ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DOMÓTICO BASADO EN LA PLATAFORMA ARDUINO GESTIONADO A TRAVÉS DEL INTERNET

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

DAVID SANTIAGO VINUEZA CHIMARRO

david.vinueza@epn.edu.ec

DIRECTOR: WILLAMS FERNANDO FLORES CIFUENTES, MSc.

fernando.flores@epn.edu.ec

Quito, febrero 2019

AVAL

Certifico que el presente trabajo fue desarrollado por David Santiago Vinueza Chimarro, bajo mi supervisión.

Ing. WILLAMS FERNANDO FLORES CIFUENTES, MSc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, David Santiago Vinueza Chimarro, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

DAVID SANTIAGO VINUEZA CHIMARRO

DEDICATORIA

Este trabajo de titulación está dedicado a mi madre Jacqueline, a mi padre Guido y a mi hermana Marjoury, por estar conmigo y apoyarme siempre.

David

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE SEGMENTOS DE CÓDIGO

RESUMEN

El presente proyecto de titulación pretende desarrollar un prototipo de sistema domótico basado en la plataforma Arduino que genere alertas de seguridad y posibilite controlar el sistema de iluminación en una vivienda a través del Internet y Bluetooth, que permita mejorar la calidad de vida de sus habitantes a través de la tecnología.

En el primer capítulo, se presenta una descripción teórica de los elementos de una solución domótica, plataforma Arduino y sistema operativo Android.

En el segundo capítulo, se determinan los requisitos del prototipo de sistema domótico. Se detalla el proceso de diseño e implementación de los componentes que conforman el prototipo, incluyendo la aplicación móvil para su gestión.

El tercer capítulo presenta los resultados de las pruebas de funcionamiento para verificar el cumplimiento del alcance del proyecto de titulación.

El cuarto capítulo muestra las conclusiones y recomendaciones en base al análisis de los resultados obtenidos al evaluar el prototipo de sistema domótico.

Se incluye una sección de Anexos en la que se muestran el código de la aplicación móvil, el programa del controlador Arduino y las características de los dispositivos electrónicos empleados en el prototipo.

PALABRAS CLAVE: domótica, Arduino, Android, circuitos electrónicos, programación.

ABSTRACT

The present titling project aims to develop an Arduino based home automation system prototype that generates security alerts and allows to control the lighting system through the Internet and Bluetooth, which allows to improve the quality of life of its inhabitants through of technology.

In the first chapter, a theoretical description of the elements of a home automation solution, Arduino platform and Android operating system is presented.

In the second chapter, the requirements of the home automation system prototype are determined. The design and implementation process of the components that make up the prototype is detailed, including the mobile application for its management.

The third chapter presents the results of the functional tests to verify compliance with the scope of the titling project.

The fourth chapter shows the conclusions and recommendations based on the analysis of the results obtained when evaluating the home automation system prototype.

A section of Annexes is included in which the code of the mobile application, the program of the Arduino controller and the characteristics of the electronic devices used in the prototype are shown.

KEYWORDS: home automation, Arduino, Android, electronic circuits, programming.

1. INTRODUCCIÓN

En los últimos años, el desarrollo de las Tecnologías de la Información y Comunicación (TIC) en el Ecuador y en el mundo ha sido significativo. Estas tecnologías presentes hasta en las cosas más cotidianas, han modificado de forma irreversible la forma de comunicarnos e interactuar con el entorno en que vivimos. Una de las áreas que ha experimentado grandes cambios debido a estos avances es el hogar.

El origen de lo que hoy se conoce como vivienda domótica o vivienda inteligente se produce al inicio de la década de los setenta del siglo pasado. Sin embargo, no es hasta que se introducen las telecomunicaciones y la informática en las edificaciones, para posibilitar la gestión centralizada y autónoma de sus instalaciones, que se las puede llamar propiamente viviendas inteligentes [1].

El acceso a estas tecnologías posibilita la implementación de sistemas domóticos para mejorar el nivel de confort, brindar nuevos servicios de seguridad, ahorro energético, entretenimiento y comunicaciones en las edificaciones; a pesar de ello, el despliegue de estos sistemas en las viviendas es reducido.

Es así, que resulta relevante el desarrollo de un Prototipo de Sistema Domótico basado en la Plataforma Arduino para ofrecer al usuario mayor confort, seguridad, ahorro energético y económico a través de la gestión cómoda y centralizada de los diferentes equipos e instalaciones de la vivienda, utilizando el Internet o Bluetooth, para mejor calidad de vida de las personas que habitan en ella a través de la tecnología.

1.1. Objetivos

El objetivo general de este Proyecto Integrador es:

- Desarrollar un prototipo de sistema domótico basado en la plataforma Arduino que genere alertas de seguridad y posibilite gestionar los equipos e instalaciones utilizando el Internet y Bluetooth que mejore la calidad de vida de las personas.

Los objetivos específicos de este Proyecto Integrador son:

- Analizar los principales elementos de una solución domótica.
- Diseñar los componentes que conforman el sistema domótico para generar alertas de seguridad y gestionar equipos e instalaciones.
- Implementar los componentes que conformarán el prototipo.

1.2. Alcance

Dentro del alcance de la implementación del sistema domótico se distingue lo siguiente:

- Detección de apertura y cierre de puertas y ventanas.
- Detección de movimiento.
- Envío de notificaciones al usuario.
- Simulación de presencia en el hogar.
- Botón de pánico.
- Obtención de imágenes de ciertas zonas de la vivienda a través de una cámara, activada por la detección de movimiento en dicha zona.
- Control del encendido y apagado de luces en las diferentes áreas de la vivienda.
- Encendido y apagado de la radio.
- Medición de temperatura.
- Aplicación de Android para gestionar el sistema domótico.
- Comunicación entre el sistema domótico y la aplicación a través de Internet y Bluetooth.

Se utilizará una placa Arduino como controlador del sistema domótico. La placa Arduino se encargará de procesar la información proveniente de los sensores para enviar órdenes a los actuadores o notificar a la aplicación móvil el estado actual del sistema, como se observa en la Figura 1.1.

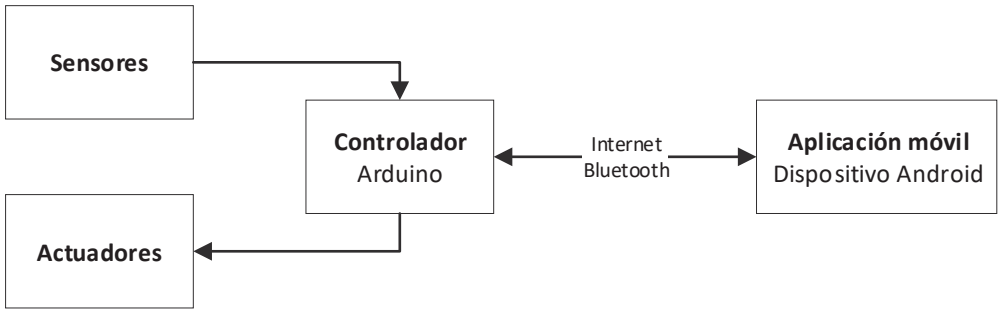

Figura 1.1. Diagrama de bloques del sistema domótico

Los equipos e instalaciones de la vivienda serán controlados a través de un *smartphone*. El prototipo permitirá encender o apagar las luminarias de la vivienda. Adicionalmente, se

podrá encender o apagar la radio, que además de evitar que esta tarea se realice manualmente, será utilizado al simular presencia en la residencia.

Otra de las funciones del sistema es la detección de movimiento inusual y envío de notificaciones a la aplicación. Esto se realizará en base al monitoreo del estado de una puerta o ventana, ya sea abierta o cerrada, y en base a detectores de movimiento en la vivienda. Al detectarse movimiento en la vivienda, una cámara presente en el lugar tomará una fotografía, a la que el usuario podrá acceder a través de su teléfono móvil mediante un correo electrónico.

Se brindará la capacidad al sistema de simular presencia en el hogar mediante la activación de distintos elementos cuando la vivienda no esté ocupada: encendido de luces en una o varias estancias y encendido de la radio.

El sistema contará con un botón de pánico, el cual será activado desde la aplicación de Android o a través de un pulsador y permitirá generar una alerta sonora en el hogar.

Con el propósito que los usuarios conozcan la temperatura al interior de su residencia, el sistema contará con un sensor de temperatura.

La comunicación entre el controlador del sistema domótico, que gestiona los diferentes sensores y actuadores, y la aplicación móvil se realizará a través de Internet o Bluetooth. Para que el usuario envíe órdenes al controlador y reciba notificaciones del mismo, es necesario, que la vivienda cuente con una pasarela residencial -que combina las funciones de módem, *switch*, *router*, cortafuegos y permite conectar las infraestructuras de telecomunicaciones que coexisten en la vivienda [2, pp. 157-159]- a la cual se conectará el controlador del prototipo y posibilitará al usuario gestionar los diferentes elementos del sistema domótico desde dentro y fuera de la vivienda a través del Internet. Adicionalmente, se podrá utilizar Bluetooth para controlar las funciones del prototipo.

Se incluirá como parte del sistema una aplicación para Android, que permitirá a un usuario gestionar el prototipo.

El prototipo de sistema domótico estará compuesto por los siguientes elementos:

- Plataforma Arduino.
- Aplicación para el sistema operativo Android.
- Módulo Bluetooth y placa Ethernet *shield* para la comunicación con la aplicación de Android.

- Módulo relé para el control a distancia de luces y de la radio.
- Sensor de movimiento PIR.
- Sensor de temperatura
- Contacto magnético para detección de apertura o cierre de puertas o ventanas.
- Cámara IP.
- Pulsador.
- Sirena para alerta sonora.

1.3. Marco Teórico

1.3.1. Domótica

En la actualidad se pueden encontrar diferentes definiciones que hacen referencia al concepto de domótica o vivienda domótica. Cada definición, por lo general, responde al enfoque del autor para potenciar algún aspecto relacionado a la misma, ya sea, tecnología utilizada, funcionalidad del sistema, beneficio para el usuario, entre otros [3, p. 7].

El origen de la domótica se produjo como resultado de investigaciones centradas en la tecnología X-10, que permitieron la aparición de los primeros dispositivos de automatización para edificaciones. Los primeros sistemas comerciales poseían limitadas funciones, principalmente relacionadas a la regulación de la temperatura ambiente en edificios de oficinas y fueron instalados en su mayoría en Estados Unidos. Posteriormente, estos sistemas fueron aplicados también a las viviendas y otras edificaciones, lo que dio origen a la vivienda domótica [2, p. 15].

Existen pocas diferencias entre una vivienda con equipamiento domótico y una vivienda tradicional. Ambas poseen los mismos materiales de construcción, forma, equipamiento similar, es decir, se trata de la misma vivienda. La diferencia radica en la adición de una mínima tecnología que posibilite gestionar de forma centralizada y con comodidad los diferentes equipos e instalaciones que forman parte de la misma [3, p. 7].

Hasta hace poco, en una vivienda únicamente se podía instalar pequeñas automatizaciones independientes entre sí para controlar distintos servicios y automatizar determinadas tareas en las viviendas de forma individual. En el mercado existen diversos dispositivos que funcionan de forma autónoma y realizan tareas como el encendido de luminarias al detectar presencia, regulación del nivel de luminosidad, activación programada de ciertos dispositivos, entre otros. Por lo tanto, no se debe confundir el

concepto de domótica con el de automatismo. La domótica no hace referencia a productos o servicios aislados, sino a la integración de aquellos aparatos automáticos (automatismos) para conseguir una mejor gestión global del sistema [4, p. 24].

Es así, que se puede definir a un sistema domótico como la tecnología que permite integrar y controlar los diferentes sistemas independientes de una vivienda desde una única ubicación y con una simple actuación para aumentar la comodidad, la seguridad, el ahorro de energía, mejorar las comunicaciones y de esta forma mejorar la calidad de vida de los habitantes [4, p. 23].

1.3.1.1. Aplicaciones de la domótica

Las viviendas inteligentes proporcionan beneficios para el usuario, que resultan de difícil acceso en edificaciones tradicionales. Las principales aplicaciones de la domótica en la vivienda se agrupan dentro de las siguientes categorías o áreas funcionales: confort, seguridad, ahorro energético y comunicaciones.

Se puede afirmar que un sistema domótico es aquel que incide en estas cuatro áreas y que permite interacción entre las mismas, sin embargo, en muchas ocasiones el usuario final sólo necesita cubrir alguna de estas áreas en concreto.

Seguridad

La seguridad ciudadana no ha sido un tema tratado con la prioridad necesaria al discutir los problemas de desarrollo en el Ecuador [5]. Las personas han comprobado que las rejas ya no son suficientes, por lo que la utilización de la tecnología para brindar seguridad en los hogares se vuelve una necesidad.

Es así, que las aplicaciones enfocadas a brindar una mayor seguridad en la vivienda son las que más han contribuido a introducir sistemas domóticos en los hogares [6, p. 198].

Las principales aplicaciones de seguridad de la domótica están enfocadas a la protección tanto de los habitantes como de los bienes en la vivienda.

J. M. Huidobro y R. J. Millán Tejedor [7, p. 285] concluyen que la implementación de un sistema que incremente la seguridad en los recintos contribuye no solo en la protección de los bienes particulares, sino también en la propia protección personal. Indican que al contar con un sistema que permita detectar intrusiones en una vivienda, en caso de existir alguna intromisión, se generaría una notificación al usuario ya sea que este se encuentre dentro o fuera de la misma. De este modo, se puede dar aviso a la policía evitando que la intromisión termine en robo o en una agresión física en caso de que alguien se encuentre presente.

Las principales aplicaciones de seguridad de un sistema domótico se encuentran orientadas a la utilización de alarmas anti-intrusión, alarmas técnicas y simulación de presencia.

En la detección de intrusos, existen principalmente dos variantes: detección perimetral y detección interna. Dentro de la detección perimetral se distinguen aplicaciones como la detección de apertura de puertas o ventanas, detección de rotura de cristales, detección de personas en las inmediaciones del hogar, entre otros, que hacen uso de sensores magnéticos, sensores de rotura de cristal o barreras infrarrojas, para activar alarmas o notificar al usuario.

En lo que respecta a detección interna de intrusos, se incluyen aplicaciones dirigidas a la detección de personas dentro de nuestra vivienda mediante el uso de sensores de movimiento u otros dispositivos.

Al hablar de alarmas técnicas, se hace referencia a la utilización de sensores de humo, gas, agua, óxido de carbono, entre otros, que al detectar alguna anomalía, por ejemplo una fuga de gas, activan una alarma y pueden incluso cortar los suministros que se han visto afectados haciendo uso de electroválvulas u otros mecanismos [8].

Como una medida extra de seguridad, se puede emplear la función de simulación de presencia, es decir, simular que existe alguien dentro del hogar mediante la activación de los diferentes equipos domésticos u otras instalaciones que forman parte de la vivienda, ya sea, encendiendo las luces u otro aparato doméstico, abriendo las persianas, etc., de tal manera que el hogar parezca estar habitado.

Gestión de energía [7]

El creciente interés de la sociedad en el cuidado del medio ambiente facilita la introducción de soluciones que permitan disminuir el consumo de energía en el hogar.

La domótica posibilita a través de la automatización de funciones y la incorporación de aparatos domésticos inteligentes, una optimización del consumo de energía en el hogar, utilizando la cantidad de energía adecuada para satisfacer las necesidades del hogar a un costo mínimo [4, p. 25].

Las principales aplicaciones dirigidas al ahorro de energía están enfocadas al control de la temperatura e iluminación y a la gestión del funcionamiento de los diferentes electrodomésticos. Algunas de las aplicaciones de la domótica dirigidas al ahorro energético son:

- Desactivación de la iluminación en caso de olvidar apagar las luminarias o al no detectar presencia.
- La programación de la climatización en la vivienda ya sea el aire acondicionado o la calefacción, pudiendo incluso, desactivar el sistema de climatización, en caso de que las puertas o ventanas se encuentren abiertas.
- Programación de los diferentes dispositivos y electrodomésticos de acuerdo a las necesidades y en determinados horarios, evitando, por ejemplo, que funcionen todos al mismo tiempo y se supere una determinada potencia.

Comunicaciones [9]

La gestión de las comunicaciones se enfoca principalmente en la conexión del sistema de control de la vivienda con el exterior.

Para mantener un buen control de las diferentes funciones de la vivienda, es necesario contar con una comunicación ágil entre la solución domótica y el usuario, que nos permita controlar el sistema, ya sea desde dentro o fuera de la edificación. Dispositivos como un mando a distancia, un computador o un teléfono móvil nos brindan dicha versatilidad.

El contar con un sistema domótico que permita controlar los equipos e instalaciones de la vivienda a distancia, desde cualquier lugar, supone una garantía de seguridad, ahorro energético y confort en todo momento [10, p. 4].

La gestión de las comunicaciones en el hogar se basa en las siguientes tareas:

- Control a distancia de los diferentes equipos domésticos, ya sea, desde una computadora o un teléfono móvil, utilizando por ejemplo el Internet.
- Transmisión de alarmas, que se refiere al envío de información relacionada a anomalías en la vivienda, como accesos indeseados o fugas de gas, a una central de alarma, un teléfono móvil o cualquier otro sistema informático.

Confort

Los servicios de la domótica orientados al confort posibilitan una mejor calidad de vida a través de la reducción del trabajo doméstico y el aumento del bienestar y seguridad de los habitantes.

La introducción de la tecnología en aparatos domésticos como lavadoras, hornos, microondas, refrigeradores, etc., permite obtener una mejora considerable en la comodidad en el hogar. La utilización de electrodomésticos que además de realizar las funciones

tradicionales de cada uno incluyen un sistema de procesamiento, control y automatización, permitirá la programación de estos aparatos para que actúen de acuerdo con las especificaciones del usuario y permitirá que dichos electrodomésticos puedan ser controlados de forma remota [7].

A través de una pasarela residencial, se tiene acceso al sistema de control de la vivienda desde dentro o fuera de la misma, lo que posibilita que las instalaciones del hogar puedan ser gestionadas desde un teléfono móvil u otros dispositivos utilizando el Internet.

A continuación, se muestran algunas de las aplicaciones de la domótica relacionadas a la mejora del confort [11, p. 23]:

- **Control de las instalaciones de la vivienda:** Control local y remoto de la iluminación, activación de la iluminación por detección de presencia, entre otros.
- **Control de electrodomésticos:** Encendido y apagado de electrodomésticos de forma remota, programación de la activación de los aparatos domésticos en determinado momento.
- **Control de alarmas de seguridad:** Monitoreo de las incidencias relativas a intrusiones en la vivienda, activación de la función de simulación de presencia de forma remota o local, etc.
- **Generación de alertas:** Generación de notificaciones al usuario sin importar si este se encuentra dentro o fuera de la vivienda.

1.3.1.2. Tipos de arquitectura

El controlador del sistema domótico es aquel capaz de recopilar información proveniente de unas entradas -mandos o sensores-, procesarla y enviar órdenes a las salidas o actuadores [10, p. 6].

En base al lugar donde reside la inteligencia encargada de procesar la información y tomar las decisiones, es decir, el controlador, la arquitectura de un sistema domótico puede ser centralizada, descentralizada y distribuida.

Arquitectura centralizada [4] [10]

Se pueden catalogar dentro de este grupo los sistemas en los que existe un solo controlador encargado de la gestión de las instalaciones de la vivienda, al cual se conectan los diferentes sensores y actuadores, como se observa en la Figura 1.2.

En los sistemas de arquitectura centralizada, los elementos encargados de supervisar y controlar (sensores, válvulas, etc.) se encuentran conectados en un solo punto, generalmente la unidad de control central, que contiene la inteligencia del sistema. Por lo tanto, en estos sistemas, la comunicación entre los diferentes elementos pasa por la unidad central o controlador.

