógica que está detrás de un programa antes de que se desee iniciar la programación de dicho proceso [51].

Para la lógica del microcontrolador se desarrolló un diagrama de flujo que describe el funcionamiento del sistema domótico adaptado a una maqueta de una casa habitación, primero se declararon las terminales con un identificador led, led1, led2, led3, led4, led5 y servo todos ellos de tipo (int), asignado la terminal 5 a la 11 de Arduino, posteriormente en la función void setup () se inicializa cada terminal digital como salida. Es importante mencionar que esta función **void setup () (**ver Figura 63 **),** inicializa todos los datos y el puerto serial donde se establecerá la comunicación entre la App y la tarjeta Arduino, dicho lo anterior se realizará al arrancar la tarjeta Arduino. También podemos observar en la figura antes mencionada, la función **void loop ()** la cual contiene la comprobación del puerto y por consiguiente en una variable **aux** de tipo (char), esta almacenará el carácter que realizará el trabajo de ser comparado en las condiciones para lograr el encendido-apagado de lámparas y puerta de la cochera.

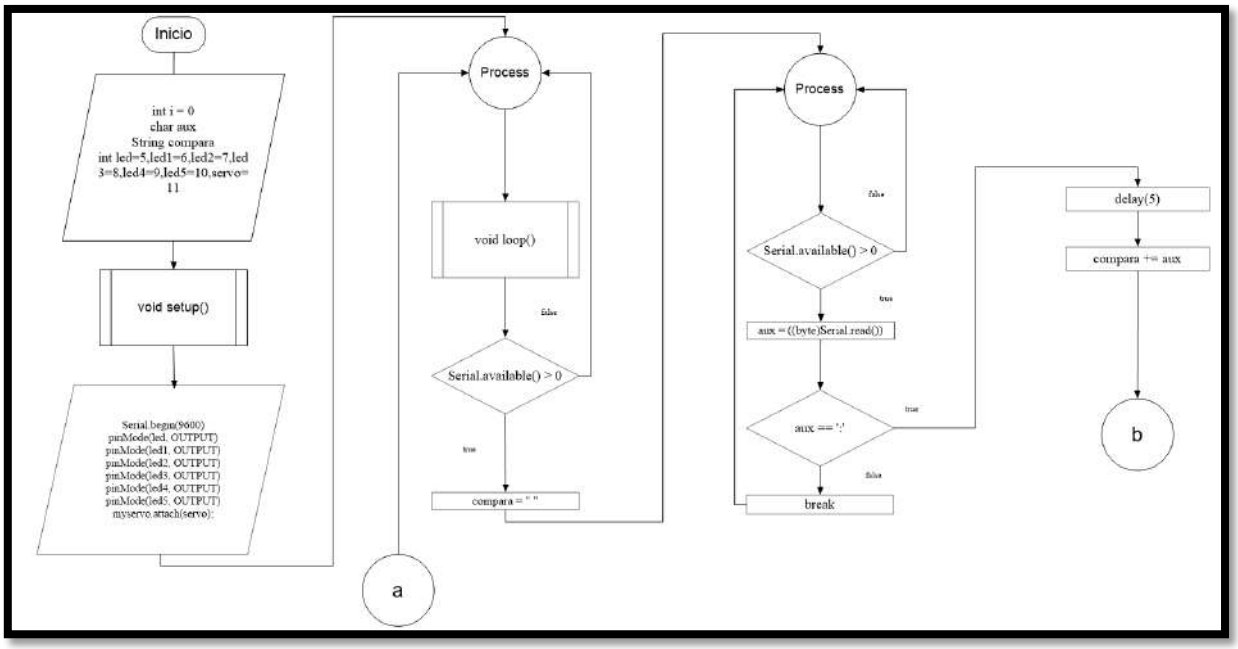


Figura 63. Diagrama de flujo del programa (sketch) de Arduino parte 1.

La Figura 64 muestra gran parte del funcionamiento del sistema domótico, ya que este contiene la lógica del encendido-apagado de las lámparas de cada parte de la casa. La variable "compara" contiene el carácter que va siendo comparado en cada caso.