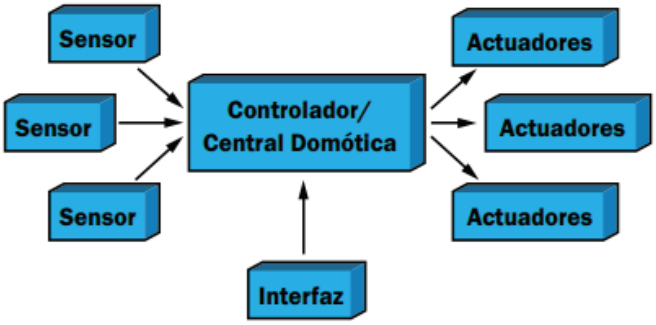

Figura 1.2. Sistema domótico de arquitectura centralizada [8, p. 31]

Esta arquitectura es la más económica y sencilla de las configuraciones. Aunque en caso de existir un fallo en el controlador central y este quede fuera de servicio, todo el sistema queda inutilizable.

Arquitectura descentralizada [10] [8]

En esta arquitectura cada componente puede actuar por sí mismo, y en caso de necesitar intercambiar información se lo hace a través de un bus central. No es necesaria la utilización de una unidad central de control, como se observa en la Figura 1.3.

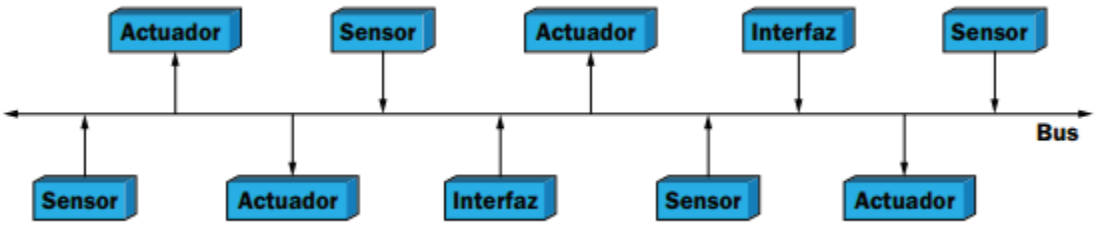

Figura 1.3. Sistema domótico de arquitectura descentralizada [8, p. 32]

Los diferentes elementos del sistema tienen la capacidad de recibir información, procesarla y en consecuencia, actuar de forma independiente, es decir, que cada sensor o actuador posee también un controlador. En este tipo de sistemas, el mal funcionamiento de cualquier elemento no afecta el funcionamiento global del sistema. Además, brinda un mayor grado de flexibilidad y permiten ahorrar en el cableado de la instalación.

Si bien este tipo de sistemas eliminan los problemas asociados a sistemas centralizados y disminuyen los costos del cableado, su costo total es mayor y requiere programar de forma independiente cada uno de sus elementos. Además, ya que es necesario que todos los equipos puedan comprender los mensajes recibidos, es necesaria la estandarización de los mensajes que se intercambian entre los diferentes nodos.

Arquitectura distribuida [8]

En esta arquitectura, cada controlador del sistema está encargado de un par sensor-actuador o un pequeño grupo de pares, como se puede ver en la Figura 1.4.

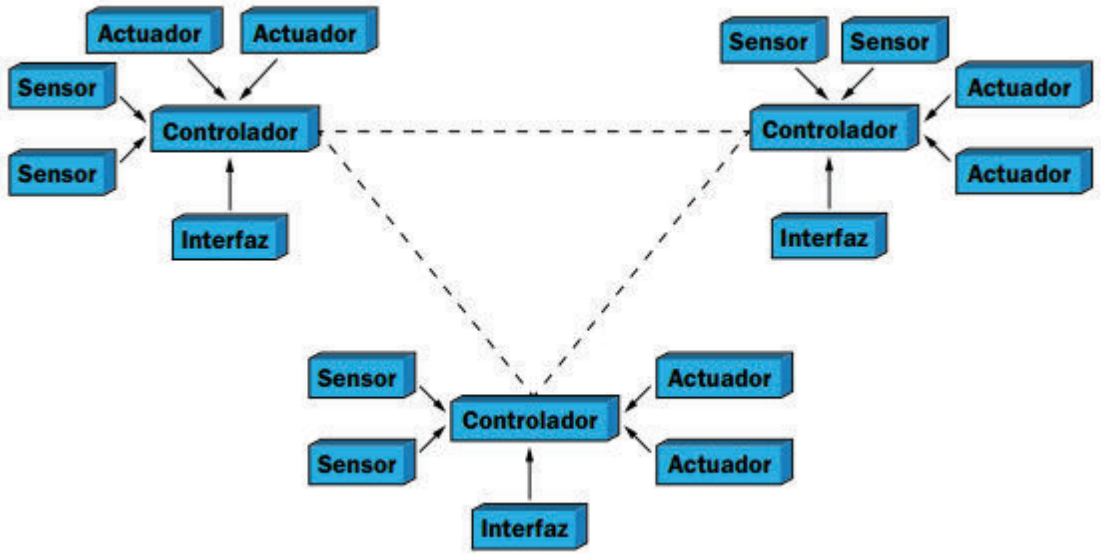

Figura 1.4. Sistema domótico de arquitectura distribuida [8, p. 32]

Estos sistemas brindan facilidad de ampliación al facilitar la adición de otros sensores y actuadores respecto a la arquitectura centralizada, aunque requieren que cada controlador del sistema sea programado de manera independiente.

1.3.1.3. Componentes de un sistema domótico [7] [10]

Controlador

Se dice que el controlador actúa como el cerebro del sistema domótico. La unidad de control o controlador es el dispositivo encargado de recoger toda la información generada por las entradas del sistema (pulsadores, sensores, interfaz de usuario, etc.), procesarla, y enviar los datos necesarios hacia un elemento de salida (electrodoméstico, interfaz de usuario, etc.) para que lleve a cabo determinada tarea.

La información recibida por la unidad de control es tratada de acuerdo al algoritmo de control con el que fue programado, es decir, en base a un conjunto de reglas predefinidas que fueron introducidas en la memoria del dispositivo [12, p. 27].

Adicionalmente, el controlador se encarga de notificar a la interfaz de usuario el estado de los diferentes componentes del sistema.

Las unidades de control se caracterizan por el número de entradas y salidas que poseen y trabajan con señales analógicas o digitales. La memoria y la capacidad de procesamiento son aspectos que también se deben tomar en cuenta al momento de elegir una unidad de control [8, p. 68].

Sensores

Son los dispositivos encargados de recopilar información acerca de diferentes parámetros o magnitudes físicas (humedad, temperatura, presión, etc.) y transmitirla hacia el controlador para que actúe en consecuencia.

Según el tipo de señal que proporcionan, se pueden clasificar a los sensores en dos grupos: sensores analógicos y digitales.

- **Sensores digitales:** Son aquellos que proporcionan información a través de una señal digital, que puede ser una secuencia de bits. En este grupo se encuentran los sensores binarios, que generan una señal digital que puede ser un 1 o un 0 lógicos, como un sensor magnético.
- **Sensores analógicos:** Proporcionan la información a través de una señal analógica, es decir, generan valores entre un mínimo y un máximo.

Existe una gran variedad de sensores empleados en la automatización de viviendas. Los más comunes suelen ser los de temperatura, humedad, gas, presencia, etc. A continuación, se describe brevemente la función de los principales sensores utilizados en aplicaciones domóticas.

- **Sensor de temperatura:** Es un dispositivo que transforma las variaciones de temperatura en variaciones de señales eléctricas. Por lo general un sensor de temperatura basa su funcionamiento en elementos como termopares o RTDs.

  El termopar es un sensor hecho de dos metales distintos, que generan un voltaje directamente proporcional a los cambios en temperatura. Un RTD (*Resistance Temperature Detector*) o detector de temperatura resistivo es una resistencia

variable que cambia su resistencia eléctrica de forma directamente proporcional a los cambios de temperatura, de manera precisa y de una forma casi lineal [13].

Estos sensores suelen ser utilizados por dispositivos como el termostato, el cual abre o cierra un contacto al superarse un determinado umbral de temperatura, para controlar de esta forma los sistemas de calefacción y aire acondicionado, y mantener una determinada temperatura en la vivienda [14].

Se considera adecuado instalarlos en lugares alejados de fuentes de calor, como radiadores, y otros fenómenos externos que puedan producir variaciones en las mediciones, como corrientes de aire, la incidencia directa de la luz del sol o la proximidad a aparatos que emanen calor (lámparas, televisores, etc.).

- **Sensor de movimiento:** Son elementos utilizados para la automatización de funciones como la iluminación y para la detección de intrusos. Estos sensores se basan principalmente en dos tecnologías: microondas e infrarrojos.

  Los sensores que utilizan ondas infrarrojas detectan movimiento en base a la detección de variaciones de temperatura en el ambiente, como las producidas con el paso de una persona. El uso de estos sensores se limita al cuarto donde fueron instalados, siempre que tengan una visión directa sobre dicha habitación.

  Los sensores de movimiento basados en tecnología microondas poseen un mayor alcance lo que les permite atravesar las paredes de varias habitaciones. No se recomienda su utilización en edificios de viviendas, ya que el movimiento en otros pisos podría generar falsas alarmas.

  Se recomienda ubicar este tipo de sensores en la esquina superior de la habitación, orientándolo para conseguir la mayor cobertura posible. Para evitar errores en la detección de movimiento, no deben ser instalados cerca de alguna fuente de calor, ya que la mayoría mide los cambios de temperatura para detectar movimiento.

- **Detector magnético** [15]**:** Suele ser utilizado para la vigilancia de puertas y ventanas, de tal manera que si uno de estos se abre, se acciona una alarma. Por lo general, los detectores magnéticos, también conocidos como contactos magnéticos, están compuestos por dos partes, una contiene un imán permanente, mientras que la otra parte consiste en un interruptor conocido como *reed switch*[1], el

[1] *Reed switch*: Es un interruptor formado por dos contactos, los cuales debido al material del que están fabricados, en presencia de un campo magnético toman polaridades diferentes. La atracción

cual al acercarse un campo magnético, cierra el circuito eléctrico que normalmente se encontraba abierto, mientras que al alejarse el imán, el campo magnético desaparece y los contactos del interruptor se separan regresando a su estado inicial [16].

El componente imantado del sensor deberá ubicarse sobre la parte móvil de la ventana o puerta, mientras que el componente fijo del sensor, que posee un cable, debe colocarse en el marco. Para que una mínima apertura de la puerta o ventana sea detectada, el sensor debe instalarse en el lado opuesto a donde se encuentran las bisagras.

- **Sensores de accionamiento manual:** En esta categoría se ubican dispositivos como interruptores, conmutadores, pulsadores, entre otros; que son utilizados en las instalaciones eléctricas convencionales. Sus mecanismos por lo general son simples, al ser accionados manualmente interrumpen o permiten el paso de corriente a través de un circuito eléctrico y permiten accionar sistemas, seleccionar opciones y otros parámetros.
- **Sensores de alarma médica y emergencia:** Se trata de un mando colgante o un pulsador, el cual una persona acciona en caso de emergencia. Al presionar el botón se genera una alarma, que puede incluir la generación de una alerta sonora, una llamada telefónica para pedir ayuda, u otra función que haya sido programada. Puede ser utilizado como un botón de pánico por personas mayores o con discapacidades.

Actuadores

Son elementos que al recibir instrucciones del controlador pueden modificar el estado de ciertas instalaciones y equipos en la vivienda. Dentro de este grupo se encuentran dispositivos como los contactores o relés de actuación, las electroválvulas, motores, reguladores de luz, entre otros. Los sensores y actuadores pueden estar integrados en un solo dispositivo, aunque no es muy frecuente.

En función del tipo de actuación que realizan, los actuadores pueden ser: analógicos o binarios (digitales).

---

de polos magnéticos opuestos causa que las dos piezas metálicas o contactos que estaban separadas se doblen una hacia la otra y hagan contacto. Esto se puede lograr al acercar un imán permanente al interruptor o al electrizar una bobina cercana al interruptor.

- **Actuadores analógicos:** Son aquellos que posibilitan la regulación de los elementos terminales, como las válvulas utilizadas para zonificar la calefacción, reguladores de intensidad luminosa, entre otros.
- **Actuadores binarios:** Permiten conectar o desconectar los elementos terminales, como los relés, las electroválvulas usadas para cortar el suministro, etc.

A continuación, se describe el funcionamiento del relé, al ser uno de los actuadores más comunes.

- **Relé** [17]**:** Es un interruptor accionado por una señal eléctrica. Suele ser utilizado para la conexión o desconexión de dispositivos que manejan una tensión mucho mayor a la señal de control.

  Básicamente, un relé es un dispositivo electromecánico formado por un electroimán y un interruptor de contactos. Al pasar corriente por la bobina del electroimán, se atrae a una pieza de hierro de uno de sus extremos, causando que el otro extremo empuje a uno de los contactos del interruptor hasta que se junten, permitiendo el paso de corriente. Esta corriente suele ser mucho mayor a la que circula por la bobina. El interruptor vuelve a la posición desconectado al dejar de actuar sobre el mismo la fuerza que mantenía unidos los contactos [18].

  Estos dispositivos permiten aislar eléctricamente el circuito de control, que maneja pequeñas tensiones y acciona el electroimán, de circuitos que pueden manejar altos voltajes o potencias, por lo que los relés son utilizados en una gran variedad de aplicaciones como sistemas eléctricos de potencia, sistemas de automatización del hogar, automóviles, equipos industriales, etc.

Dispositivos bajo control

Los dispositivos bajo control son todos aquellos componentes del hogar, como electrodomésticos o dispositivos electrónicos que están conectados y son controlados por el sistema de automatización.

Cada vez un número mayor de aparatos domésticos incluyen la funcionalidad que les permite conectarse de forma directa a la red de control de la vivienda, ya que cuentan con servidores web e interfaces WLAN (*Wireless Local Area Network*) o Bluetooth incluidas en el mismo electrodoméstico [12, p. 23].

Los denominados electrodomésticos inteligentes o domóticos son capaces de intercambiar información, comunicarse unos con otros, ser programados y controlados a través del

Internet. Dentro de este grupo se incluyen electrodomésticos como frigoríficos capaces de revisar las provisiones existentes dentro de la misma y alertar en caso de que un producto esté por terminarse, hornos inteligentes capaces de encenderse o apagarse de forma remota, pequeños dispositivos aspiradora que se activan cuando no existe nadie presente y pueden esquivar objetos gracias a radares y parachoques de alta sensibilidad, etc.

A diferencia de los electrodomésticos inteligentes, que suelen ser empleados en la realización de tareas cotidianas y pueden encontrarse principalmente en la cocina, los aparatos electrónicos son empleados para otras tareas como entretenimiento u ocio, y suelen ubicarse en las diferentes estancias de la vivienda. La tecnología incorporada en estos dispositivos posibilita que estos se comuniquen entre sí, intercambien información y accedan a servicios de Internet utilizando la red de datos de la vivienda.

El principal inconveniente en la utilización de estos electrodomésticos y otros aparatos inteligentes es su alto costo, que dificulta su implementación en hogares con sistemas domóticos.

Dispositivos para control remoto

Una de las principales razones en el incremento de la aceptación de los sistemas de automatización o sistemas domóticos en el segmento residencial es que, debido a la omnipresencia de teléfonos inteligentes y tabletas, la necesidad de dispositivos dedicados para el control del sistema domótico ha desaparecido [12, p. 27]. En los últimos años, prácticamente todos los sistemas de automatización del hogar disponibles en el mercado han introducido aplicaciones para el control del sistema que funcionan sobre teléfonos inteligentes o tabletas.

Los dispositivos para el control remoto del sistema domótico funcionan conectándose a la aplicación de control del hogar que reside en el controlador. La conexión se realiza a través de la misma red de control o utilizando cualquier otra interfaz que el controlador provea, ya sea mediante la red de datos de la vivienda, la red de telefonía móvil o utilizando el Internet. Es así, que el uso de *smartphones* posibilita el control remoto de una vivienda a través de Internet u otras redes.

1.3.2. Plataforma Arduino

No mucho tiempo atrás, trabajar con hardware significaba construir circuitos desde cero utilizando cientos de diferentes componentes electrónicos. Cada circuito era diseñado para una aplicación específica, y hacer cambios requería cortar cables, soldar conexiones, entre otras cosas.

Con la aparición de tecnologías digitales, microprocesadores, microcontroladores, entre otros, las funciones que una vez fueron implementadas con cables y otros componentes, fueron reemplazadas por programas de software. El software es más fácil de modificar que el hardware, lo que posibilita cambiar la lógica de un dispositivo y probar diferentes versiones de un programa en un tiempo mucho menor [19, p. 17].

Arduino es una plataforma de computación física de código abierto que simplifica el proceso de trabajo con microcontroladores. Entre otras cosas, puede ser utilizado para desarrollar objetos interactivos autónomos y puede ser conectado a software en una computadora [19, p. 1].

1.3.2.1. Características de la plataforma Arduino [20]

Arduino es una plataforma electrónica de código abierto que incluye componentes de hardware y software. Arduino está compuesto principalmente por la placa Arduino, que es la pieza de hardware en la que se trabaja al construir proyectos, y el entorno de desarrollo de Arduino, la pieza de software que se ejecuta en el computador y permite la creación de *sketches*, es decir, programas que se pueden cargar a la placa de Arduino y le indican a la placa qué hacer.

Hardware de la placa Arduino

La placa de Arduino está formada básicamente por una tarjeta de circuito impreso que contiene al microcontrolador y puertos asociados a entradas y salidas analógicas y digitales, como se observa en la Figura 1.5.

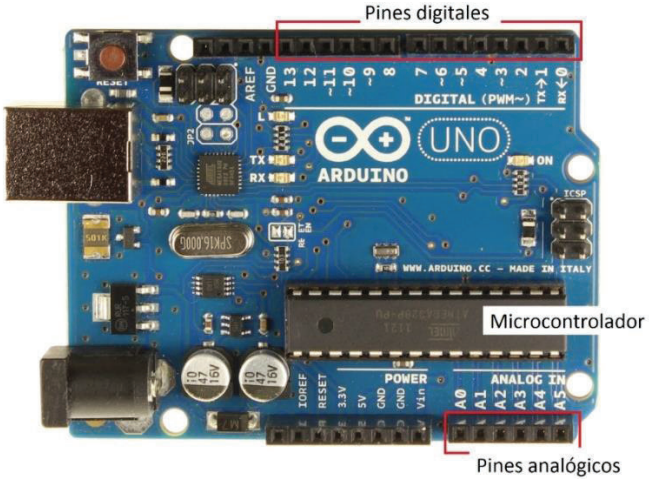

Figura 1.5. Principales componentes de la placa Arduino [21]

- **Microcontrolador** [22]**:** Un microcontrolador es un circuito integrado cuyo propósito es leer y ejecutar las líneas de código que componen el programa escrito por el

usuario. El hecho de que un microcontrolador sea programable brinda flexibilidad y modularidad, dado que un mismo circuito puede utilizarse en diferentes tareas tan solo al cambiar el programa que reside en la memoria del microcontrolador, lo que simplifica el diseño de circuitos electrónicos [23].

El microcontrolador es considerado como una computadora dentro de un pequeño chip y es el principal componente de la placa Arduino. Es miles de veces menos potente que el procesador de un computador, sin embargo, es mucho menos costoso y muy útil al construir diversos dispositivos. La placa Arduino contiene todos los componentes necesarios para que el microcontrolador funcione correctamente y pueda comunicarse con el computador [19, p. 18].

A diferencia de los microprocesadores que se han desarrollado principalmente para estaciones de trabajo y ordenadores personales, donde es necesario una potencia de cálculo elevada y velocidad de procesamiento, los microcontroladores suelen ser utilizados en diversas aplicaciones, como equipos de comunicaciones, instrumentos electrónicos, electrodomésticos, equipos industriales y médicos, entre otros. Los microcontroladores suelen ser utilizados en aplicaciones puntuales, donde realizan un limitado número de tareas a un bajo costo. En esas aplicaciones el microcontrolador ejecuta un programa que fue almacenado de forma permanente en su memoria y se comunica con el exterior mediante las entradas y salidas que posee.

Un microcontrolador está compuesto fundamentalmente de tres bloques: la Unidad Central de Procesamiento (CPU), la memoria, y los recursos de entrada y salida. Estos bloques se interconectan a través de buses o grupos de líneas eléctricas.

La CPU es el cerebro del microcontrolador. Realiza sus tareas en función del programa almacenado en memoria. Se encarga de obtener de la memoria las instrucciones del programa una a una, las interpreta (decodifica) y ejecuta. La CPU contiene la Unidad Aritmética y Lógica (ALU) encargada de realizar operaciones elementales con datos binarios. Además, la CPU cuenta con diferentes registros de propósito general o propósito específico.