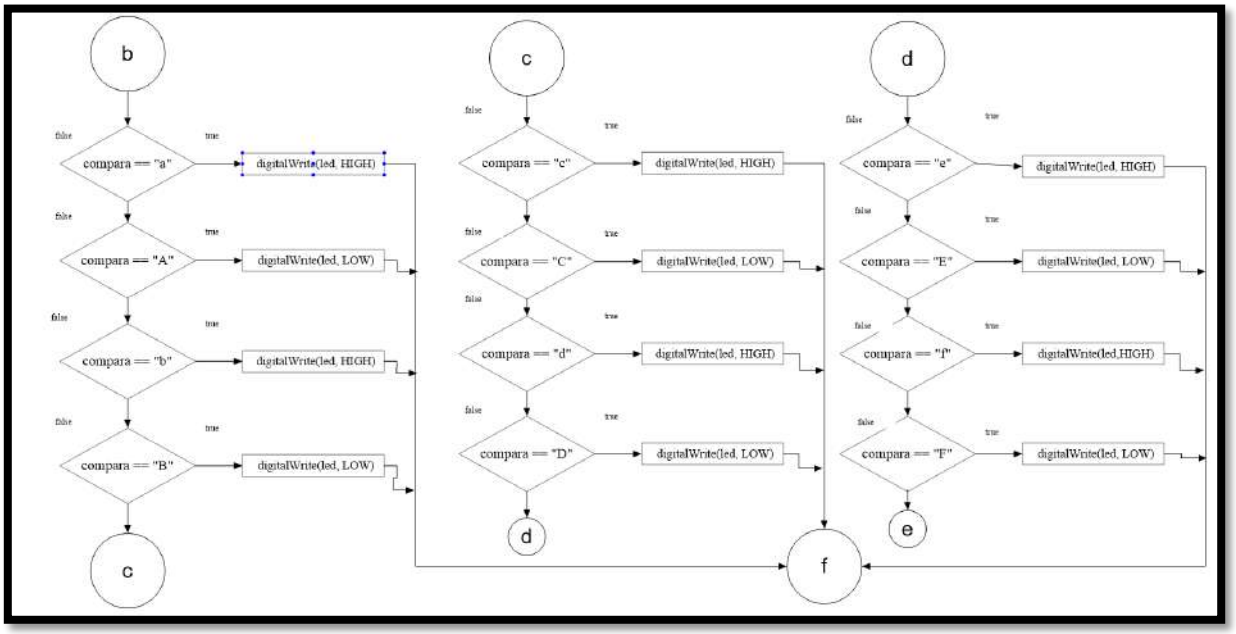


Figura 64. Diagrama de flujo del programa (sketch) de Arduino parte 2.

La parte de la lógica que controla el servomotor quien simula la puerta de la cochera se puede ver en la Figura 65, en este caso si en la condición la variable "**compara"** es igual al carácter "g" abre la pluma de la cochera, en caso contrario si es el carácter "G", cierra la pluma de la cochera, si ya no entra ningún carácter al puerto serial enviado por el control de la App, la variable "compara" se libera y termina el programa de no serlo así entra de regreso a la función void loop () para realizar todo el proceso nuevamente.