La memoria del microcontrolador almacena las instrucciones del programa y los datos que utiliza. Un microcontrolador siempre tiene dos tipos de memoria: la memoria ROM (*Read Only Memory*) y la memoria RAM (*Random Access Memory).* El término acceso aleatorio hace referencia a que el tiempo necesario para encontrar un dato no depende del lugar en la memoria donde fue almacenado.

Ambas, las memorias RAM y ROM son de acceso aleatorio. La RAM es una memoria volátil de lectura y escritura, por lo que al interrumpirse la energía que alimenta la memoria pierde la información almacenada. La ROM es una memoria no volátil de solo lectura. Existen diferentes tecnologías para crear memorias ROM, como las memorias EEPROM y Flash. La memoria ROM permite almacenar de forma permanente las instrucciones que ejecuta el microcontrolador, mientras que la memoria RAM almacena de forma temporal los datos que necesita el programa.

- **Puertos de entrada/salida:** Los pines de entrada y salida del microcontrolador permiten su interacción con el exterior y se conectan a los puertos de la placa Arduino. Pueden utilizarse para conectar tarjetas de expansión conocidas como *shields* que permiten añadir funcionalidad a la placa de Arduino, además, pueden ser utilizados como puertos para comunicación serial, para gestionar interruptores y temporizadores. Un microcontrolador puede incluir también entradas o salidas analógicas asociadas a convertidores D/A o A/D.

Software de Arduino

Dentro del software de la plataforma Arduino se incluyen el entorno de desarrollo integrado, el lenguaje de programación y el cargador de arranque (*bootloader*).

- **Cargador de arranque** [24]**:** El *bootloader* es un programa sencillo que permite subir programas a la placa de Arduino sin la necesidad de utilizar hardware adicional externo. Suele ser almacenado en la memoria Flash del microcontrolador en la placa de Arduino. De forma general, son diseñados para ser pequeños y simples, ya que utilizan espacio, que de otro modo sería utilizado por la aplicación.

  Al reiniciar un controlador, este siempre empezará ejecutando el código ubicado en una locación de memoria particular. Si se coloca en esa dirección de memoria al *bootloader*, este será ejecutado cada vez que se encienda el microcontrolador o se presione el botón de *reset*, momento en el que el *bootloader* espera que llegue un nuevo *sketch* por el puerto serial desde el IDE de Arduino.

  En caso de recibir un *sketch*, este es almacenado en la memoria Flash reemplazando la aplicación que se encontraba guardada previamente, y la nueva se ejecuta. Al no recibir un programa nuevo, ejecuta el programa cargado previamente. Para indicar al *bootloader* si debe cargar una nueva aplicación o ejecutar una ya existente, se suele utilizar una señal externa, como un comando en el puerto serial.

- **Lenguaje de programación:** El lenguaje de programación para la plataforma Arduino está basado en el lenguaje C++. Para simplificar la programación del microcontrolador, el equipo de Arduino desarrolló una librería estándar que provee un conjunto de funciones, la misma que utiliza la librería avr-libc. La librería avr-libc contiene varias de las funciones de una librería C estándar y funciones específicas para microcontroladores Atmel AVR de 8 bits, ya sea, funciones para realizar operaciones matemáticas, funciones de bajo nivel asociadas a entradas/salidas, *timers* del sistema, entre otras [25].

- **Entorno de desarrollo integrado** [26]**:** El Entorno de Desarrollo Integrado o IDE de Arduino es un programa que se ejecuta en el computador y permite escribir *sketches* o programas para la placa de Arduino, como se muestra en la Figura 1.6. Está basado en el IDE de *Processing*[2], lo que brinda un entorno de desarrollo simple y fácil de utilizar [27]. Al momento de escribir un programa, este se almacena con la extensión ino. El código escrito en el IDE es traducido a lenguaje C++ y enviado al compilador avr-gcc, un compilador de C y C++, para generar un ejecutable a ser usado por el microcontrolador.

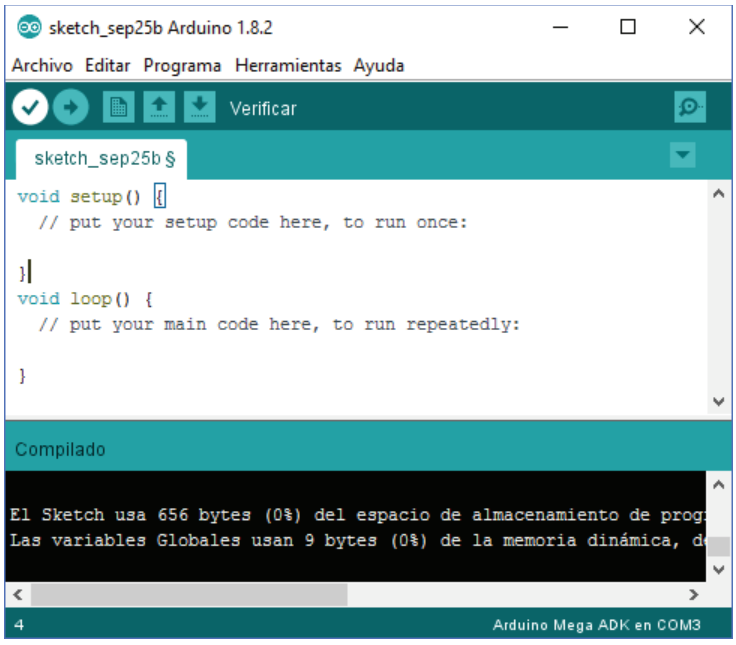

Figura 1.6. IDE de la plataforma Arduino

[2] *Processing*: Es un lenguaje de programación basado en Java diseñado específicamente para el desarrollo de aplicaciones gráficas. Permite utilizar una sintaxis más simple respecto a Java lo cual facilita su aprendizaje. Posee un IDE propio conocido como *Processing Developement Environment*, sencillo y fácil de utilizar.

El entorno de desarrollo de Arduino contiene un editor de texto para escribir código, una consola de texto, una barra de herramientas, un área de mensajes y una serie de menús. El área de mensajes brinda información al momento de guardar o exportar *sketches* y muestra los errores generados. La consola muestra una salida de texto que incluye mensajes de error detallados y otra información. La barra de herramientas permite crear, depurar y guardar programas, además de acceder al monitor serial.

1.3.2.2. Ventajas de la plataforma Arduino [19]

Además de Arduino, existen otras plataformas de funcionalidad similar. Sin embargo, Arduino posee algunas ventajas respecto a otras opciones disponibles:

- **Costo bajo:** Las placas Arduino tienen un menor costo respecto a otras plataformas microcontroladoras como BeagleBone o Raspberry Pi.
- **Multiplataforma:** A pesar de que la mayoría de plataformas microcontroladoras se limitan a Windows, el software de Arduino trabaja en los sistemas operativos GNU/Linux, Mac OS y Windows. Se puede programar mediante un cable USB y no un puerto serial, lo que resulta útil dado que la mayoría de computadoras actuales no poseen puertos seriales.
- **Entorno de programación claro y simple:** El entorno de programación es fácil de utilizar para usuarios inexpertos, y flexible para usuarios experimentados. El entorno de desarrollo de Arduino se basa en el IDE de programación de *Processing*, un entorno de desarrollo fácil de utilizar.
- **Software extensible y de fuente abierta:** Arduino fue publicado como una herramienta de código abierto. Se puede añadir librerías propias de C++.
- **Hardware extensible y de fuente abierta:** Arduino utiliza principalmente los microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos de las diferentes placas están publicados con una licencia *Creative Commons*, lo que permite la creación de una versión propia de la placa, extendiendo su funcionalidad y mejorando sus características. Si se desea, se puede descargar el diagrama de circuito, adquirir los componentes necesarios y fabricar la placa por cuenta propia.

1.3.2.3. Modelos de Arduino

Existen varios modelos de placas que pueden ser utilizadas en una amplia variedad de aplicaciones en combinación con Arduino.

En la Tabla 1.1 se presenta una comparación de las principales placas disponibles: Arduino 101, Arduino Lilypad, Arduino Mega 2560, Arduino Uno, Arduino Nano, Arduino Leonardo y Arduino Due.

Tabla 1.1. Comparación de las principales placas de Arduino [28]

Nombre	Procesador	Voltaje de operación	Frecuencia de reloj	Pines analógicos	Pines digitales	Puertos seriales
101	Intel Curie	3.3 V	32 MHz	6	14	1
Lilypad	ATmega168V ATmega328P	2.7-5.5 V	8 MHz	6	14	-
Mega 2560	ATmega2560	5 V	16 MHz	16	54	4
Uno	ATmega328P	5 V	16 MHz	6	14	1
Nano	ATmega168 ATmega328P	5 V	16 MHz	8	14	1
Leonardo	ATmega32U4	5 V	16 MHz	12	20	1
Due	ATSAM3X8E	3.3 V	84 MHz	12	54	4

Para seleccionar la placa Arduino, se tomaron en cuenta los siguientes aspectos:

- **Pines analógicos:** Se requiere al menos un pin analógico para realizar la medición de temperatura a partir de la señal generada por el sensor.
- **Pines digitales:** Se requiere contar con al menos 25 pines digitales para recibir datos de los sensores, enviar órdenes a los actuadores y conectar los módulos o *shields* para ampliar las capacidades de Arduino al comunicarse con la aplicación móvil.
- **Puertos seriales:** Son necesarios 2 puertos seriales que posibiliten conectar la placa a un módulo Bluetooth y permitan vincular la placa al computador para depurar la aplicación.

Las placas Arduino Mega 2560 y Arduino Due cumplen con los requisitos necesarios para actuar como controlador del sistema domótico. Sin embargo, se seleccionó la placa Arduino Mega 2560 debido a que presenta un costo menor en comparación a la placa Arduino Due.

A continuación, se presenta una descripción de la placa Arduino Mega y de las placas Ethernet *shield* y módulo Bluetooth, necesarias para la comunicación a través de Internet y Bluetooth.

Placa Arduino Mega 2560 [29]

Esta placa de Arduino está formada principalmente por un microcontrolador ATmega2560 y entradas y salidas analógicas y digitales.

Como se observa en la Figura 1.7, esta placa cuenta con 54 pines digitales, los cuales pueden ser utilizados como entradas o salidas, operan a 5 V y pueden proveer o recibir hasta un máximo de 40 mA. La placa tiene 16 entradas analógicas, cada una de las cuales provee una resolución de 10 bits, es decir, puede representar hasta 1024 valores diferentes. Por defecto, los pines analógicos miden desde 0 V (valor correspondiente a tierra) a 5 V. De ser necesario los pines analógicos pueden ser utilizados como pines digitales de entrada/salida.

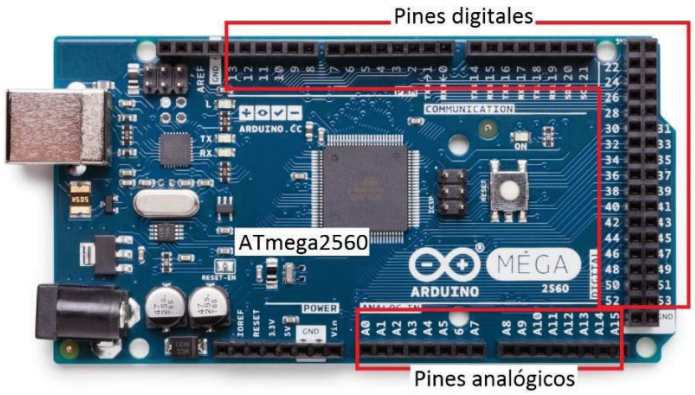

Figura 1.7. Placa Arduino Mega 2560 [30]

- **Memoria:** El microcontrolador de ATmega2560 posee 256 KB de memoria Flash, 8 KB de SRAM (memoria estática de acceso aleatorio) y 4 KB de EEPROM (memoria programable borrable de solo lectura).

  La memoria EEPROM es una forma de memoria no volátil. Es una memoria de solo lectura que puede ser borrada y reprogramada. Los datos contenidos en esta memoria solo pueden ser leídos o escritos byte a byte.

  La memoria Flash es utilizada principalmente para almacenar el código de usuario, el *bootloader* y cualquier dato con un valor inicial. La memoria Flash es un tipo de memoria EEPROM, sin embargo, en una memoria Flash es posible realizar operaciones de lectura/escritura con bloques completos de datos. Una memoria Flash es generalmente borrada en bloques de 64 KB. El número de ciclos de escritura que este tipo de memoria soporta generalmente es menor a los ciclos de escritura que una memoria EEPROM puede manejar, es decir, que se puede escribir en una memoria EEPROM un mayor número de veces que en una memoria

Flash. Sin embargo, el tiempo de acceso a una memoria EEPROM es mayor en comparación a la memoria Flash [31].

La memoria SRAM puede ser leída y escrita desde el programa en ejecución. Almacena variables estáticas y globales, como variables con valores iniciales, en cuyo caso el sistema copia el valor inicial desde la Flash cuando el programa inicia. También almacena las variables locales y un registro de interrupciones y llamadas a funciones.

- **Comunicación serial:** Este microcontrolador posee cuatro UARTs (Transmisor-Receptor Asíncrono Universal) o puertos seriales en hardware. La comunicación serial en los pines de transmisión o recepción utiliza la lógica TTL (5 V o 3.3 V). Un puerto serial utiliza tres pines para la comunicación: TX, RX y GND, donde el pin TX de un dispositivo se conecta al pin RX del otro dispositivo.

  El microcontrolador cuenta con 4 puertos seriales en hardware. El puerto Serial 0 está ubicado en los pines 0 (RX) y 1 (TX), el puerto Serial 1 se ubica en los pines 19 (RX) y 18 (TX), el puerto Serial 2 se ubica en los pines 17 (RX) y 16 (TX) y Serial 3 en los pines 15 (RX) y 14 (TX).

- **Interrupciones externas:** La placa dispone de 6 pines destinados a interrupciones. La interrupción 0 se ubica en el pin 2, la interrupción 1 en el pin 3, la interrupción 2 en el pin 21, la interrupción 3 en el pin 20, la interrupción 4 en el pin 19 y la interrupción 5 en el pin 18. Estos pines pueden ser configurados para activar una interrupción al recibir un nivel bajo o 0L, al producirse un cambio de estado o por un flanco de bajada o subida, producidos al cambiar de estado de 1L a 0L y de 0L a 1L respectivamente.

- **PWM (*Pulse Width Modulation*):** De los 54 pines de entrada/salida digitales, 14 pueden ser utilizados como salidas PWM. Se pueden generar señales PWM desde los pines del 2 al 13 y del 44 al 46. A la salida se obtiene una señal cuyo ancho de pulso presentará una variación de acuerdo a un valor específico entre 0 y 255, es decir, se puede cambiar el ciclo de trabajo o el tiempo que la señal digital está en estado lógico alto.

  Para proteger los puertos USB del computador de cortos y sobrecorriente, la placa cuenta con fusibles reajustables. Si se aplican más de 500 mA de corriente al puerto USB, el fusible abrirá el circuito hasta que el corto circuito o la sobrecarga sean removidos.

- **Alimentación:** Se puede alimentar a la placa a través desde el computador mediante un puerto USB o con una fuente externa, ya sea un adaptador AC a DC o una batería. Al utilizar una fuente externa de poder en lugar de alimentar a la placa a través de un puerto USB, se puede utilizar el pin V_{in} o el conector de alimentación con un voltaje de 7 a 12 V.

- **Programación:** Para cargar un programa al microcontrolador es necesario utilizar el IDE de Arduino y una conexión al computador a través de un puerto USB. Esto es posible gracias a que el microcontrolador ATmega2560 de la placa viene preprogramado con un *bootloader*, lo que evita el uso de hardware adicional para programar el microcontrolador.

  Sin embargo, es posible programar el microcontrolador sin hacer uso del *bootloader*. Al cargar un programa directamente al microcontrolador utilizando hardware externo mediante el protocolo SPI (Interfaz Periférica Serial), se puede remover el *bootloader* de la memoria Flash y usar el espacio extra para grabar el programa. También se pueden utilizar los pines SPI para grabar el *bootloader* en el microcontrolador.

  SPI es un protocolo que describe la transferencia serial síncrona de datos entre dos dispositivos, un maestro y un esclavo, posibilitando que estos dispositivos se comuniquen entre ellos. Es utilizado por microcontroladores para comunicarse con uno o más dispositivos periféricos o para comunicarse con otro microcontrolador [32]. En el proceso de comunicación se utilizan líneas separadas para el envío de datos, señal de reloj y la señal de selección. A continuación, se describen las cuatro líneas utilizadas:

  - **MOSI (*Master Out – Slave In*):** Está ubicado en el pin 51 de la placa Arduino. Es una línea para la transmisión de datos desde el maestro a un esclavo. Es una línea común a todos los dispositivos.

  - **MISO (*Master In – Slave Out*):** Está ubicado en el pin 50 de la placa Arduino. Es una línea para la transmisión de datos desde un esclavo a un maestro. Es una línea común a todos los dispositivos.

  - **SCLK (*Serial Clock*):** Ubicado en el pin 52 de la placa. Una señal de reloj enviada desde el maestro a todos los esclavos. Es una línea común a todos los dispositivos.

- **SS (*Slave Select*):** Ubicado en el pin 53 de la placa en caso de que esta actúe como esclavo. Al trabajar como maestro, se utiliza un pin por cada esclavo para seleccionar con el dispositivo que se comunicará.

En la Tabla 1.2 se resumen las principales características de la placa. Para mayor detalle ver el Anexo I.

Tabla 1.2. Características de la placa Arduino Mega 2560

Característica	Descripción
Microcontrolador	ATmega2560
Voltaje de operación	5 V
Voltaje de alimentación	7 a 12 V
Pines digitales	54 pines digitales de entrada/salida
Pines analógicos	16 pines de entrada analógicos
Corriente máxima por pin de entrada/salida	40 mA
Memoria	256 KB de memoria Flash de los cuales 8 KB son utilizados por el *bootloader* 8 KB de SRAM 4 KB de EEPROM
Velocidad del reloj	16 MHz
Comunicación serial	El microcontrolador cuenta con 4 puertos seriales en hardware
Interrupciones	La placa dispone de 6 pines destinados a interrupciones
PWM	14 pines permiten obtener a la salida una señal modulada por ancho de pulso o señal PWM

Ethernet *shield*

La limitada memoria de Arduino imposibilita implementar un *stack* TCP/IP completo en esta placa. En su lugar, el *shield* Ethernet permite conectar una placa Arduino a Internet. El *shield* está basado en el Wiznet Ethernet W5100, un chip diseñado para aplicaciones embebidas que implementa protocolos como Ethernet, TCP, UDP e IP. Este chip soporta hasta cuatro conexiones simultáneas, lo que quiere decir que se podría ejecutar un servidor

que permita hasta cuatro usuarios conectados simultáneamente. Sin embargo, por la forma en que fue diseñada la librería Ethernet, el Ethernet *shield* permite una sola conexión.

Como se muestra en la Figura 1.8, esta placa incluye un *slot* para una tarjeta micro-SD, que puede ser utilizada para almacenar archivos y enviarlos a través de la red. La placa Arduino se comunica con el chip W5100 y con la tarjeta SD utilizando el bus SPI de la placa, es decir, a través de los pines 50 (MISO), 51 (MOSI) y 52 (SCLK). Los pines 10 y 4 se conectan al chip W5100 y a la tarjeta SD respectivamente, y permiten seleccionar con cual de estos dispositivos se comunica el microcontrolador, por lo que estos pines no pueden ser utilizados para otro propósito.

Debido a que el chip W5100 y la tarjeta SD comparten el bus SPI, es decir, comparten las líneas de MOSI, MISO y SCLK, solo uno de ellos puede estar activo a la vez. En caso de no utilizar una tarjeta SD, esta debe ser desactivada, para lo cual es necesario configurar el pin 4, que permite la selección de esclavo, como salida en estado alto o 1L [33].