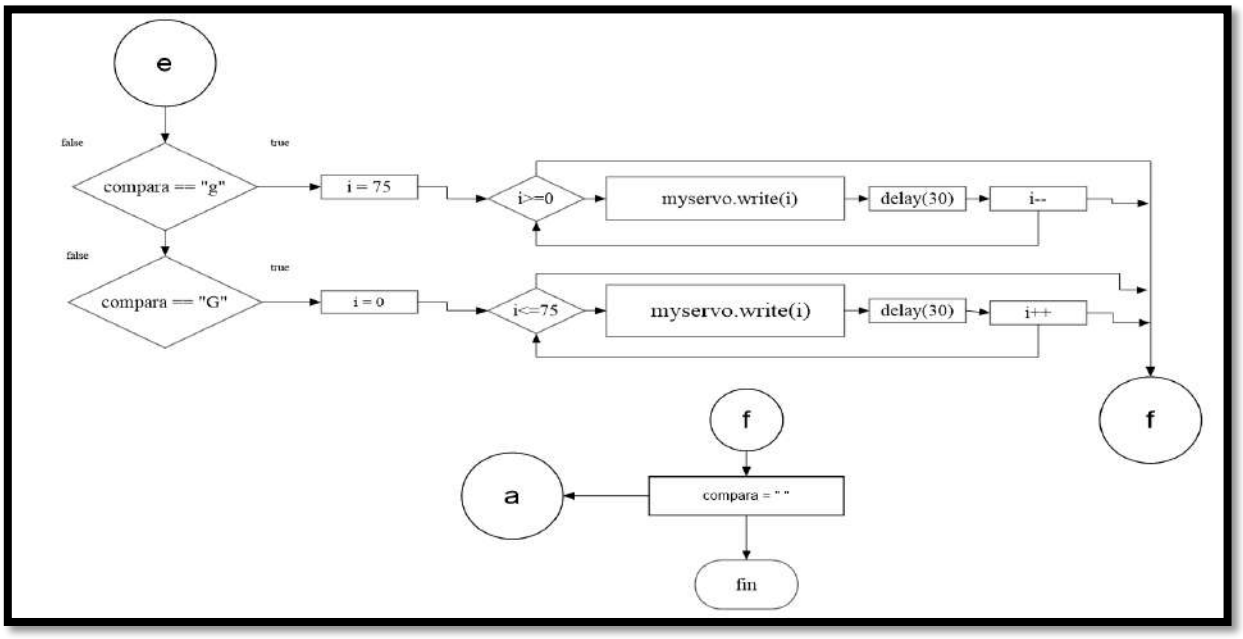


Figura 65. Diagrama de flujo del programa (sketch) de Arduino parte 3.

CAPÍTULO IV RESULTADOS Y CONCLUSIONES

PRUEBAS REALIZADAS AL SISTEMA DOMÓTICO

La Figura 66 muestra el sistema domótico de la casa controlada a través de la App desarrollada. Se puede observar del lado izquierdo la pantalla principal en la que se controla al sistema. Esta pantalla se encuentra conformada por botones que muestran su estado (activo/inactivo) mediante el color de cada botón; verde indica al usuario que ha activado las luces en la sección correspondiente de la casa, y en rojo se indica que no se encuentra activa. Esta aplicación es muy intuitiva y fácil de usar por personas de tercera edad y gente que presenta algún tipo de discapacidad motora, además es fácil de manejar para toda la gente en general.

Figura 66. Prueba del sistema domótico, encendido de luces y pluma de acceso de la cochera.

Se puede observar en la Figura 66 que se encuentran activadas las luces de la casa y la pluma de la cochera. Por otro lado, en la Figura 67 se muestra la vista de la App controlando el apagado de las luces y la pluma de la cochera. Una vez estando los botones en modo desactivado puede usarse el botón de color azul que esta situado en la esquina inferior izquierda para regresar a la vista anterior y poder salir de la aplicación.