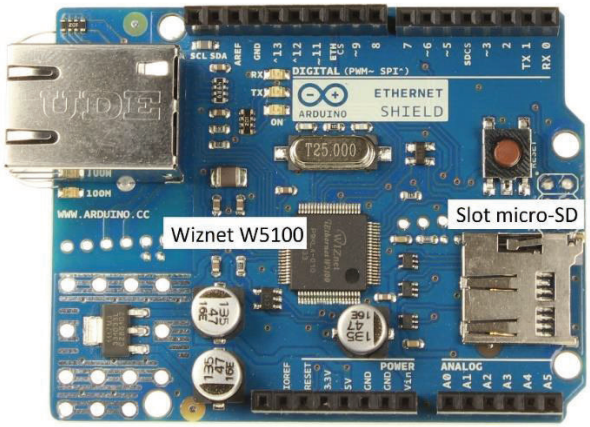

Figura 1.8. Ethernet *shield* [34]

El *shield* contiene una serie de leds que indican si el *shield* Ethernet está energizado, la presencia de un enlace de red, si la conexión es *half duplex* o *full duplex*, si la conexión es de 10 Mbps o 100 Mbps, si el *shield* transmite o recibe tráfico, e indica si se detectan colisiones en la transmisión.

La hoja técnica de esta placa se muestra en el Anexo II.

Módulo Bluetooth

La función principal del módulo serial Bluetooth es reemplazar la línea de conexión serial entre dos dispositivos.

Un módulo Bluetooth puede desempeñar el rol de maestro o esclavo. El maestro tiene la habilidad de iniciar una conexión con un dispositivo esclavo. Una vez establecida la conexión entre los dispositivos estos pueden iniciar el intercambio de información. El maestro puede enviar información a cualquiera de los esclavos y solicitar información de estos. Los esclavos únicamente pueden transmitir y recibir de su maestro [35].

El módulo Bluetooth HC-05 puede ser configurado como maestro o esclavo, mientras que los módulos HC-04 y HC-06 funcionan únicamente como esclavos, por lo que necesitan de un dispositivo que actúe como maestro.

La configuración de los módulos Bluetooth se realiza a través de comandos AT. Los comandos AT son instrucciones, pequeñas secuencias de caracteres, que suelen ser utilizadas para configurar o controlar un módem. La palabra AT es utilizada como prefijo para iniciar cada comando del módem y no forma parte del nombre del comando. Para finalizar un comando se envían los caracteres *carriage return* (\r) y *line feed* (\n) [36].

Para realizar la conexión de este dispositivo a la placa, el pin de transmisión (TX) del microcontrolador se debe conectar al pin de recepción (RX) del módulo Bluetooth, y el pin RX del microcontrolador al pin TX del módulo Bluetooth.

El módulo tiene dos modos de operación, el modo de comandos y el modo de datos. En el modo de comandos es posible configurar parámetros como el nombre del dispositivo, la contraseña, entre otros. El modo de datos permite al módulo Bluetooth transmitir y recibir datos de otro dispositivo.

Como se muestra en la Figura 1.9, el módulo Bluetooth incluye principalmente un chip de memoria, un cristal de 26 MHz, un transceptor o *transceiver*, dispositivo que incluye un transmisor y un receptor, que se conectan a la misma antena.

Se incluyen los pines de conexión de las señales de datos y alimentación del módulo, además de un regulador de voltaje de 5 V a 3.3 V.

El módulo Bluetooth cuenta principalmente con los siguientes pines:

- **TX:** Salida serial del módulo que se conecta al pin RX del microcontrolador.
- **RX:** Entrada serial del módulo que se conecta al pin TX del microcontrolador.
- **GND:** Pin de conexión a tierra.
- **Vcc:** El voltaje de alimentación debe estar en el rango de 3.3 a 6 V.

La hoja técnica del módulo HC05 se adjunta en el Anexo III.

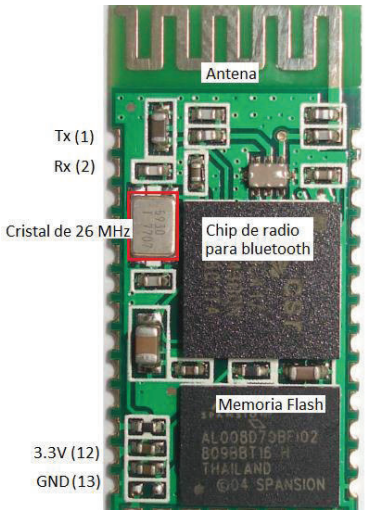

Figura 1.9. Módulo HC-05 [37]

1.3.3. Sistema Operativo Android

Android es una plataforma de software basado en el *kernel* de Linux enfocado a trabajar en dispositivos móviles.

La constante mejora de sus diferentes versiones ofreciendo nuevas características en cada una de ellas, además de la colaboración con fabricantes de dispositivos móviles contribuyó a la amplia difusión de este sistema operativo.

1.3.3.1. Arquitectura de la plataforma Android [38] [39]

Android es una pila o *stack* de software creada para diversos dispositivos. Un *stack* es un conjunto de componentes de software estructurados en forma de capas, donde una capa superior tiene un mayor nivel de abstracción, a diferencia de capas inferiores, que son más cercanas al hardware del dispositivo.

Las diferentes capas que conforman la arquitectura Android se resumen en la Figura 1.10 y se muestran a continuación:

1. *Kernel* de Linux
2. Capa abstracción de hardware
3. Librerías / Entorno de ejecución de Android
4. Capa de marco de trabajo de aplicaciones
5. Capa aplicación

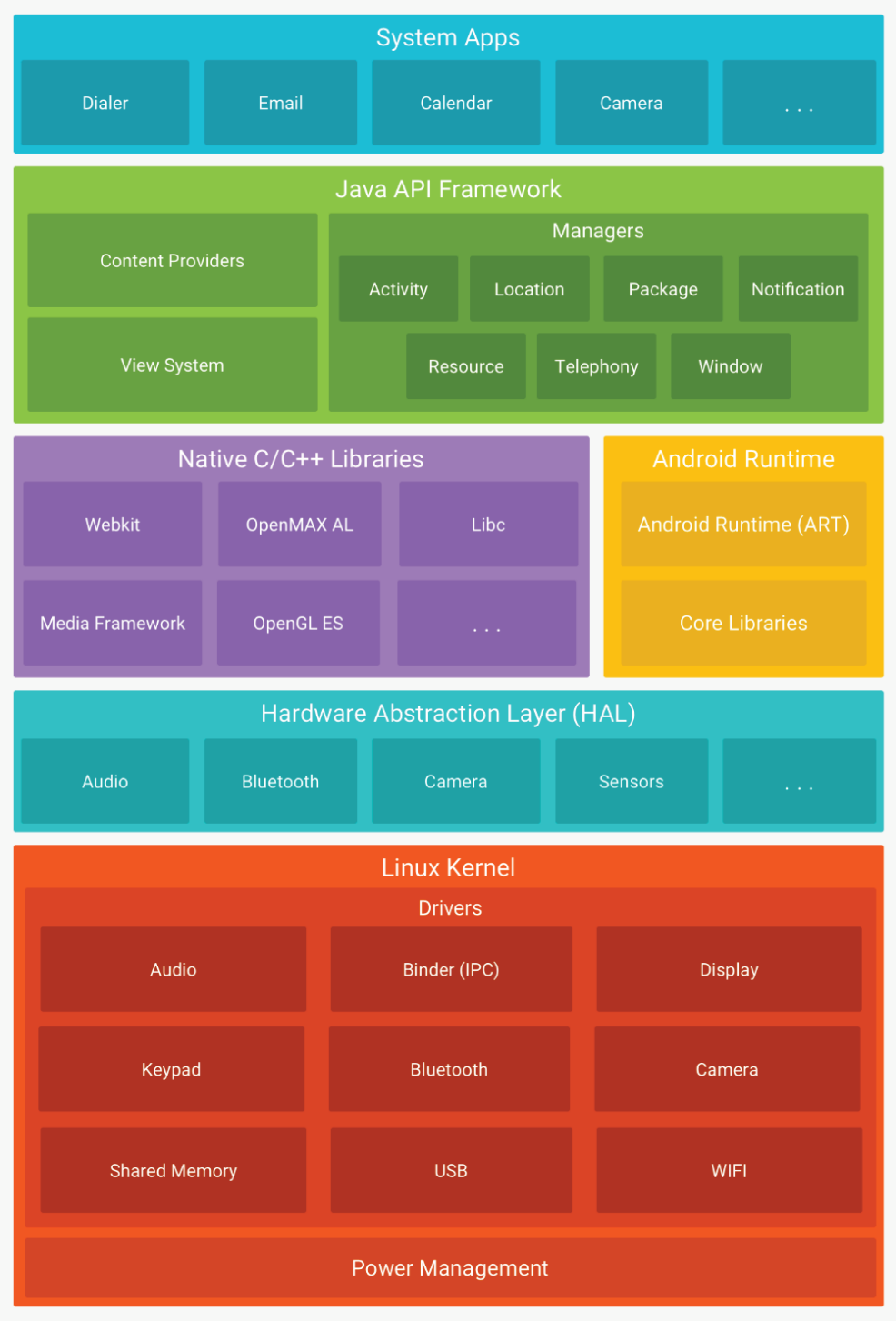

Figura 1.10. Arquitectura de la plataforma Android [38]

Kernel de Linux

El *kernel* de Linux provee un nivel de abstracción entre el hardware del dispositivo móvil y las capas superiores del *stack* de software de Android. El *kernel* provee servicios de sistema de bajo nivel como la gestión de energía, procesos y memoria, además de brindar una gran variedad de *drivers* para que cualquier componente de hardware pueda ser

utilizado haciendo uso de las librerías de control o *drivers* correspondientes. Cuando un fabricante incluye un nuevo elemento de hardware en un dispositivo móvil es necesario crear los *drivers* en el *kernel* de Linux de Android.

Capa de abstracción de hardware

Está formado por diferentes módulos cada uno de los cuales implementa una interfaz para un componente de hardware específico. Brinda una interfaz única hacia las implementaciones de diferentes fabricantes de hardware.

Librerías y entorno de ejecución de Android

Sobre el *kernel* de Linux se ubican un conjunto de librerías C/C++ utilizadas por los componentes del sistema Android. Varios servicios y componentes de este sistema operativo están basados en código nativo que requiere estas bibliotecas. La plataforma Android brinda una API para exponer la funcionalidad de algunas de estas librerías a las aplicaciones, lo que permite por ejemplo, acceder a la librería OpenGL ES mediante la API Java OpenGL y así añadir a la aplicación móvil compatibilidad con dibujos y manipular gráficos en 2D y 3D. A continuación, se describen algunas de las librerías principales:

- **Surface Manager:** Se encarga de preparar los elementos de navegación de pantalla, además de gestionar ventanas correspondientes a las diferentes aplicaciones activas.
- **OpenGL|SL:** Es una librería que permite manejar gráficos en 3D.
- **Media libraries:** Brinda los códecs necesarios para la grabación y reproducción de contenido multimedia, como video, audio, imágenes animadas y estáticas.
- **SGL:** Es una librería que provee gráficos en 2D, por lo que suele ser una librería comúnmente utilizada por las aplicaciones. Esta librería gráfica junto con OpenGL/SL, conforman la capacidad gráfica de Android.
- **FreeType:** Permite trabajar de forma simple y rápida con diferentes tipos de fuentes.
- **SSL (*Secure Sockets Layer*):** Es utilizado para proveer seguridad en Internet, ya que posibilita la utilización de SSL para establecer comunicaciones seguras.
- **SQLite:** Esta librería provee un motor de base de datos que se encuentra disponible para todas las aplicaciones. Permite crear y gestionar bases de datos relacionales.

- **WebKit:** Actúa como un motor para aplicaciones tipo navegador y es el núcleo del *browser* incluido por defecto en Android.
- **Libc:** Es una derivación de la librería estándar para el lenguaje de programación C libc, pero optimizada para dispositivos embebidos basados en Linux.

Al mismo nivel que las librerías de Android se encuentra el entorno de ejecución o *Android Runtime*, que está formado por el ART (*Android Runtime*) y un conjunto de librerías centrales. ART es el entorno de ejecución de aplicaciones usado por el sistema operativo Android. Desde la versión 5.0 de Android, ART reemplaza a la máquina virtual Dalvik utilizada originalmente por Android y se encarga de ejecutar las aplicaciones en el dispositivo.

Como se muestra en la Figura 1.11, el proceso de compilación de aplicaciones Android es muy diferente del que se lleva a cabo con otras aplicaciones Java.

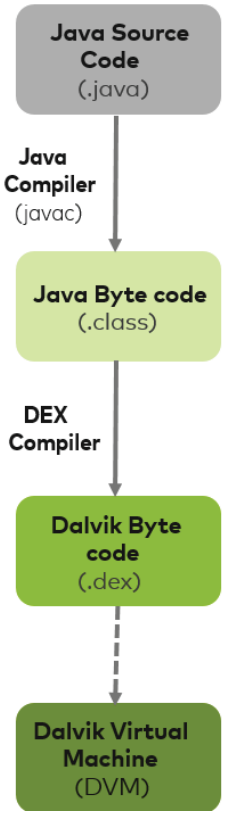

Figura 1.11. Proceso de compilación en Android [40]

Para empezar, el código fuente es compilado para generar el Java ByteCode. Posteriormente los ficheros de extensión class son transformados en ficheros de extensión dex (*Dalvik Executable*), para que sea ejecutado por el ATR, el que finalmente convertirá esos ficheros a código máquina.

Marco de trabajo de aplicaciones

El conjunto de las funciones del sistema operativo se encuentra disponible a través de APIs escritas en lenguaje Java. Estas API permiten la creación de aplicaciones simplificando la reutilización de componentes del sistema y servicios centrales y modulares.

Todas las aplicaciones para Android ya sean aplicaciones desarrolladas por Google o por terceros, utilizan las mismas APIs presentadas en este nivel.

El Java API Framework contiene un conjunto de bloques que son utilizados por las diferentes aplicaciones. Está preinstalado en los dispositivos Android y consiste de los siguientes componentes:

- **Gestor de actividades:** Este componente se encarga de gestionar el ciclo de vida de una aplicación.
- **Proveedores de contenido:** Este componente permite que una aplicación comparta datos con otras aplicaciones, de tal forma que, aplicaciones como la información de la agenda, los contactos, los mensajes, entre otros, sean accesibles a otras aplicaciones.
- **Window Manager:** Gestiona las diferentes ventanas de las aplicaciones. Utiliza la librería Surface Manager.
- **Telephone Manager:** Incluye las API relacionadas a las funciones propias de un teléfono, como llamadas, mensajes, etc.
- **Vista del sistema:** Brinda los elementos necesarios para poder construir interfaces de usuario (GUI), como botones, *check boxes*, listas, entre otros.
- **Gestor de ubicaciones:** Este componente hace posible que las aplicaciones de un dispositivo Android tengan acceso a información de posicionamiento o localización.
- **Gestor de notificaciones:** Este componente permite que una aplicación notifique al usuario, utilizando un mismo formato, sobre un evento significativo, como la recepción de un mensaje, sin interrumpir la tarea que el usuario está efectuando.

Capa aplicación

Este nivel incluye tanto las aplicaciones incluidas por defecto, como aquellas aplicaciones creadas por terceros. Dentro de las aplicaciones incluidas en Android se encuentran aquellas empleadas para el correo electrónico, envío de SMS, calendario, contactos, teclado, entre otros. Las *apps* incluidas en el sistema cumplen con funciones claves a las

que los desarrolladores pueden acceder desde sus aplicaciones. Por ejemplo, al requerir enviar mensajes de texto SMS desde nuestra aplicación, en lugar de codificar esa funcionalidad desde cero, se puede utilizar una *app* existente.

1.3.3.2. Componentes de una aplicación [41]

Un componente es un bloque utilizado al crear una aplicación. Cada componente existe como un ente individual y desempeña una tarea específica que contribuye a definir el comportamiento de la aplicación.

Existen principalmente cuatro tipos de componentes: actividades, servicios, proveedores de contenido y receptores de mensajes.

Actividades [42]

Una actividad es un componente de la aplicación asociada típicamente a una ventana con una interfaz de usuario. Este componente es implementado utilizando la clase `Activity`. Por lo general, las aplicaciones contienen varias actividades vinculadas entre sí, sin embargo, la actividad que se presenta al usuario cuando este inicia una aplicación se denomina actividad principal.

Como se observa en la Figura 1.12, una actividad posee un ciclo de vida definido. Los diferentes estados de una actividad son activa, pausada, parada y cerrada, los cuales son descritos a continuación:

- **Estado activo:** Una actividad está activa cuando se encuentra en primer plano (*foreground*) en la pantalla, es decir, es visible e interactúa con el usuario.
- **Estado pausado:** Una actividad pausada es aquella que aún es visible, pero no interactúa con el usuario. Una actividad en este estado se está ejecutando, pero podría ser cerrada por el sistema de ser necesario. Una actividad pausada conserva su estado y la información de las variables miembro. Una actividad pasa a este estado al colocar sobre esta otra actividad que no cubre toda la pantalla, una actividad transparente o al cerrar la actividad. Una vez que la actividad ha sido obstruida completamente y ya no sea visible, esta actividad pasa al estado parado o detenido.
- **Estado detenido:** Una actividad en este estado no es visible por el usuario, pero continúa ejecutándose. Una actividad detenida conserva su estado y la información de las variables miembro. Una actividad detenida podría ser cerrada por el sistema de ser necesario.

- **Estado cerrado:** Este estado indica que la actividad ha sido finalizada por el sistema. Si Android finaliza el proceso asociado a una aplicación, todas sus actividades son cerradas.

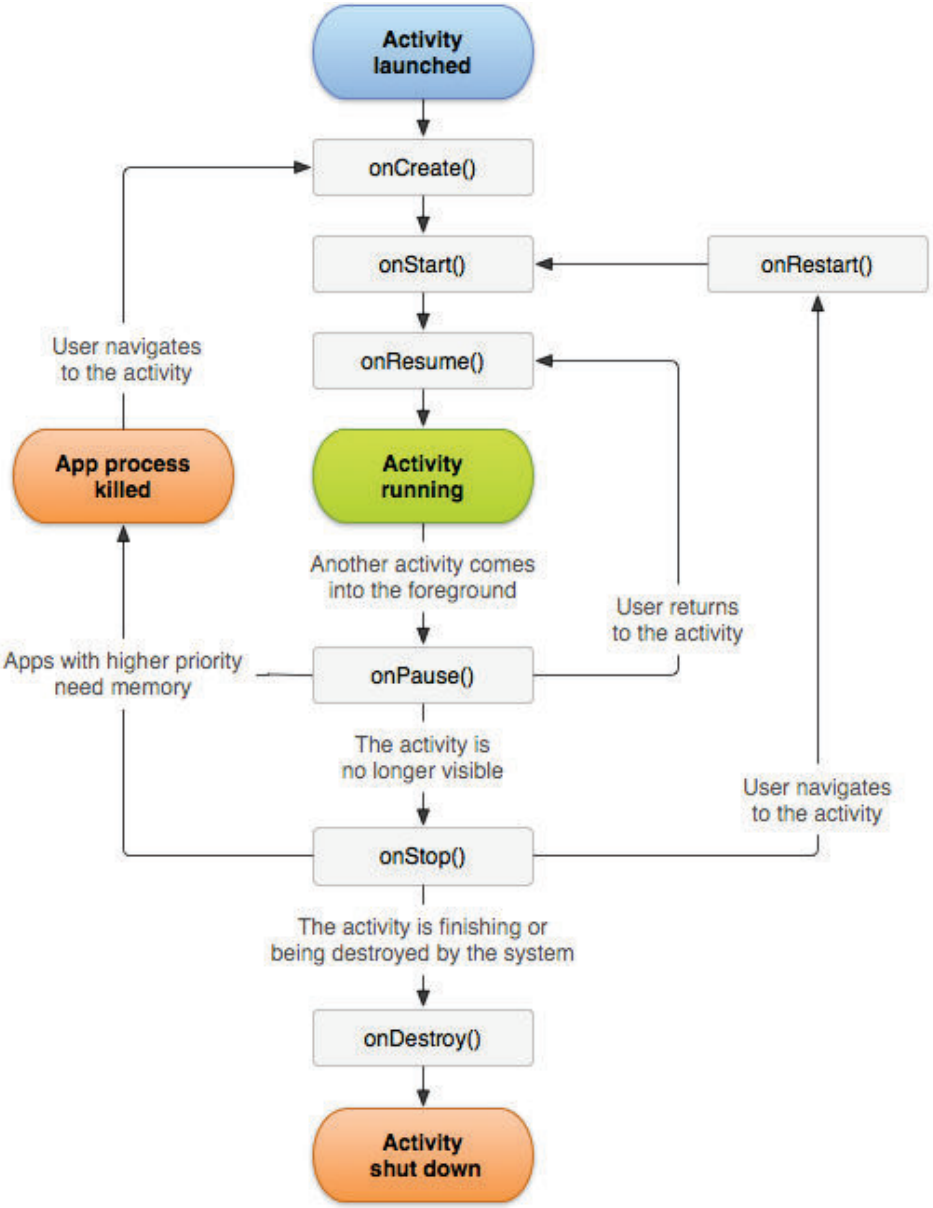

Figura 1.12. Ciclo de vida de una actividad [43]

La transición de una actividad por cada uno de los estados mencionados, es decir, el ciclo de vida de una actividad es controlado por 7 métodos de la clase `Activity`: `onCreate`, `onStart`, `onResume`, `onPause`, `onStop`, `onRestart` y `onDestroy`.

Receptores de mensajes *broadcast*

El sistema y las aplicaciones de Android utilizan mensajes de *broadcast* para avisar a otras aplicaciones sobre un evento de interés [44]. Un receptor de mensajes de *broadcast*

permite registrar los eventos producidos por las aplicaciones o el sistema y llevar a cabo determinada acción. Los receptores se asocian a un evento determinado y son notificados por el *Android Runtime* una vez que el evento se presenta. Este componente no posee una interfaz de usuario, pero puede recibir y responder a eventos como la recepción de mensajes SMS, aviso batería baja, inicio de la carga del dispositivo, entre otros.

Servicios [45]

Un servicio es un componente que se ejecuta en segundo plano utilizado para llevar a cabo tareas de larga ejecución que no requieren interacción con el usuario. Adicionalmente, permite a la aplicación exponer su funcionalidad a otras aplicaciones.

A diferencia de una actividad, un servicio no brinda una interfaz de usuario y puede ser iniciado desde otro componente como una actividad. Además, al igual que otros componentes, un servicio se ejecuta en el hilo principal del proceso asociado a la aplicación, por lo que se debería generar un hilo para que el servicio lleve a cabo tareas que hagan un uso intensivo del CPU o realicen operaciones bloqueantes. Estos componentes suelen ser utilizados para tareas repetitivas o que requieren un largo tiempo para su ejecución, como descargas de Internet, procesamiento de datos, manejar transacciones de red, entre otros.

Un servicio se ejecuta con una prioridad mayor que una actividad que se encuentra en estado detenido, por lo que es menos probable que el sistema Android finalice un servicio. Sin embargo, se puede configurar un servicio para que este reinicie en caso de que sea finalizado debido a la carencia de recursos como memoria en el sistema. Un ejemplo del uso de este componente se observa en un reproductor de música, donde la música continúa en reproducción a pesar de que se ejecute una aplicación diferente.

Proveedores de contenido

Este componente administra el acceso a un repositorio de datos al que acceden diferentes aplicaciones [46]. En la mayoría de casos, los datos son almacenados en una base de datos SQlite, ficheros de texto, en la web o en otra ubicación a la que la aplicación pueda acceder. Un proveedor de contenido es implementado por una aplicación que desee que la información que almacena esté disponible al resto de aplicaciones del sistema. Un proveedor de contenidos presenta datos a aplicaciones externas utilizando tablas similares a las tablas de una base de datos relacional, y por lo general posee su propia interfaz de usuario para realizar consultas, editar el contenido, e incluso añadir o borrar contenido.

Este componente suele ser implementado por aplicaciones como el gestor de contactos o la agenda.

De los componentes mencionados, los servicios, actividades y receptores de mensajes son activados utilizando un mensaje conocido como *intent*. Un *intent* permite solicitar la realización de una acción, petición a la cual responden las aplicaciones dependiendo de su capacidad o no para realizar la acción solicitada. Un *intent* describe el tipo de acción a realizar para que el sistema busque un componente capaz de realizar dicha acción e iniciar con la misma. Al existir varios componentes que puedan realizar esa acción, el usuario indica el que desea utilizar [41].

El sistema puede encontrar componentes capaces de responder a un *intent* al comparar al mismo con filtros de *intents* especificados en el archivo de manifiesto de una aplicación. Al declarar una actividad en el archivo de manifiesto de una aplicación, se pueden añadir filtros *intent* que reflejen la capacidad de esa actividad para realizar una tarea y responder a los *intents* enviados por otras aplicaciones. Por ejemplo, al desarrollar una aplicación de correo electrónico que incluya una actividad capaz de redactar un nuevo correo, se puede utilizar un filtro para que ese componente responda a las *intents* que solicitan esa acción y el usuario pueda enviar un correo desde dicha actividad.

Por lo tanto, si un componente quiere invocar otro componente para llevar a cabo una tarea tiene que utilizar un *intent* especificando la tarea a realizar. Mediante los *intent-filters* declarados en el archivo de manifiesto de las aplicaciones, el sistema identifica los componentes capaces de realizar dicha tarea. Uno de esos componentes es invocado por la plataforma Android para llevar a cabo esa tarea, lo que significa que ninguno de los componentes conoce la existencia del otro, pero puede trabajar en conjunto para obtener el resultado deseado.

1.3.3.3. El archivo de manifiesto [47]

El fichero AndroidManifest.xml permite describir la funcionalidad y requerimientos de la aplicación al sistema Android. El objetivo principal del archivo de manifiesto es informar al sistema sobre los componentes que forman parte de la aplicación, como las actividades, servicios, entre otros. Este fichero se ubica en la raíz del directorio del proyecto asociado a dicha aplicación. Además de especificar los componentes de la aplicación, el archivo de manifiesto realiza entre otras cosas, las siguientes tareas:

- Indica las características de software y hardware que la aplicación necesita, como una cámara o Bluetooth.

- Identifica los permisos que requiere la aplicación por parte del usuario, como acceso a la información de contactos o Internet.
- Indica las bibliotecas que la aplicación requiere, como la biblioteca utilizada para trabajar con Google Maps, entre otras.
- Especifica el mínimo nivel de API que requiere la aplicación para ejecutarse (*minSdkVersion*), indica la API utilizada para compilar la aplicación (*targetSdkVersion*), entre otros. El nivel de API identifica la versión del *API framework* utilizado por las aplicaciones para interactuar con el sistema [48].

Cada versión de la plataforma Android está asociada a un único nivel de API como se observa en la Tabla 1.3, donde se muestran las principales versiones de la plataforma Android, su nombre código, nivel de API y la cantidad relativa de dispositivos que utilizan esas versiones de la plataforma.

Tabla 1.3. Nivel de API de las principales versiones de Android [49]

Versión	Nombre código	Nivel de API	Distribución
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.6%
4.3		18	0.5%
4.4	KitKat	19	7.8%
5.0	Lollipop	21	3.6%
5.1		22	14.7%
6.0	Marshmallow	23	21.6%
7.0	Nougat	24	19.0%
7.1		25	10.3%
8.0	Oreo	26	13.4%
8.1		27	5.8%
9	Pie	28	<0.1%

2. METODOLOGÍA

El presente trabajo de proyecto integrador se centra en el desarrollo de un prototipo de sistema domótico basado en la plataforma Arduino que permita generar alertas de seguridad al detectar una posible intrusión en la residencia y posibilite gestionar diferentes dispositivos e instalaciones, empleando un dispositivo móvil Android para monitorear y controlar el sistema desde dentro o fuera de la vivienda, a través del Internet o Bluetooth.

En este apartado se definen los requisitos, se especifica la estructura general del sistema domótico y se describe el desarrollo de la aplicación móvil para Android y el programa para el controlador Arduino. Además, se detalla el hardware empleado y se describe el ensamblaje del prototipo de sistema domótico.

2.1. Fase de diseño

En esta sección se detalla el proceso de diseño del sistema de control domótico, incluyendo una descripción de los recursos de hardware y software necesarios para desarrollar el prototipo.

2.1.1. Requisitos del sistema Domótico

El prototipo de sistema domótico busca mediante el uso de la tecnología satisfacer ciertas necesidades de los habitantes de una vivienda, como la reducción del trabajo doméstico y el apoyo a la seguridad. A continuación, se describen las características o funciones que debe tener el sistema domótico para satisfacer esas necesidades.

2.1.1.1. Análisis de la situación actual

El desarrollo tecnológico ha generado un gran impacto en la forma de vida de las personas en el mundo entero. El uso de un computador personal, un teléfono móvil o el acceso a Internet resultan imprescindibles al realizar las actividades diarias. Según la “Encuesta Nacional de Empleo, Desempleo y Subempleo” [50], elaborado por el INEC, el equipamiento tecnológico en el hogar ecuatoriano se ha incrementado en los últimos años.

Para el año 2017, el uso de computadores portátiles en los hogares se incrementó en 12,1 puntos, respecto al 2012. El acceso a Internet alcanzó un 37,2%, lo que supone un incremento de 14,7 puntos respecto al 2012. Además, para el mismo año, el 90,7% de los hogares del país poseía al menos un teléfono celular, 9 puntos por encima del valor registrado en el 2012. Respecto a la población de 5 años y más, la tenencia de un *smartphone* o teléfono inteligente alcanzó un 37,2 % al 2017, lo que evidencia un incremento de 31 puntos con relación al 2012.

A pesar de ello, parece haber una gran distancia entre el acceso a la tecnología, es decir los medios disponibles y las dificultades para su uso, además de un limitado manejo, lo que deja entrever una barrera enorme entre las personas y la tecnología.

Estas dificultades en el uso de la tecnología suponen una traba para el mercado de la edificación inteligente en nuestro país. En lugar de aparecer ante la opinión pública como herramientas que simplifican la vida de forma útil y confortable, las aplicaciones domóticas han sido consideradas, a menudo de manera injustificada, innecesarias, costosas y difíciles de usar. Es así que, en la mayoría de casos, la domótica ha sido relegada a un estrato social medio alto y alto [51, p. 10].

El interés en el desarrollo de la vivienda inteligente es algo común a los países desarrollados, en donde suele ser la normativa vigente la que obliga a la implementación de parte de las funcionalidades que abarca una edificación inteligente, para controlar por ejemplo, la eficiencia energética, lo que implica directa o indirectamente la utilización de sistemas de automatización, monitoreo y control en las edificaciones [1, pp. 10-13]. Sin embargo, desde un enfoque legal, en el Ecuador no existe una legislación en la que se trate directamente la automatización de viviendas o edificios, lo cual, sumado al desconocimiento de los beneficios que esta conlleva, ha dificultado el despliegue de este tipo de sistemas.

La mayoría de edificaciones en el país no disponen de sistemas domóticos instalados, por lo que surge la necesidad de contar con un sistema domótico en las viviendas, que comprometa un uso más eficiente de la tecnología y permita centralizar la gestión de la vivienda para ofrecer a las personas que habitan en ella mayor confort, seguridad, ahorro energético y económico.

La seguridad es la aplicación que más está ayudando a introducir sistemas domóticos en el hogar. El incremento de la seguridad en una vivienda no sólo repercute en la protección de los bienes particulares de sus habitantes, sino también en su protección personal, algo a lo que contribuirá el presente proyecto. El uso de la tecnología posibilita la utilización de sistemas de alarma cada vez más sofisticados en la vivienda. El monitoreo de las puertas y ventanas, la generación de alertas, la simulación de presencia y la vigilancia con una cámara, son funciones del sistema domótico propuesto que proporcionarán mayor seguridad a los habitantes de la vivienda y contribuirá a una mejor calidad de vida.

Además, la vivienda domótica brinda una mayor comodidad a sus habitantes al evitarles la realización de tareas repetitivas y ofreciendo servicios para mejorar su confort. La gestión centralizada de las instalaciones al encender o apagar las luces, la radio y medir la

temperatura proporcionarán con este sistema domótico un aumento del confort en los habitantes de la vivienda.

La escasez de energía y el cambio climático son considerados como uno de los problemas más importantes a nivel mundial. Según datos de la Agencia de Regulación y Control de Electricidad, los ecuatorianos cada vez consumen más energía [52]. Después del gas licuado de petróleo (GLP), la electricidad es el tipo de energía más utilizada por el sector residencial. En el año 2016, el consumo de energía del sector residencial suponía ya un 32,23 % del consumo total de energía del país [53, p. 79].

Este sistema domótico posibilitará, por ejemplo, conocer el estado (encendido/apagado) de las luces del hogar aun estando fuera de la vivienda. Esto permitirá al usuario apagar las luces encendidas, ya sea que se encuentre al interior o fuera de la edificación. Evidentemente, esta optimización del consumo de recursos, como la energía eléctrica, redundará en un ahorro económico para los habitantes de la vivienda [6, p. 200]. Por lo tanto, esta solución domótica contribuirá al ahorro de energía y económico, sobre todo en el sector residencial.

La gestión del sistema se realizará desde un dispositivo móvil. Para el primer trimestre del 2018, se reporta un aproximado de 383 millones de dispositivos móviles vendidos a nivel mundial, de los cuales, un 85,9% emplea el sistema operativo Android [54]. Considerando que Android es una de las plataformas más utilizadas, la aplicación móvil será desarrollada para este sistema operativo.

En definitiva, cada contexto social requiere un entorno tecnológico distinto. La realidad social de los hogares en el país es muy diversa, por lo que es necesario ofrecer soluciones tecnológicas también diversas que se adapten a cada tipo de hogar.

2.1.1.2. Definición de requisitos

A partir del análisis realizado en el apartado anterior, se plantea un prototipo de sistema domótico que cumpla con los siguientes requisitos:

- **Encendido y apagado de luminarias:** Brindar un mecanismo para gestionar de forma remota el encendido y apagado de luminarias utilizando un dispositivo móvil.
- **Encendido y apagado de la radio:** Proveer un mecanismo para encender o apagar de forma remota una radio empleando un dispositivo móvil.
- **Medición de temperatura:** Brindar un mecanismo para informar la temperatura al interior de la vivienda.

- **Seguridad en los accesos de la vivienda:** Brindar un mecanismo para informar si la puerta o las ventanas se encuentran abiertas o cerradas.
- **Detección de movimiento:** Brindar un mecanismo para informar la detección de movimiento al interior de la vivienda.
- **Simulación de presencia:** Proveer un mecanismo para encender y apagar las luces y la radio de forma automática que permita simular presencia.
- **Botón de pánico:** Proveer un mecanismo para generar una alerta sonora. La activación del botón de pánico se realizará desde la aplicación móvil o utilizando un pulsador.
- **Monitoreo con cámara de video:** Implementar un mecanismo que permita capturar fotografías al detectar movimiento.
- **Aplicación Android para gestionar el sistema**: Se plantea como requisito la gestión del sistema domótico desde una aplicación móvil para el sistema operativo Android.
- **Gestión del sistema de forma remota:** Proveer un mecanismo que permita gestionar el sistema domótico al encontrase dentro o fuera de la vivienda, utilizando Bluetooth o el Internet.

2.1.1.3. Especificación de los requisitos funcionales

Los requisitos funcionales describen el comportamiento deseado del sistema domótico. Este comportamiento puede ser expresado como servicios, tareas o funciones que debe brindar el sistema.

Para describir los requisitos funcionales del sistema domótico se emplean casos de uso. Un caso de uso representa una función o una acción llevada a cabo por el sistema, indica las tareas que realiza el sistema, sin especificar como las realiza, además, permite describir paso a paso el proceso que sigue un usuario para conseguir una meta.

La interacción entre el usuario y los diferentes casos de uso del sistema se expresa utilizando los diagramas de las Figuras 2.1 y 2.2.

En el diagrama de casos de uso de la Figura 2.1 se muestran las principales funciones del prototipo domótico, mientras que en el diagrama de la Figura 2.2, se incluyen las funciones del sistema domótico orientadas a la generación de alertas de seguridad.

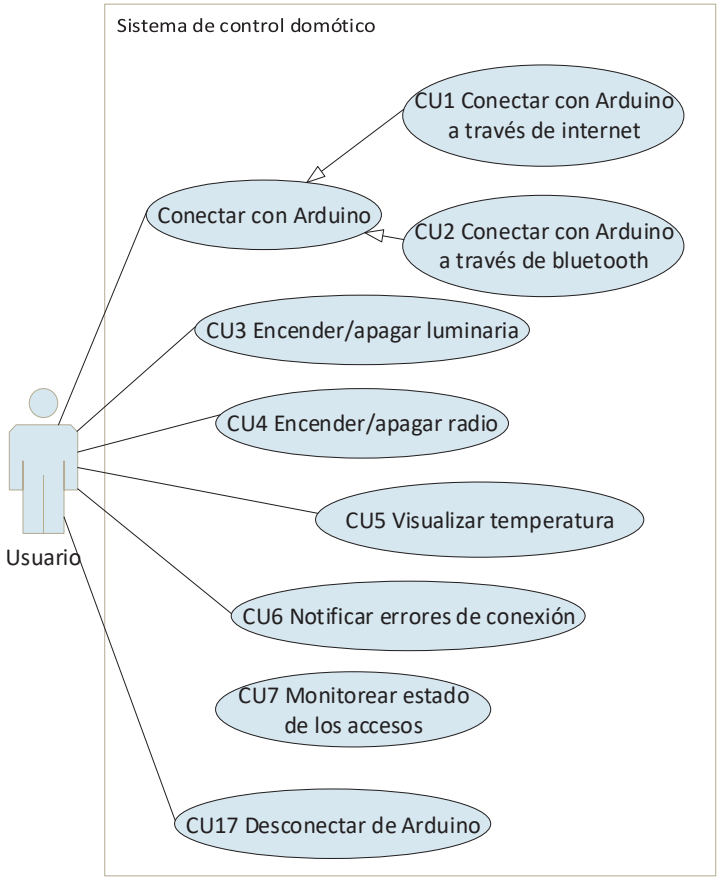

Figura 2.1. Diagrama de casos de uso del sistema

La flecha representa una relación de generalización. La flecha va desde el caso de uso especializado al caso de uso más general, donde el caso de uso especializado representa una forma particular de conseguir la meta especificada por el caso de uso general. Es decir, los casos de uso especializados describen diferentes formas en que el sistema domótico puede conseguir una misma meta. Además, dado que los casos de uso especializados heredan las metas y los actores del caso de uso general, no es necesario definir escenarios para el caso de uso general.

Una relación del tipo *include* muestra que un caso de uso describe una secuencia de eventos que forman parte de otro caso de uso.

La descripción de los casos de uso incluye los siguientes elementos:

- **Id:** Identificador del caso de uso, compuesto por las letras CU y un número.
- **Nombre:** Nombre del caso de uso, indica el enfoque del caso de uso.
- **Actor:** Representa un objeto, persona o hardware que interactúa con el sistema de control domótico.

- **Prioridad:** La prioridad asignada a cada caso de uso puede ser baja, media o alta.
- **Descripción:** Una breve descripción del caso de uso.
- **Precondición:** Las condiciones que deben cumplirse antes de que este caso de uso pueda iniciar.
- **Flujo normal:** Muestra la secuencia de acciones que realiza un actor para cumplir con la meta planteada por el caso de uso. Describe de forma clara como responde el sistema ante cada acción del usuario.
- **Flujo alternativo:** Muestra la secuencia de eventos alternativa al producirse errores o excepciones que impiden cumplir con la meta planteada.