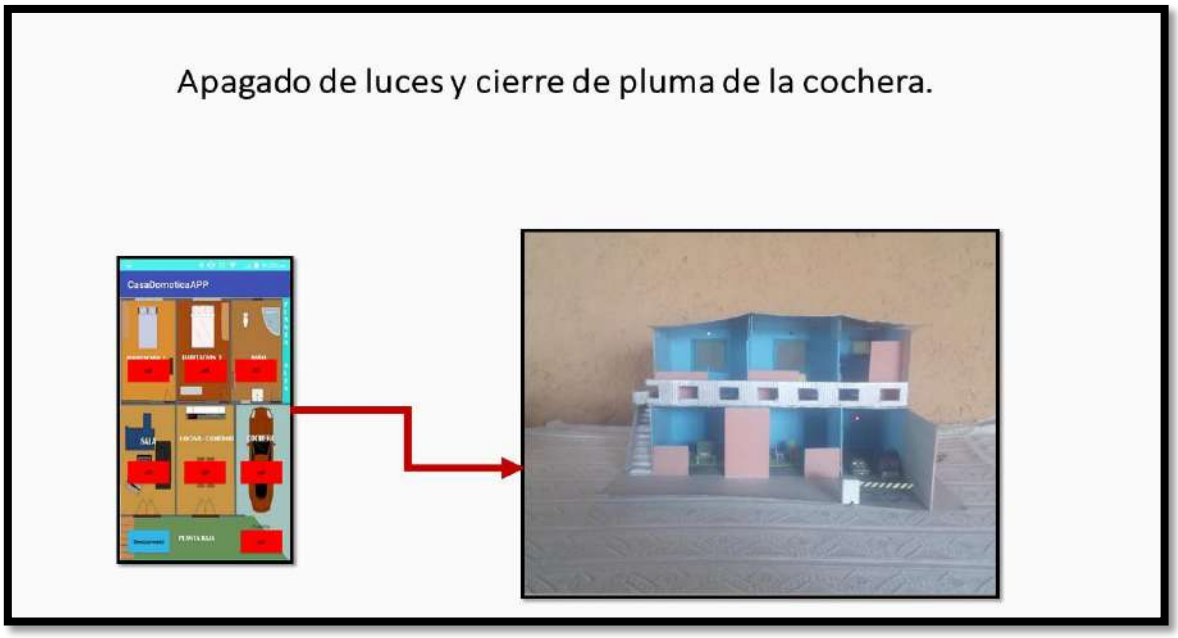


Figura 67. Prueba del sistema domótico, apagado de luces y pluma de acceso de la cochera.

CONCLUSIONES

En cuanto a la investigación abordada anteriormente, se retomaron temas relacionados con la domótica en general como; dispositivos móviles, protocolos de comunicación inalámbrica como lo es el Bluetooth con objetivo de crear este prototipo de sistema domótico apegado a una maqueta que simula una casa habitación, y así implementar una opción más en la innovación de las viviendas en México para brindar una mejor calidad de vida a las personas que presentan algún tipo de incapacidad motora y gente de la tercera edad.

Este prototipo de sistema de automatización para una casa habitación a escala (maqueta), maneja las herramientas adecuadas como; Arduino; el módulo Bluetooth HC-05; actuadores y la interfaz desarrollada para dispositivos móviles con SO Android que realiza el control de todo el sistema integrado a una casa habitación.

Al emplear las tecnologías de hardware y software que existen hoy en día que están al alcance de toda la gente, se puede lograr el desarrollo de la domótica en cualquier lugar o espacio del mundo real. Por ello estos sistemas pretenden que las personas logren una mejor calidad de vida y un ahorro a sus bolsillos, también tener en cuenta la preservación del medio ambiente.

La domótica va en crecimiento conforme al avance e innovación de la tecnología lo cual hace que este tipo de sistemas inteligentes puedan ser integrados a los hogares, ya sea para aumentar la seguridad o simplemente tener una mayor comodidad y cumplir esas necesidades que un hogar común no tiene.

TRABAJOS FUTUROS

Este trabajo presenta el encendido-apagado de luces y control de la puerta de acceso a la cochera, el cual es parte básica de un sistema domótico mucho más complejo, también proporciona las bases para el sistema domótico a futuro el cual contara con controles automáticos para la apertura-cierre de persianas y ventanas, control de temperatura, control de encendido-apagado de aparatos eléctricos y electrónicos del hogar, activación de cámaras de seguridad, monitoreo de niveles de gas y agua en tinacos; sensores que detecten humo de incendios y fugas de gas; sensores que controlen la climatización del hogar; sensores de movimiento para activar luces de los pasillos; también la integración de este sistema domótico con el Internet de la cosas para poder controlar cada parte que integra la vivienda desde cualquier parte del mundo.

Cabe destacar que este proyecto es una parte básica de un sistema domótico, pero sirve de base como sustento teórico y estudio del estado del arte de la domótica, todo ello para la continuación en la integración e implementación de este sistema domótico más complejo.

REFERENCIAS

[1] U. de S. G. F. de I. Auburn, "Controlling Computers via X10," 2018. [Online]. Available: http://www.eng.auburn.edu/~doug/x10.html. [Accessed: 07-May-2018].

[2] H. M. Domínguez and F. Sáez, *Domótica: Un enfoque sociotécnico*, Primera ed. Madrid España: Universida Politécnica de Madrid, 2006.

[3] J. M. Huidobro Moya and R. J. Millán Tejedor, *Manual de Domótica*. Creaciones Copyright SL, 2010.

[4] Fundación de la Energia de la comunidad de Madrid, "La Domótica como solución de Futuro," España, 2007.

[5] VENTO Domótica del Hogar, "Partes de un sistema domótico,Sensores, controladores y actuadores." [Online]. Available: http://sistemasdomoticos.com/partes-de-un-sistema-domotico/. [Accessed: 07-Apr-2018].

[6] L. G. C. Ramirez, G. S. A. Jiménez, and J. M. Carreño, *Sensores y Actuadores:* Grupo Editorial Patria, 2014.

[7] R. Hernández Balibrea, "Tecnología domótica para el control de una vivienda.," Universidad Politécnica de Cartagena, 2012.

[8] GRUPOTECMARED, "Domótica . CASADOMO." [Online]. Available: https://www.casadomo.com/domotica. [Accessed: 08-Aug-2018].

[9] Grupo Intercom, "Ventajas e incovenientes de la domótica," 1995. [Online]. Available: https://www.domotica365.com/articulos/ventajas-e-inconvenientes-de-la-domotica. [Accessed: 05-Apr-2018].

[10] D. I. Constantino Leon, "DOMÓTICA E INMÓTICA: VIVIENDAS Y EDIFICIOS INTELIGENTES," Universidad Veracruzana, 2011.

[11] Arduino, "¿Qué es Arduino?" [Online]. Available: http://arduino.cl/que-es-arduino/. [Accessed: 05-Apr-2018].

[12] Ó. Torrente Artero, *ARDUINO Curso práctico de formación*, Primera Ed. México: Alfaomega, 2013.

[13] hacedores Maker community, "¿Cuántos tipos de Arduino hay?" [Online]. Available: hacedores.com/cuantos-tipos-diferentes-de-arduino-hay/. [Accessed: 07-Apr-2018].

[14] ARDUINO, "Arduino Uno Rev3." [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Accessed: 07-Apr-2018].

[15] G. de Canarias, “Características técnicas del ARDUINO UNO.” [Online]. Available: www.gobiernodecanarias.org/principal/. [Accessed: 07-Apr-2018].

[16] jecresprom, “esquemas eléctricos | Aprendiendo Arduino,” 2014. [Online]. Available: https://aprendiendoarduino.wordpress.com/tag/esquemas-electricos/. [Accessed: 08-Aug-2018].

[17] B. Evans, *Beginning Arduino Programming*. Apress, 2011.

[18] Arduino, “Arduino - Libraries.” [Online]. Available: https://www.arduino.cc/en/Reference/Libraries. [Accessed: 07-Apr-2018].

[19] Arduino, “Arduino - Servo.” [Online]. Available: https://www.arduino.cc/en/Reference/Servo. [Accessed: 07-Apr-2018].

[20] Geekfactory, “Shields Arduino- Comprar Shields para Arduino en méxico.” [Online]. Available: https://www.geekfactory.mx/categoria-de-producto/shields-arduino/. [Accessed: 08-Apr-2018].

[21] SparkFun, “Arduino Shields - learn.sparkfun.com,” 2003. [Online]. Available: https://www.sparkfun.com/about_sparkfun?_ga=2.28164415.2080440731.1523245481-494258608.1519186823. [Accessed: 08-Apr-2017].

[22] F. REYES CORTÉS, *Robótica CONTROL DE ROBOTS MANIPULADORES*, PRIMERA ED. 2011.

[23] Electronilab, “Micro Servo 9g SG90 TowerPro-ELECTRONILAB.” [Online]. Available: https://electronilab.co/tienda/micro-servo-9g-towerpro/. [Accessed: 08-Apr-2018].

[24] INTPLUS, “Trabajar con Servos,” 2002. [Online]. Available: http://www.superrobotica.com/servosrc.htm. [Accessed: 08-Apr-2018].

[25] panamahitek, “panamahitek.com/que-es-y-como-funciona-un-servomotor.” [Online]. Available: http://panamahitek.com/que-es-y-como-funciona-un-servomotor/. [Accessed: 08-Aug-2018].

[26] INTPLUS, “Trabajar con Servos.” [Online]. Available: http://www.superrobotica.com/servosrc.htm. [Accessed: 08-Aug-2018].

[27] Universidad Técnica Federico Santa María, “Servomotores,” 2003. [Online]. Available: http://www2.elo.utfsm.cl/~mineducagv/docs/ListaDetalladadeModulos/servos.pdf.

[Accessed: 08-Apr-2018].

[28] E. C. Baig, *Macs Para Dummies*, 10th ed. Wiley, 2009.

[29] FayerWayer, "La historia del nacimiento del Bluetooth." [Online]. Available: https://www.fayerwayer.com/2011/09/la-historia-del-nacimiento-de-bluetooth/. [Accessed: 16-Jul-2018].

[30] T-Bem, "Módulo Bluetooth serial HC-05." [Online]. Available: http://teslabem.com/modulo-bluetooth-serial-hc-05.html. [Accessed: 08-Apr-2018].