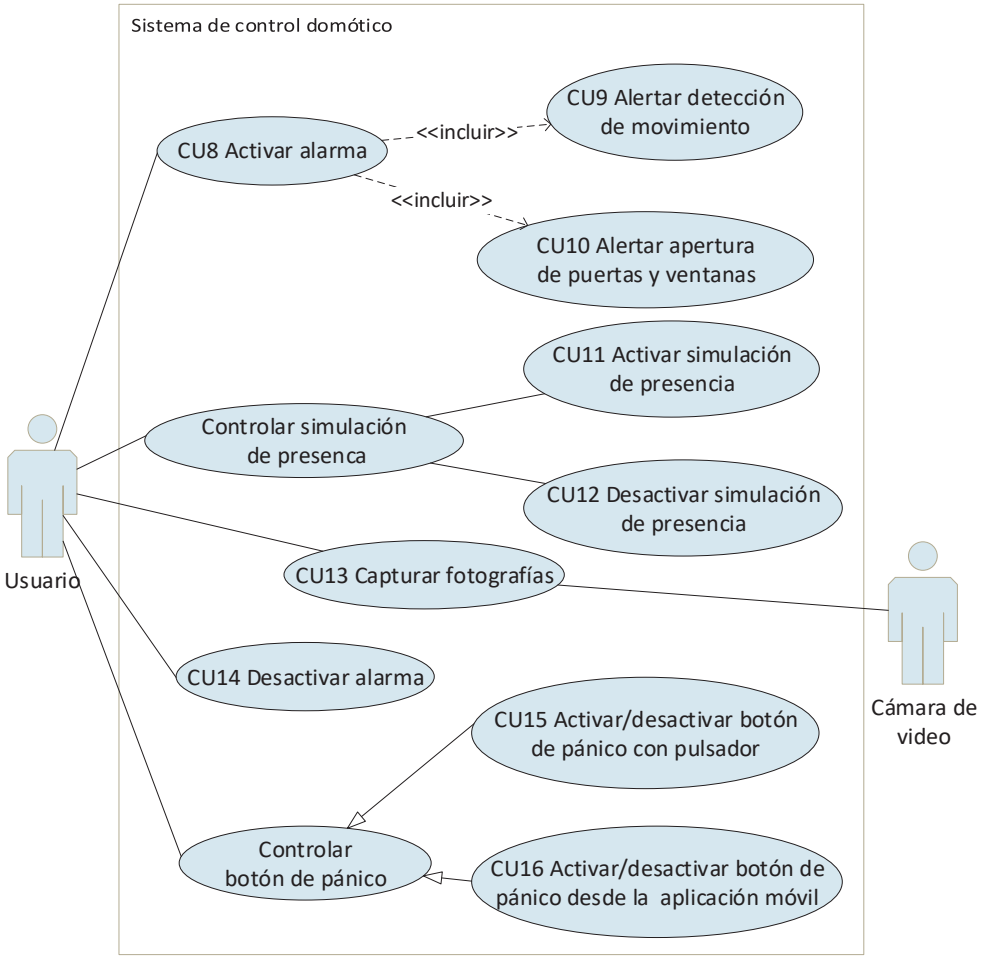

Figura 2.2. Diagrama de casos de uso orientados a generar alertas de seguridad

Las Tablas 2.1 a 2.17 detallan los requisitos funcionales del sistema domótico a través de casos de uso.

Tabla 2.1. Conectar con Arduino a través de Internet

Id: CU1	**Nombre:** Conectar con Arduino a través de Internet
Actor: Usuario	**Prioridad:** Alta
Descripción: El usuario accede a la aplicación móvil, ingresa un nombre de usuario, contraseña y dirección IP del controlador. Se establece una conexión entre la aplicación y el controlador del sistema domótico a través del Internet.	
Precondición: Ninguno	
Flujo Normal 1. El usuario accede a la aplicación móvil y se muestra la interfaz de conexión. 2. El usuario selecciona “Internet” como método de conexión, ingresa el nombre de usuario, contraseña, dirección IP y puerto del controlador. 3. El usuario presiona “Conectar”. Mientras se verifica la identidad del usuario, la aplicación muestra un mensaje indicando que se está conectando con el controlador. 4. Una vez autenticado el usuario, la aplicación muestra la interfaz de consola desde la cual se gestiona el sistema domótico.	
Flujo Alternativo 4.1 El proceso de autenticación entre la aplicación móvil y el controlador falla, por lo tanto, la aplicación muestra un mensaje indicando que el nombre de usuario o la contraseña son incorrectos. 4.2 Si se pierde la conectividad a Internet o se produce un error de conexión entre la aplicación y el controlador, la aplicación indica que no es posible conectarse al controlador.	

Tabla 2.2. Conectar con Arduino a través de Bluetooth

Id: CU2	**Nombre:** Conectar con Arduino a través de Bluetooth
Actor: Usuario	**Prioridad:** Alta
Descripción: El usuario accede a la aplicación móvil, ingresa un nombre de usuario y contraseña. Se establece una conexión entre la aplicación y el controlador del sistema domótico a través de Bluetooth.	
Precondición: El dispositivo Android y el módulo Bluetooth de Arduino deben estar emparejados.	
Flujo Normal 1. El usuario accede a la aplicación móvil y se muestra la interfaz de conexión. 2. El usuario selecciona “Bluetooth” como método de conexión, ingresa el nombre de usuario y contraseña. 3. El usuario presiona “Conectar”. Mientras se verifica la identidad del usuario, la aplicación muestra un mensaje indicando que se está conectando con el controlador. 4. Una vez autenticado el usuario, la aplicación muestra la interfaz de consola desde la cual se gestiona el sistema domótico.	
Flujo Alternativo 4.1 El proceso de autenticación entre la aplicación móvil y el controlador falla, por lo tanto, la aplicación muestra un mensaje indicando que el nombre de usuario o la contraseña son incorrectos. 4.2 Si se produce un error en la conexión entre la aplicación y el controlador, la aplicación indica que no es posible conectarse al controlador.	

Tabla 2.3. Notificar errores de conexión

Id: CU6	**Nombre:** Notificar errores de red
Actor: Usuario	**Prioridad:** Media
Descripción: Se notifica al usuario cuando la comunicación con el controlador es interrumpida.	
Precondición: Conexión con el controlador establecida.	
Flujo Normal 1. El dispositivo móvil verifica el estado de la conexión con el controlador de forma periódica. 2. Al detectar que la conexión entre la aplicación móvil y el controlador permanece establecida no se genera ninguna alerta.	
Flujo Alternativo 2.1 Si se detecta que la conexión se ha cerrado, se genera un mensaje que indique al usuario que la comunicación entre la aplicación móvil y el controlador se ha interrumpido. El usuario puede regresar a la interfaz de conexión para establecer la conexión nuevamente usando Internet o Bluetooth, para ello selecciona la opción "Conexión" del menú de opciones de la aplicación móvil.	

Tabla 2.4. Visualizar temperatura

Id: CU5	**Nombre:** Visualizar temperatura
Actor: Usuario	**Prioridad:** Media
Descripción: El usuario visualiza la temperatura en la vivienda desde la aplicación móvil.	
Precondición: Conexión con el controlador establecida.	
Flujo Normal 1. El usuario observa la temperatura de la vivienda en la interfaz gráfica. El sistema domótico actualiza la temperatura mostrada de forma periódica.	
Flujo Alternativo 1.1 Al no recibir un mensaje de actualización de temperatura desde el controlador, se indica al usuario que se ha producido una falla en la conexión.	

Tabla 2.5. Encender/apagar luminaria

Id: CU3	**Nombre:** Encender/apagar luminaria
Actor: Usuario	**Prioridad:** Alta
Descripción: El usuario enciende o apaga una luminaria utilizando la interfaz visual de la aplicación móvil.	
Precondición: Conexión con el controlador establecida. La función de simulación de presencia permanece desactivada.	
Flujo Normal 1. El usuario enciende o apaga una luminaria a través de la interfaz visual. 2. Se notifica de forma gráfica al usuario el cambio de estado de la luminaria.	
Flujo Alternativo 2.1 Se muestra un mensaje indicando que no es posible realizar esa operación debido a una falla en la conexión entre la aplicación y el controlador.	

Tabla 2.6. Encender/apagar radio

Id: CU4	**Nombre:** Encender/apagar radio
Actor: Usuario	**Prioridad:** Alta
Descripción: El usuario enciende o apaga la radio utilizando la interfaz visual de la aplicación móvil.	
Precondición: Conexión con el controlador establecida. La función de simulación de presencia permanece desactivada.	
Flujo Normal 1. El usuario enciende o apaga la radio desde la interfaz visual de la aplicación móvil. 2. Se notifica de forma gráfica al usuario el cambio de estado de la radio.	
Flujo Alternativo 2.1 Se muestra un mensaje indicando que no es posible realizar esa operación debido a una falla en la conexión entre la aplicación y el controlador.	

Tabla 2.7. Activar alarma

Id: CU8	**Nombre:** Activar alarma
Actor: Usuario	**Prioridad:** Alta
Descripción: El usuario activa el sistema de alarma, lo que activa la detección de movimiento y la generación de alertas por apertura de puertas y ventanas.	
Precondición: Conexión con el controlador establecida. Los accesos de la vivienda (puertas y ventanas) deben estar cerrados.	
Flujo Normal 1. El usuario activa la alarma a través de la interfaz visual. 2. El sistema domótico activa la generación de alertas por detección de movimiento o apertura de puertas y ventanas.	
Flujo Alternativo 2.1 Se muestra un mensaje indicando que no es posible realizar esa operación debido a una falla en la conexión entre la aplicación y el controlador.	

Tabla 2.8. Desactivar alarma

Id: CU14	**Nombre:** Desactivar alarma
Actor: Usuario	**Prioridad:** Alta
Descripción: El usuario desactiva el sistema de alarma, lo que desactiva la detección de movimiento y las alertas por detección de apertura/cierre de puertas y ventanas.	
Precondición: Conexión con el controlador establecida.	
Flujo Normal 1. El usuario desactiva la alarma a través de la interfaz visual. 2. El sistema domótico desactiva la detección de movimiento. 3. El sistema desactiva la generación de alertas por apertura o cierre de puertas y ventanas. 4. Si la simulación de presencia permanecía activa, es desactivada.	
Flujo Alternativo 1.1 Se muestra un mensaje indicando que no es posible desactivar la alarma debido a una falla en la conexión entre la aplicación y el controlador.	

Tabla 2.9. Activar simulación de presencia

Id: CU11	**Nombre:** Activar simulación de presencia
Actor: Usuario	**Prioridad:** Alta
Descripción: Al activar esta función las luces y la radio se encienden y apagan de forma automática.	
Precondición: La alarma permanece activa.	
Flujo Normal 1. El usuario activa la simulación de presencia. 2. El sistema enciende y apaga las luces y la radio sin la intervención del usuario, e impide el control de esos dispositivos al usuario mientras esta función permanezca activada. 3. La aplicación móvil muestra cada cambio en el estado de las luces y la radio.	
Flujo Alternativo 3.1 No es posible notificar al usuario el cambio en el estado de los elementos debido a un fallo en la conexión entre la aplicación móvil y el controlador. Se notifica al usuario del error en la conexión.	

Tabla 2.10. Desactivar simulación de presencia

Id: CU12	**Nombre:** Desactivar simulación de presencia
Actor: Usuario	**Prioridad:** Alta
Descripción: Se desactiva el encendido y apagado automático de las luces y la radio.	
Precondición: La alarma permanece activa.	
Flujo Normal 2. El usuario desactiva la simulación de presencia 3. El sistema apaga los elementos que estaban encendidos y devuelve el control de las luces y la radio al usuario.	
Flujo Alternativo 1.1 Se muestra un mensaje indicando que no es posible desactivar esta función debido a una falla en la conexión entre la aplicación y el controlador.	

Tabla 2.11. Monitorear estado de los accesos

Id: CU7	**Nombre:** Monitorear estado de los accesos
Actor: Usuario	**Prioridad:** Alta
Descripción: La aplicación muestra el estado de los accesos de la vivienda, es decir, indica si las puertas y ventanas permanecen abiertas o cerradas.	
Precondición: Conexión con el controlador establecida.	
Flujo Normal 1. La interfaz de usuario indica si una puerta o ventana permanece abierta o cerrada.	
Flujo Alternativo 1.1 No es posible conocer el estado actual de las puertas o ventanas debido a un fallo en la conexión entre la aplicación móvil y el controlador.	

Tabla 2.12. Activar/desactivar botón de pánico con pulsador

Id: CU15	**Nombre:** Activar/desactivar botón de pánico con pulsador
Actor: Usuario	**Prioridad:** Media
Descripción: El usuario activa o desactiva la alerta sonora al activar o desactivar el botón de pánico respectivamente.	
Precondición: Conexión con el controlador establecida.	
Flujo Normal 1. El usuario presiona un pulsador que permite activar o desactivar el botón de pánico. 2. El sistema domótico activa o desactiva la sirena utilizada para generar la alerta sonora. 3. La interfaz visual de la aplicación móvil muestra el estado del botón de pánico, activado o desactivado.	
Flujo Alternativo 3.1 La interfaz visual no muestra el estado actual del botón de pánico debido a una falla en la conexión entre la aplicación y el controlador.	

Tabla 2.13. Activar/desactivar botón de pánico desde la aplicación móvil

Id: CU16	**Nombre:** Activar/desactivar botón de pánico desde la aplicación móvil
Actor: Usuario	**Prioridad:** Alta
Descripción: El usuario activa o desactiva la alerta sonora al activar o desactivar el botón de pánico respectivamente.	
Precondición: Conexión con el controlador establecida.	
Flujo Normal 1. El usuario activa o desactiva el botón de pánico desde la interfaz visual de la aplicación móvil. 2. El sistema domótico activa o desactiva la sirena utilizada para generar la alerta sonora.	
Flujo Alternativo 2.1 Se muestra un mensaje indicando que no es posible activar o desactivar el botón de pánico debido a una falla en la conexión entre la aplicación y el controlador.	

Tabla 2.14. Alertar detección de movimiento

Id: CU9	**Nombre:** Alertar detección de movimiento
Actor: Usuario	**Prioridad:** Alta
Descripción: Se alerta al usuario al detectar movimiento en la vivienda.	
Precondición: La alarma está activada.	
Flujo Normal 1. Al detectarse movimiento, la aplicación móvil muestra un mensaje al usuario alertando el suceso.	
Flujo Alternativo 1.1 No es posible alertar la detección de movimiento al usuario al producirse un fallo en la conexión entre la aplicación móvil y el controlador. En ese caso se notifica al usuario del error en la conexión.	

Tabla 2.15. Alertar apertura de puertas y ventanas

Id: CU10	**Nombre:** Alertar apertura de puertas y ventanas
Actor: Usuario	**Prioridad:** Alta
Descripción: Se alerta al usuario cuando una puerta o ventana se abre.	
Precondición: La alarma permanece activa.	
Flujo Normal 1. Se alerta al usuario al detectar que una puerta o ventana se abre. 2. Se genera una alerta sonora en la vivienda mediante una sirena.	
Flujo Alternativo 1.1 No es posible alertar al usuario debido a un fallo en la conexión entre la aplicación móvil y el controlador. En ese caso se notifica al usuario del error en la conexión.	

Tabla 2.16. Capturar fotografías

Id: CU13	**Nombre:** Capturar fotografías
Actor: Usuario, cámara de video	**Prioridad:** Alta
Descripción: Se toma una fotografía al detectar movimiento en una estancia de la vivienda.	
Precondición: Especificar una dirección de correo electrónico para el envío de las imágenes capturadas.	
Flujo Normal 1. Al detectar movimiento en la vivienda se toma una fotografía. 2. Se envía la imagen capturada al correo electrónico del usuario.	
Flujo Alternativo 2.1 La cámara de video pierde conexión a Internet, lo que impide el envío de las imágenes capturadas al detectar movimiento.	

Tabla 2.17. Desconectar de Arduino

Id: CU17	**Nombre:** Desconectar de Arduino
Actor: Usuario	**Prioridad:** Alta
Descripción: La conexión entre la aplicación móvil y el controlador se cierra.	
Precondición: Ninguno	
Flujo Normal 1. El usuario accede al menú de opciones en la aplicación móvil. 2. El usuario selecciona la opción “Salir”. 3. La aplicación se cierra en el dispositivo móvil. 4. El sistema sigue ejecutando las funciones indicadas por el usuario a pesar de no existir un usuario que gestione el sistema.	

2.1.1.4. Especificación de requisitos no funcionales [55]

Un requisito funcional permite describir las tareas que realiza el sistema, mientras que un requisito no funcional, define restricciones de como el sistema cumplirá con dichas tareas.

A continuación, se describen los requisitos no funcionales del prototipo.

- **Accesibilidad:** La accesibilidad hace referencia a la habilidad de acceder y beneficiarse de un sistema [56]. Un sistema es accesible cuando puede ser utilizado por personas que tengan algún tipo de discapacidad.

  El prototipo de sistema domótico es accesible a personas con impedimento de movilidad que presenten discapacidad para caminar. Por tal motivo, se propone la utilización de una aplicación móvil para la gestión del sistema, que permita al usuario acceder a las funcionalidades del prototipo domótico desde cualquier lugar de la vivienda, y por lo tanto, permita al usuario realizar tareas como el encendido o apagado de luminarias del hogar, sin necesidad de desplazarse.

- **Soporte:** Se requiere contar con documentación del sistema domótico que facilite la operación del mismo.

2.1.2. Estructura General del Sistema Domótico

La Figura 2.3 muestra la estructura del prototipo de sistema domótico.

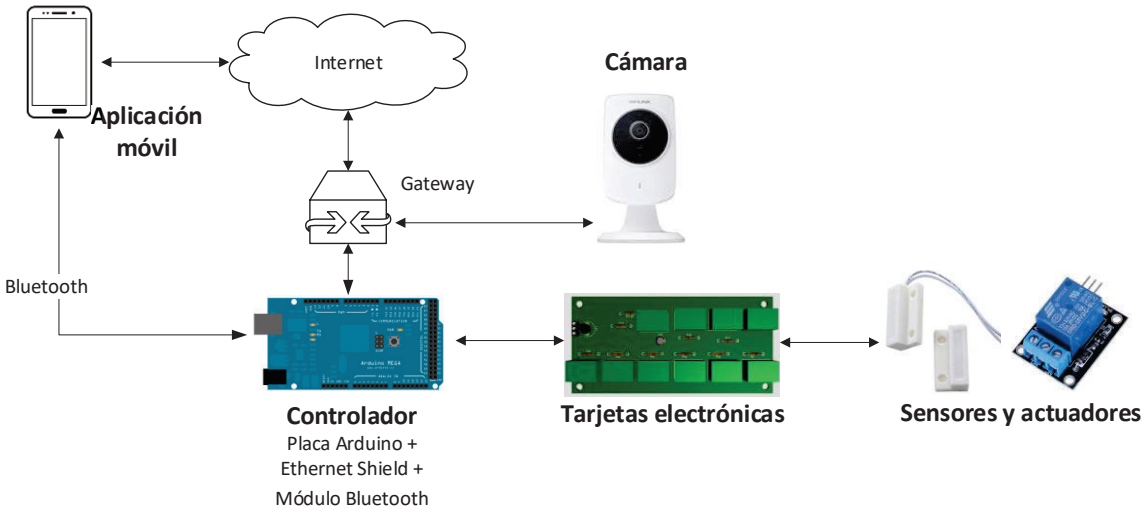

Figura 2.3. Estructura del sistema domótico

El prototipo de sistema domótico propuesto contará con los siguientes componentes:

- **Controlador:** Es el dispositivo encargado de recibir datos de los sensores a través de tarjetas electrónicas que permiten adaptar las señales que estos generan. Además, el controlador se encarga de enviar órdenes a los actuadores para llevar a cabo una acción. Se emplea como controlador la placa Arduino Mega 2560, descrita en el capítulo 1.

- **Sensores:** Dispositivos que recolectan información para cumplir con los requisitos del prototipo domótico.
- **Actuadores:** Dispositivos que realizan acciones para dar solución a los problemas encontrados en una vivienda.
- **Tarjetas electrónicas:** Circuitos electrónicos que permiten adaptar los niveles de voltaje al rango de trabajo del controlador.
- **Aplicación móvil:** Aplicación para el sistema operativo Android utilizada para gestionar el sistema domótico.
- **Cámara:** Dispositivo empleado para la captura de imágenes en la vivienda.

2.1.2.1. Conexión sensores-controlador

El controlador Arduino se conectará de forma cableada a los sensores. Cada sensor estará asociado a un determinado pin en la placa. Para adaptar los datos generados por los sensores de tal forma que puedan ser procesados por Arduino, se utilizan circuitos electrónicos para cada tipo de sensor, como se muestra en la Figura 2.4.

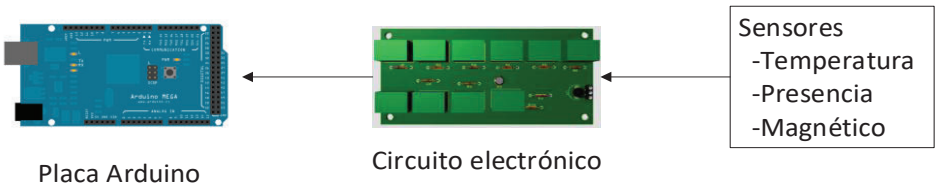

Figura 2.4. Conexión Arduino-Sensores

2.1.2.2. Conexión actuadores-controlador

Los actuadores se conectarán de forma cableada con los pines de la placa Arduino. Cada actuador está asociado a un pin en la placa, la cual se encarga de enviar órdenes a los actuadores a través de un circuito electrónico, como se muestra en la Figura 2.5.

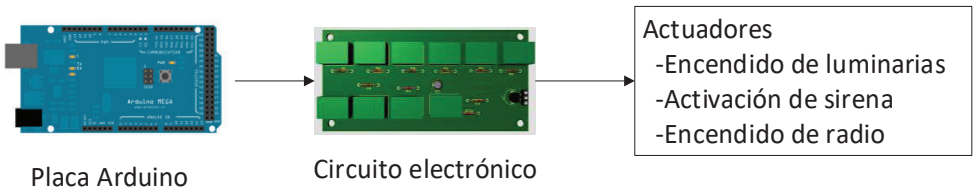

Figura 2.5. Conexión Arduino-Actuadores

2.1.3. Identificación de recursos necesarios

A continuación, se describen los principales elementos utilizados al desarrollar el prototipo de sistema domótico y los servicios domóticos a brindar en cada espacio físico de la vivienda.

2.1.3.1. Selección del hardware

Hardware para control de luminarias

Las luminarias en una vivienda funcionan con 120 V, lo que hace necesario la utilización de relés para encender o apagar dichos dispositivos. El principio de funcionamiento de un relé se describe en el Capítulo 1.

Se selecciona un relé del tipo polo simple doble salida (*Single-Pole Double-Throw*), como se muestra en la Figura 2.6.

Figura 2.6. Relé de polo simple doble salida

Al ser de polo simple, permite controlar solo un circuito eléctrico. Al ser de dos salidas, la entrada C puede ser conectada a una de dos salidas, A o B. Los dos pines restantes, denominados 1 y 2, permiten energizar la bobina [57].

En la Tabla 2.18, se muestran las principales características eléctricas del relé seleccionado. El voltaje en la bobina representa el voltaje en los pines 1 y 2, requerido para activar el relé y cambiar la posición del interruptor de B hacia A. La carga máxima representa el voltaje y corriente que el relé puede soportar en sus terminales A, B y C. La hoja técnica de este dispositivo se muestra en el Anexo IV.

Tabla 2.18. Características eléctricas del relé empleado [58]

Característica	Descripción
Voltaje en la bobina	5 V DC
Corriente en la bobina	70 mA
Carga máxima AC	10 A a 250/125 V AC
Carga máxima DC	10 A a 30 V DC

Hardware para sensado de apertura/cierre de puertas y ventanas

Para conocer si una puerta o ventana permanece abierta o cerrada se utiliza un contacto magnético. El contacto magnético seleccionado consta de dos partes, una móvil y una fija, como se muestra en la Figura 2.7.

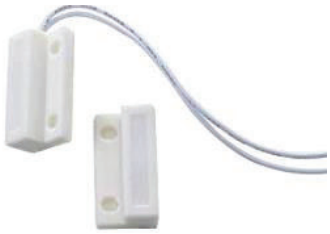

Figura 2.7. Contacto magnético

Cuando ambas partes permanecen juntas, el contacto magnético, que funciona como un interruptor, cierra el interruptor y permite el paso de corriente a través de sus dos terminales. Si ambas partes del contacto magnético se separan, se abre el interruptor y se detiene el paso de corriente a través de sus terminales.

Hardware para detección de movimiento

Para la detección de movimiento se emplea un sensor PIR HC-SR501, como se muestra en la Figura 2.8. Para distinguir la presencia, el sensor PIR detecta la diferencia de calor emitida por una persona y su alrededor. Al detectar una variación, genera una señal eléctrica que es procesada por el controlador Arduino. Su hoja técnica se incluye en el Anexo V.

Figura 2.8. Sensor PIR

Hardware para control de la sirena

Se emplea un *buzzer* para representar una sirena y generar una alerta sonora en la vivienda. El *buzzer* funciona con un voltaje de 3 a 24 V, y es activado a través de un circuito cuyo principal componente es el relé descrito anteriormente.

Hardware para botón de pánico

Para la activación del botón de pánico se utiliza un pulsador, como se observa en la Figura 2.9. Cuando este es presionado permite conectar dos puntos en un circuito electrónico para enviar una señal al controlador.

Figura 2.9. Pulsador

Hardware para sensado de temperatura [59]

Para la medición de temperatura en la vivienda se utiliza un sensor LM35, como se indica en la Figura 2.10. Este circuito integrado entrega una tensión de salida proporcional a la temperatura, genera una salida de 10 mV por cada grado centígrado, lo que permite obtener la temperatura al medir el nivel de la señal generada por el sensor.

Figura 2.10. Sensor de temperatura

A continuación, se describen las principales características de este sensor:

- Posee una precisión de 0.5 °C (a 25 °C).
- Medición de temperatura entre -55 a +150 °C.
- Voltaje de alimentación de 4 a 30 V.
- Consumo de corriente inferior a los 60 uA.
- Voltaje de salida linealmente proporcional a la temperatura en grados centígrados.

La hoja técnica de este dispositivo se muestra en el Anexo VI.

Hardware para monitoreo con cámara de video

Para la captura de imágenes en la vivienda se utiliza una cámara IP, cuyas características se muestran en la Tabla 2.19.

Tabla 2.19. Características cámara D-Link [60]

Característica	Descripción
Marca	D-link
Modelo	DCS-932L
Interfaz de red	*Wireless* (802.11 b/g/n) y puerto Ethernet (10/100 Base TX - Fast Ethernet)
Visión nocturna	Infrarrojo
Resolución	Hasta 640x480
Alimentación	5 V – 1 A
Dimensiones	65.8x65x126 mm
Detección de movimiento y envío de alertas por *e-mail*	
	

La hoja técnica de este dispositivo se muestra en el Anexo VII.

Hardware para las tarjetas electrónicas

A continuación, se describen los principales elementos utilizados para desarrollar los circuitos electrónicos que permiten al controlador Arduino interactuar con los diferentes sensores y actuadores.

- **Capacitores** [61]: Un capacitor o condensador es un dispositivo capaz de almacenar energía eléctrica. En la Figura 2.11 se observa la imagen de un capacitor.

Figura 2.11. Capacitor

Está compuesto por dos objetos conductores, por lo general, hojas o placas, colocadas una cerca de la otra sin que entren en contacto. Estas placas están separadas por un material aislante conocido como dieléctrico. Cuando un

condensador es conectado a un circuito, circula corriente por dicho dispositivo, lo que causa que este acumule carga entre sus placas. La corriente deja de circular por el circuito cuando el condensador se encuentra completamente cargado.

Los condensadores almacenan carga que puede ser liberada posteriormente. La cantidad de carga eléctrica que un condensador puede almacenar es conocida como capacitancia.

- **Diodo emisor de luz** [62]**:** Un diodo emisor de luz o led es un elemento semiconductor que emite luz cuando circula corriente a través del mismo, del ánodo al cátodo. Es decir, transforma energía eléctrica en radiación electromagnética (luz).

  Un LED está asociado a una caída de voltaje característica, que depende del material semiconductor utilizado, y que suele variar entre 1.5 y 4 V. El Led se indica en la Figura 2.12.

Figura 2.12. Led

- **Resistencias** [63]**:** Un resistor es un dispositivo electrónico empleado para generar resistencia al paso de corriente eléctrica entre dos puntos en un circuito.

  Para medir la resistencia eléctrica se utiliza como unidad el ohmio. La resistencia se define a través de la ley de Ohm como $R = V/I$. Un resistor se muestra en la Figura 2.13.

Figura 2.13. Resistor

2.1.3.2. Identificación del espacio físico de la vivienda

Para determinar los elementos requeridos al desarrollar el sistema domótico, es necesario conocer el espacio físico de la vivienda donde se implementará dicho prototipo.

La Figura 2.14 muestra la distribución de las diferentes estancias de la vivienda de una sola planta.

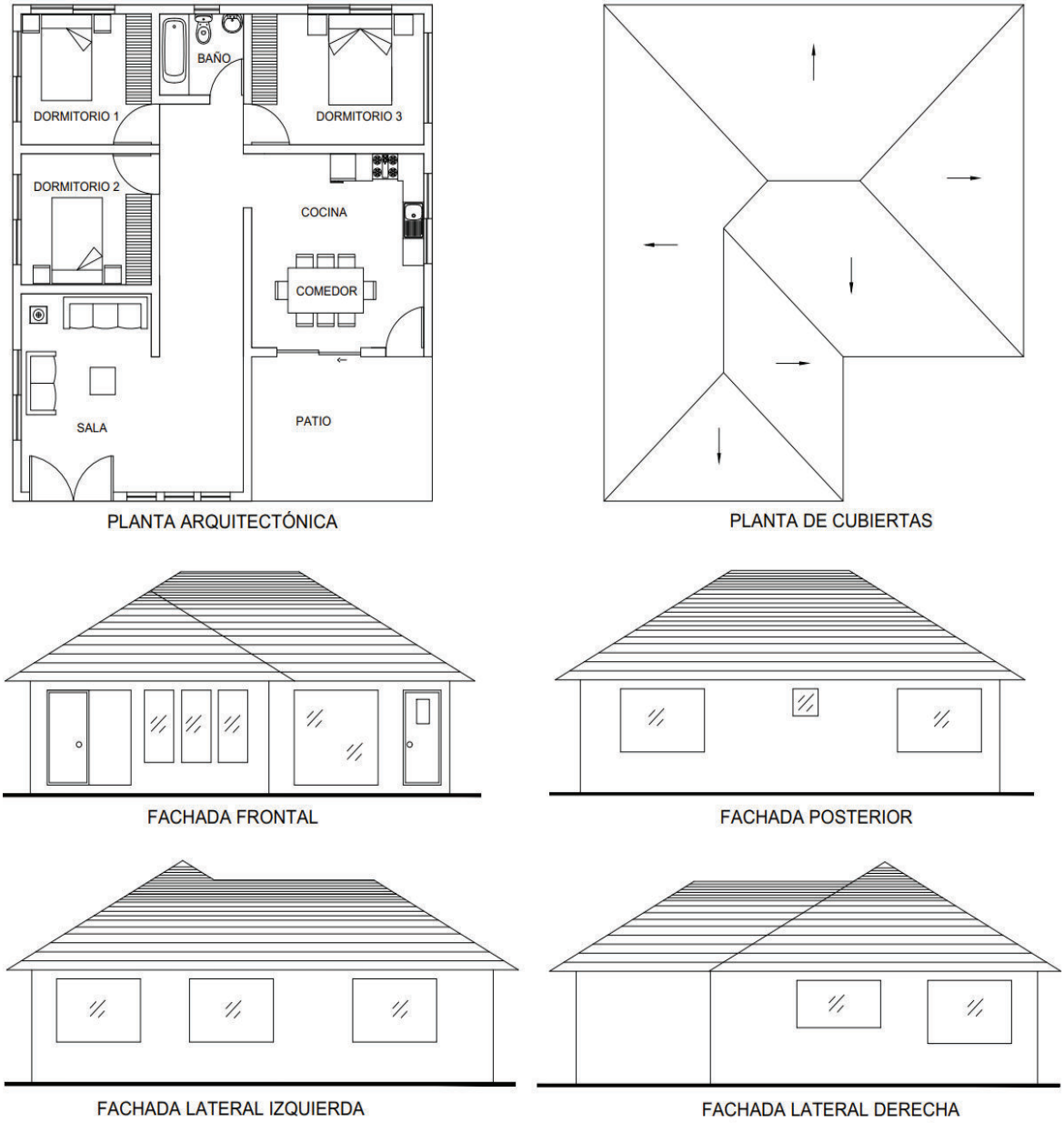

Figura 2.14. Planos para el prototipo de sistema domótico

2.1.3.3. Aplicaciones domóticas en cada espacio físico

En la Tabla 2.20, se describen los servicios domóticos a brindar en cada espacio físico de la vivienda.

Para el diseño del prototipo se consideran: 3 dormitorios, un comedor/cocina, una sala, un baño y el acceso principal, en donde se ubicará el controlador, además de los diferentes sensores y actuadores del prototipo de sistema domótico.

2.1.4. Programación del controlador Arduino

A continuación, se describe el programa desarrollado para el controlador Arduino, el cual se encarga del encendido/apagado de luminarias y la radio, medición de temperatura, monitoreo de los accesos de la vivienda, detección de movimiento, simulación de presencia, gestión del botón de pánico y comunicación con la aplicación móvil para Android.

Tabla 2.20. Servicio domótico en cada dependencia

Espacio físico	Aplicación domótica
Dormitorios 1,2 y 3	- Encendido/apagado luminaria - Monitoreo del estado de la ventana - Simulación de presencia
Sala	- Encendido/apagado luminaria - Control del tomacorriente - Monitoreo del estado de la ventana y puerta - Detección de movimiento - Medición de temperatura - Simulación de presencia - Botón de pánico
Comedor/cocina	- Encendido/apagado luminaria - Monitoreo del estado de la ventana - Simulación de presencia
Baño	- Encendido/apagado luminaria - Simulación de presencia
Acceso principal	- Monitoreo con cámara

2.1.4.1. Programación en Arduino

La programación del controlador Arduino se realiza en base a una adaptación del lenguaje C++ que facilita el desarrollo de programas para esta plataforma.

Como se observa en el Código 2.1, la estructura básica de un programa para el microcontrolador es bastante simple y está compuesto por lo menos de las funciones: `init()`, `setup()` y `loop()`.

```
int main(){
init();
setup();
for (;;) {
loop();}
return 0;}
```

Código 2.1. Estructura de programa para Arduino

Una función es un bloque de código que contiene un grupo de instrucciones que se ejecutan al llamar a la función.

La función `init()` inicializa el contador del *timer*, el conversor AD y la comunicación serial.

La función `setup()` se ejecuta una sola vez, razón por la que incluye las instrucciones de configuración del modo de trabajo de los pines, configuración de la comunicación serie, el estado inicial de los puertos de salida entre otros.

La función `loop()` contiene el programa que será ejecutado en un bucle por el microcontrolador. En esta función se suele realizar la lectura de entradas, activación de salidas, entre otros [64, pp. 7-13].

Sin embargo, Arduino simplifica la creación de programas al hacer que sea necesario implementar únicamente las funciones `setup()` y `loop()`, como se observa el Código 2.2.

```
void setup(){
// código a ejecutar una vez
}
void loop(){
// código a ejecutar repetidamente
}
```

Código 2.2. Estructura simplificada de un programa para Arduino

2.1.4.2. Estructura del programa para el controlador

La placa Arduino se encargará de controlar las diferentes funciones del sistema domótico, como se describe a continuación:

- Control del encendido o apagado de luminarias, activación o desactivación del tomacorriente de la radio, gestión del botón de pánico, simulación de presencia.
- Monitoreo de los accesos de la vivienda, medición de temperatura, detección de movimiento.
- Comunicación con la aplicación para dispositivos Android.

En la Figura 2.15, se muestra la estructura general del programa para el controlador Arduino.

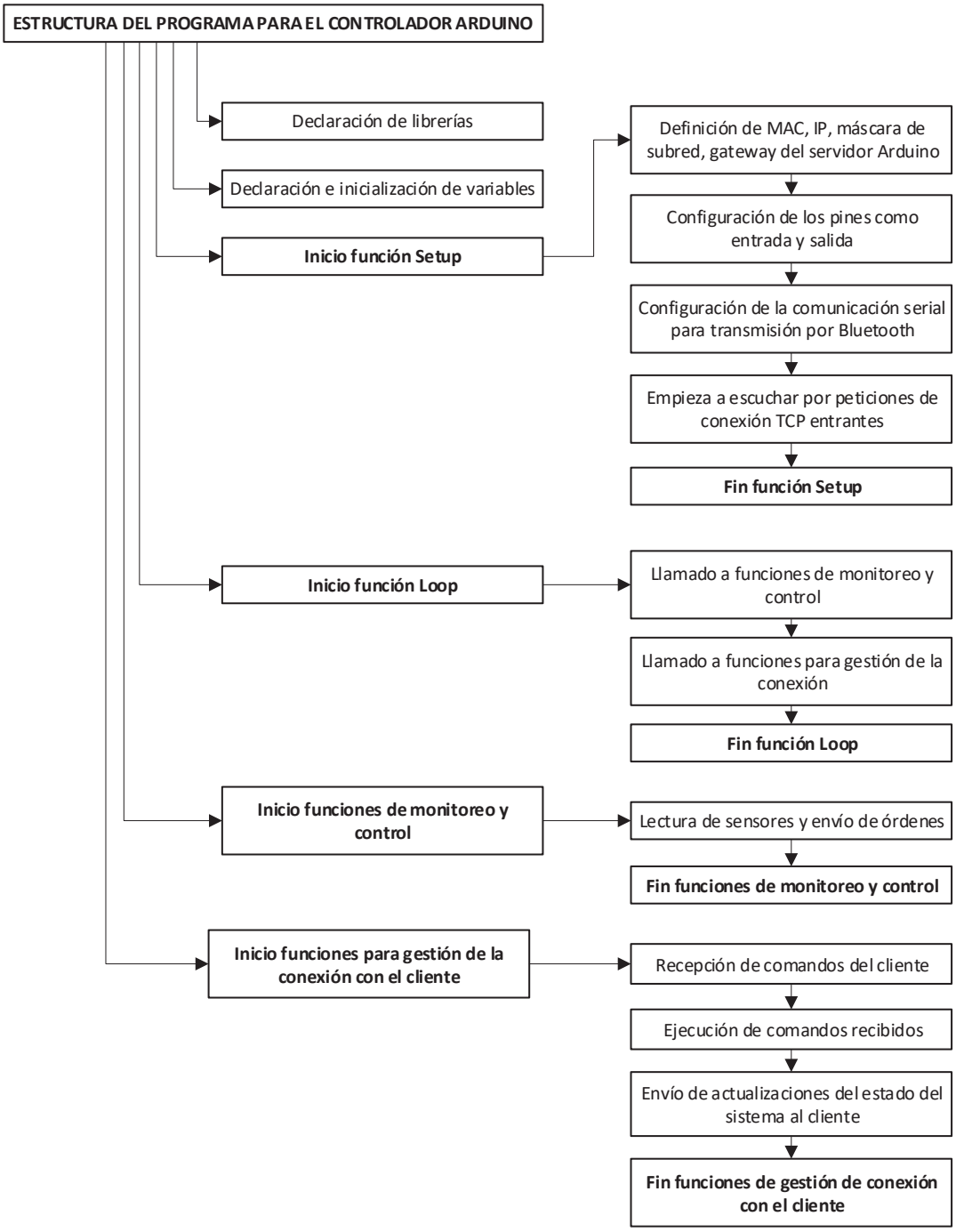

Figura 2.15. Estructura del programa para Arduino

2.1.5. Diseño de la aplicación móvil

La aplicación para el sistema operativo Android permite al usuario gestionar las funcionalidades del sistema domótico desde dentro o fuera de la vivienda, utilizando el Internet o Bluetooth para conectarse al controlador Arduino.

2.1.5.1. Software para el desarrollo de la aplicación Android

Para el desarrollo de la aplicación Android se requiere principalmente de los siguientes componentes de software:

- **Java Development Kit (JDK):** El Java Development Kit es un entorno de desarrollo de aplicaciones y otros componentes para el lenguaje de programación Java [65]. Incluye herramientas útiles para desarrollar, depurar y monitorear aplicaciones Java.
- **Android Software Development Kit:** El Android SDK provee un conjunto de herramientas necesarias para el desarrollo de aplicaciones Android. Existen diferentes lenguajes de programación, como Java o C#, y diversos IDEs, como Android Studio o Visual Studio, que pueden ser utilizados al desarrollar una aplicación, pero en cualquier caso se requiere del Android SDK para acceder a las características del sistema operativo y posibilitar la ejecución de la aplicación en un dispositivo Android. El SDK de Android incluye librerías y herramientas que permiten compilar, probar y depurar aplicaciones Android [66].
- **Xamarin:** Xamarin es una plataforma de desarrollo de aplicaciones móviles que posibilita el desarrollo de aplicaciones nativas para Android, iOS y Windows utilizando el lenguaje de programación C#. Las aplicaciones escritas con Xamarin y C# tienen la posibilidad de crear interfaces de usuario nativas para disminuir el impacto en el rendimiento en tiempo de ejecución. Esto permite que desarrolladores familiarizados con C#, .NET y el IDE Visual Studio sean capaces de crear aplicaciones móviles sin tener que emplear un lenguaje de programación nativo, como Objective-C o Java, en el caso de iOS y Android respectivamente [67].