[31] Electrónicos CALDAS, "Módulo Bluetooth HC-05." [Online]. Available: http://www.electronicoscaldas.com/modulos-rf/452-modulo-bluetooth-hc-05.html. [Accessed: 08-Apr-2018].

[32] M. del C. García, "Configurar módulos Bluetooth HC-05 y HC-06 mediante comandos AT | Mi Arduino UNO tiene un BLOG," 2016. [Online]. Available: https://miarduinounotieneunblog.blogspot.mx/2016/12/configurar-modulos-bluetooth-hc-05-y-hc.html. [Accessed: 08-Apr-2018].

[33] HC Serial Bluetooth Products, "Módulo Bluetooth Módulo Inalámbrico serie Bluetooth Módulo inalámbrico HC MóduloBluetooth Guangzhou Huicheng Information Technolology Cp.,Ltd." [Online]. Available: http://www.wavesen.com/. [Accessed: 08-Apr-2018].

[34] Wolters Kluer, "Bluetooth Module HC-05 & HC-06 Master - Slave mode |Trade Me." [Online]. Available: https://www.trademe.co.nz/electronics-photography/other-electronics/electronic-components/other/listing-1727630429.htm. [Accessed: 08-Aug-2018].

[35] D. Robledo, *Desarrollo de aplicaciones para Android I*. Ministerio de Educación de España, 2017.

[36] R. Moya, R. Invarato, and D. Cortes, "ART vs DALVIK, arquitectura Android," 2018. [Online]. Available: https://jarroba.com/. [Accessed: 08-Apr-2018].

[37] J. Tómas Gironés, *El gran libro de Android*, Segunda Ed. Barcelon, España: Alfaomega, 2012.

[38] J. D. L. Castillo and J. D. L. Castillo, *Android : aprende desde cero a crear aplicaciones*. RC Libros, 2014.

[39] Basterra - Bertea - Borello - Castillo - Venturi, "Características--Android OS 0.1

documentation," 2012. [Online]. Available: http://androidos.readthedocs.io/en/latest/data/caracteristicas/. [Accessed: 08-Aug-2018].

[40] Ó. ÁVILA MEJÍA, "Android," México, p. 9, 11-Jul-2011.

[41] Google, "Arquitectura de la plataforma," 2018. [Online]. Available: https://developer.android.com/guide/platform/index.html?hl=es-419#api-framework. [Accessed: 04-Apr-2018].

[42] Google, "Arquitectura de la plataforma | Android Developers." [Online]. Available: https://developer.android.com/guide/platform/?hl=es-419. [Accessed: 08-Aug-2018].

[43] Google, "<uses.sdk> | Android." [Online]. Available: https://developer.android.com/guide/topics/manifest/uses-sdk-element.html?hl=es-419. [Accessed: 08-Apr-2018].

[44] Google, "Paneles de control | Android Developers," 2018. [Online]. Available: https://developer.android.com/about/dashboards/index.html?hl=es-419. [Accessed: 08-Apr-2018].

[45] Google, "Cómo descargar Android Studio y SDK Tools| Android Studio," 2018. [Online]. Available: https://developer.android.com/studio/index.html?hl=es-419. [Accessed: 08-Apr-2018].

[46] Google, "Download Android Studio and SDK tools | Android Developers." [Online]. Available: https://developer.android.com/studio/?hl=es-419. [Accessed: 08-Aug-2018].

[47] Google, "Conoce Android Studio|Android Studio." [Online]. Available: https://developer.android.com/studio/intro/index.html. [Accessed: 08-Apr-2018].

[48] CICE LA ESCUELA PROFESIONAL DE NUEVAS TECNOLOGIAS, "Top 5 plataformas de desarrollo iOS y Android - CICE." [Online]. Available: https://www.cice.es/noticia/top-5-plataformas-desarrollo-ios-android/. [Accessed: 08-Apr-2018].

[49] Google, "Diseños | Android Developers." [Online]. Available: https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419. [Accessed: 29-May-2018].

[50] Lucidchart, "Diagrama de clases | LUcidchart." [Online]. Available:

https://www.lucidchart.com/pages/es/diagrama-de-clase. [Accessed: 29-May-2018].

[51] Lucidchart, “Qué es un diagrama de flujo.” [Online]. Available: https://www.lucidchart.com/pages/es. [Accessed: 29-May-2018].