2.1.5.2. Especificación de la interfaz de usuario

La interfaz de usuario (UI) de la aplicación móvil será desarrollada para la plataforma Android. El diseño de la interfaz visual permite definir su estructura visual. La interfaz visual de la aplicación se define de dos maneras:

- A través de XML[3] para declarar los elementos de la interfaz de usuario.

[3] XML o *eXtensible Markup Language* es una especificación que posibilita la definición de etiquetas personalizadas para organizar y describir datos. Representa la información de manera estructurada para que pueda ser almacenada y transportada.

- De forma programática, al crear instancias o modificar las propiedades de los elementos visuales en tiempo de ejecución.

La ventaja al declarar los elementos de la UI a través de XML radica en la separación de la presentación de la aplicación, del código que se utiliza para manejar su comportamiento.

La Figura 2.16 muestra la estructura de la interfaz de usuario de la aplicación móvil Android.

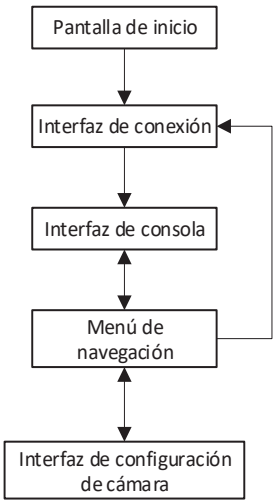

Figura 2.16. Estructura de la interfaz de usuario

Pantalla de inicio

La pantalla de inicio o *splash screen* es la primera experiencia del usuario con la aplicación. Esta interfaz contiene un logo u otro elemento que permita identificar el prototipo de sistema domótico. La vista de la pantalla de inicio se muestra en la Figura 2.17.

Figura 2.17. *Sketch* de la pantalla de inicio

Interfaz de conexión

Permite al usuario autenticarse y establecer una conexión con el controlador del sistema domótico a través de Bluetooth o Internet. La vista de la interfaz de conexión se muestra en la Figura 2.18.

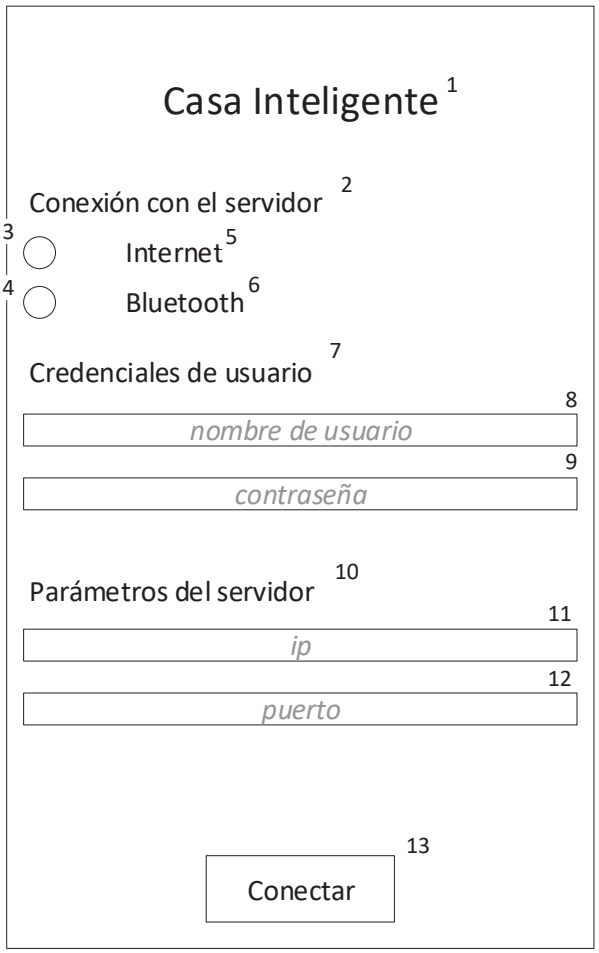

Figura 2.18. *Sketch* de la interfaz de conexión

En la Tabla 2.21, se describe los elementos empleados en la interfaz de conexión.

Interfaz de consola

Permite al usuario controlar los diferentes dispositivos y visualizar el estado del sistema domótico empleando componentes gráficos. Incluye un menú que posibilita navegar entre las diferentes interfaces de la aplicación.

La Figura 2.19 muestra los componentes de la interfaz de consola empleados al monitorear el estado de los accesos de la vivienda. Se incluyen los componentes utilizados al controlar las luces, visualizar la temperatura y activar la alarma. El usuario puede visualizar los elementos de la interfaz de consola al desplazarse hacia arriba o abajo.

Tabla 2.21. Descripción de la interfaz de conexión

Id	Tipo de elemento	Descripción
1	ImageView	Imagen que muestra el logo de la aplicación móvil
2	TextView	Muestra el texto “Conexión con el servidor”
3	RadioButton	Representa la conexión con el servidor a través de Internet
4	RadioButton	Representa la conexión con el servidor a través de Bluetooth
5	TextView	Muestra el texto “Internet”
6	TextView	Muestra el texto “Bluetooth”
7	TextView	Muestra el texto “Credenciales de usuario”
8	EditText	Elemento donde se ingresa el nombre de usuario
9	EditText	Elemento donde se ingresa la contraseña
10	TextView	Muestra el texto “Parámetros del servidor”
11	EditText	Elemento donde se ingresa la IP del servidor
12	EditText	Elemento donde se ingresa el puerto del servidor
13	AppCompatButton	Botón que permite conectarse al servidor

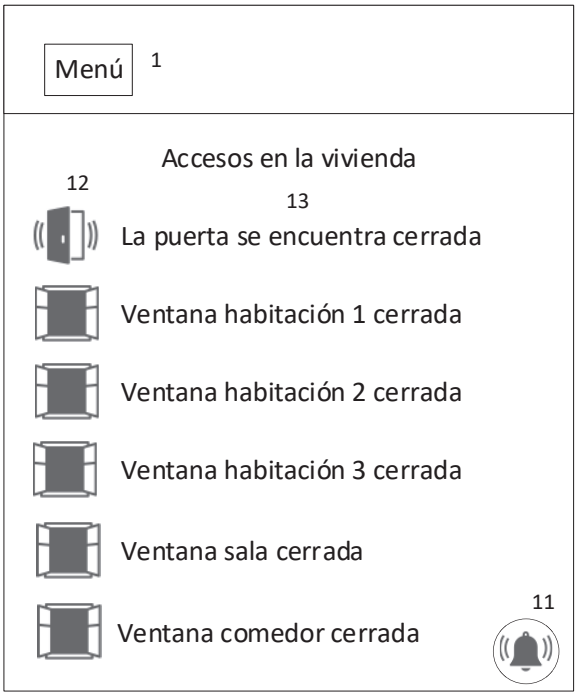

Figura 2.19. *Sketch* de la interfaz de consola (Parte 1 de 2)

Figura 2.19. *Sketch* de la interfaz de consola (Parte 2 de 2)

En la Tabla 2.22, se describen los elementos que componen la interfaz de consola.

Interfaz de configuración de cámara

Figura 2.20. *Sketch* interfaz de configuración de cámara

Posibilita acceder a una aplicación externa al sistema que permite configurar la cámara IP. Incluye el menú para navegar a las diferentes interfaces de la aplicación. La Figura 2.20 muestra la vista de la interfaz de para acceder a la configuración de la cámara.

Tabla 2.22. Descripción de la interfaz de consola

Id	Tipo de elemento	Descripción
1	Button	Botón que permite desplegar el menú de navegación.
2	ImageView	Imagen que muestra una luminaria que cambia de color al ser encendida.
3	TextView	Muestra un texto para identificar en donde se ubica cada luminaria.
4	Switch	Es un botón que puede permanecer en dos estados, *ON/OFF*. Permite encender o apagar una luminaria.
5	ImageView	Imagen que muestra un termómetro.
6	TextView	Texto que indica que la medición de la temperatura se realiza en la vivienda.
7	TextView	Texto que representa la temperatura en la vivienda.
8	ImageView	Imagen que cambia de color para indicar si una función del sistema está activada o desactivada.
9	TextView	Texto que indica una función del sistema domótico relacionado a la alarma.
10	Switch	Botón que puede permanecer en dos estados, *ON/OFF*. Permite activar o desactivar una función del sistema domótico relacionado a la alarma.
11	FloatingActionButton	Botón flotante que permite activar o desactivar la función de botón de pánico.
12	ImageView	Imagen que cambia de color para indicar si una puerta o ventana en la vivienda permanece abierta o cerrada.
13	TextView	Muestra un texto para identificar la estancia de la vivienda donde se ubican las puertas y ventanas.

La Tabla 2.23 describe los elementos que forman parte de la interfaz de configuración de cámara.

Tabla 2.23. Descripción de la interfaz de configuración de la cámara

Id	Tipo de elemento	Descripción
1	Button	Botón que permite desplegar el menú de navegación.
2	ImageView	Imagen que indica al usuario que está por acceder a la configuración de la cámara.
3	TextView	Texto que al ser presionado abre una aplicación externa al sistema para configurar la cámara.
4	FloatingActionButton	Botón flotante que permite activar o desactivar la función de botón de pánico.

Panel lateral de navegación

Posibilita navegar entre las interfaces de la aplicación móvil. Permite acceder a las interfaces de consola, configuración de cámara y conexión. Además, cuenta con una opción que permite cerrar la aplicación.

Se encuentra formado por una cabecera -que muestra el nombre de la aplicación- y por un menú, que a su vez está compuesto por ítems que actúan como enlaces a otras interfaces de la aplicación, como se observa en la Figura 2.21.

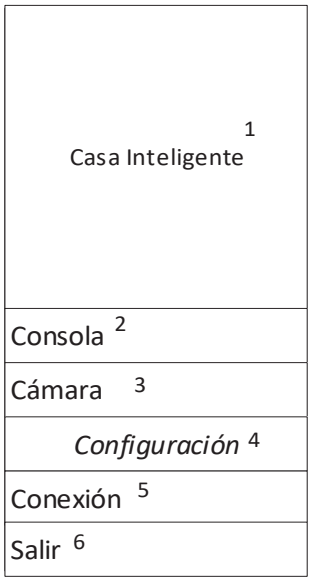

Figura 2.21. *Sketch* del panel lateral de navegación

En la Tabla 2.24 se describen los componentes del menú de navegación.

Tabla 2.24. Elementos del panel de navegación

Id	Tipo de elemento	Descripción
1	FrameLayout	Representa la cabecera del menú de navegación. En esta se ubica el nombre de la aplicación móvil.
2	Item	Es un enlace a la interfaz de consola.
3	Item	Es un enlace a la interfaz de configuración de la cámara.
4	Item	Contiene el texto “Configuración”.
5	Item	Es un enlace a la interfaz de conexión.
6	Item	Permite salir de la aplicación móvil.

2.1.5.3. Diagrama de clases de la aplicación móvil

Un diagrama de clases modela el comportamiento estático de un sistema enfocándose en los elementos que lo conforman.

Permite modelar la estructura de la aplicación móvil al describir las clases que la componen, sus atributos, métodos, y las asociaciones entre dichas clases.

En un diagrama de clases, cada clase es representada a través de un rectángulo con 3 secciones. La primera contiene el nombre de la clase, la segunda sección contiene los atributos de la clase, y la tercera sección contiene las operaciones o métodos que la clase puede utilizar.

Se emplea una línea con una punta de flecha que se dirige de la clase derivada a la clase base para representar una relación de herencia. Se utiliza una línea sólida que conecta dos clases para representar una relación de asociación.

Para gestionar la conexión con el servidor incluyen las clases ConexionInternet, ConexionBluetooth y GestorComunicacion.

Para notificar al usuario los cambios en el estado del sistema se emplea la clase GestorNotificacion.

Para mantener el estado del sistema en la aplicación móvil se emplean las clases Radio, Luminaria, Sirena, Dispositivo, Ventana, Puerta, Alarma y Controlador.

La Figura 2.22 muestra el diagrama de clases de la aplicación móvil.

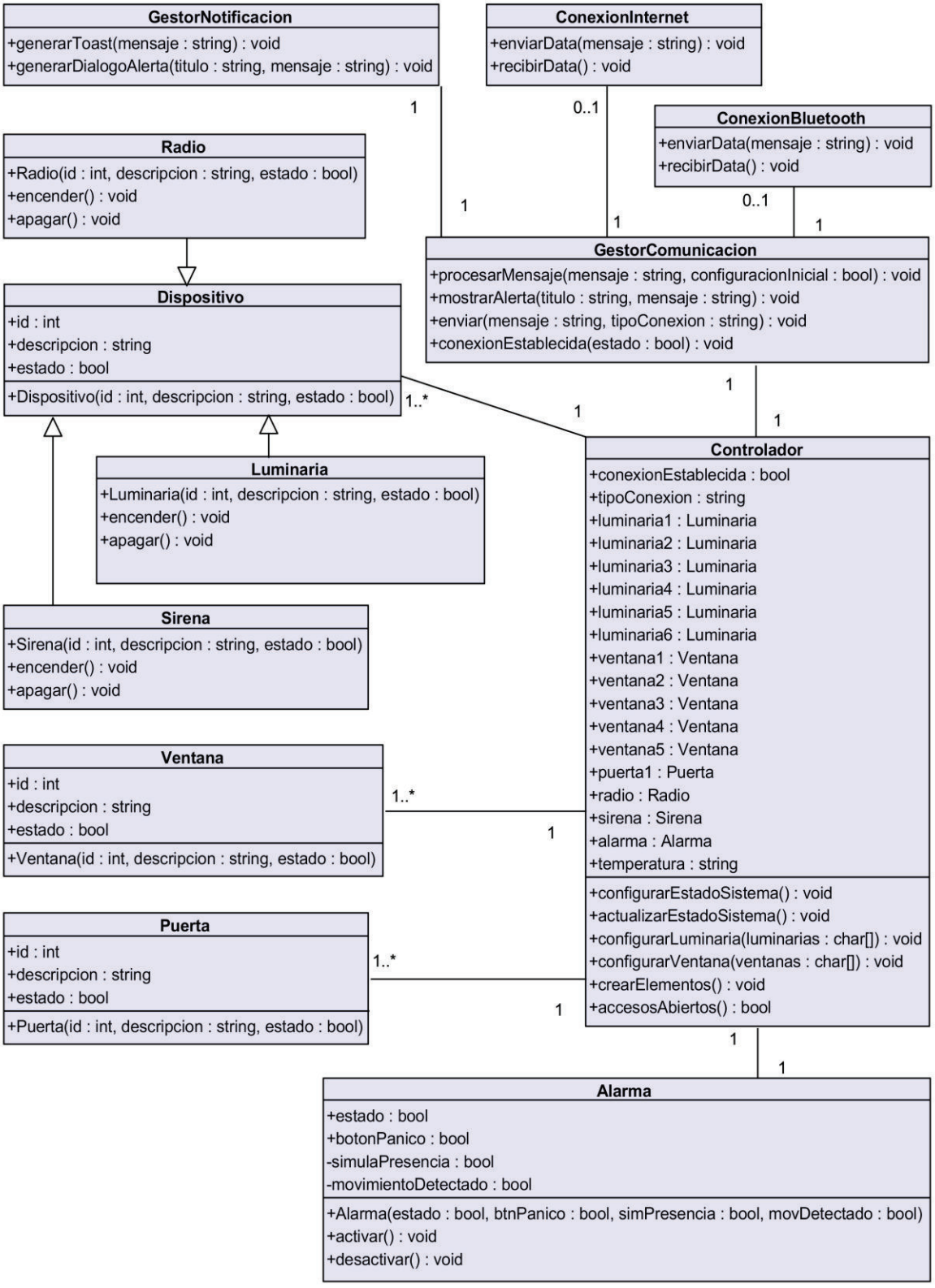

Figura 2.22. Diagrama de clases de la aplicación para Android

A continuación, se describen las clases que componen la aplicación móvil para el sistema operativo Android.

Clase ConexionInternet

Esta clase permite a la aplicación Android enviar y recibir información del controlador Arduino a través de Internet. Para ello, se emplean los métodos `enviarData (string mensaje)` y `recibirData()`. La clase ConexionInternet se muestra en la Figura 2.23.

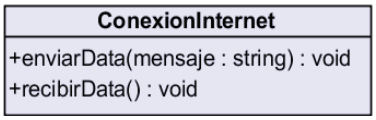

Figura 2.23. Clase ConexionInternet

Clase ConexionBluetooth

Permite a la aplicación enviar y recibir información con el controlador Arduino a través de la interfaz Bluetooth del dispositivo Android. Los métodos que componen esta clase se muestran en la Figura 2.24.

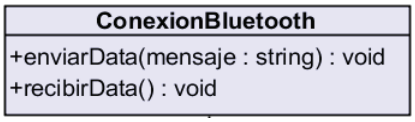

Figura 2.24. Clase ConexionBluetooth

Clase GestorNotificacion

La clase GestorNotificacion es utilizada para comunicar mensajes al usuario. Esta clase contiene los métodos `generarToast` y `generarDialogoAlerta`, como se muestra en la Figura 2.25.

- **`generarToast:`** Este método permite mostrar información de forma temporal en la pantalla a través de un *toast*. Un *toast* es un mensaje que se despliega en la interfaz de usuario dentro de pequeños rectángulos traslúcidos, que desaparecen de forma automática transcurrido un tiempo. Un *toast* suele ser utilizado para mostrar información sencilla al usuario en sin requerir ningún tipo de confirmación por parte del mismo.

- **`generarDialogoAlerta:`** Este método emplea cuadros de diálogo, es decir, ventanas pequeñas que no ocupan toda la pantalla, que requieren la interacción del usuario para poder continuar, con el fin de mostrar información en la pantalla.

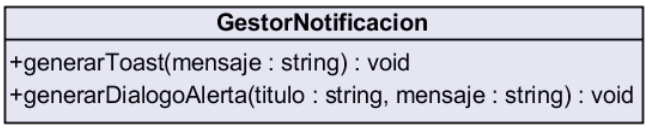

Figura 2.25. Clase GestorNotificacion

Clase GestorComunicación

Esta clase está compuesta por los métodos `procesarMensaje`, `mostrarAlerta`, `enviar` y `conexionEstablecida`, como se indica en la Figura 2.26.

- **`procesarMensaje`:** Este método permite actualizar el estado del sistema domótico en la aplicación Android en base al mensaje recibido desde el controlador Arduino, lo que posibilita mostrar el estado actual de los diferentes dispositivos de la vivienda en la interfaz de usuario.
- **`mostrarAlerta`:** Permite acceder a los métodos de la clase GestorNotificacion, utilizados para mostrar cuadros de texto con información al usuario.
- **`enviar`:** Posibilita acceder a los métodos de las clases ConexionBluetooth y ConexionInternet, empleados para enviar información al controlador.
- **`conexionEstablecida`:** Configura el estado de una variable, que indica si la conexión entre la aplicación y el controlador Arduino está establecida o cerrada.

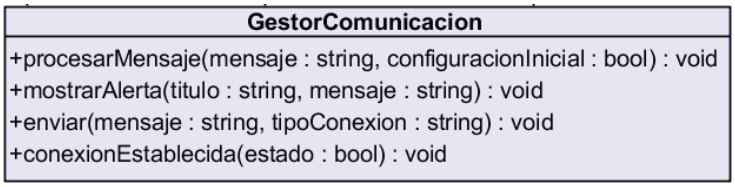

Figura 2.26. Clase GestorComunicacion

Clase Radio

Esta clase almacena el estado de la radio, es decir, permite conocer si la radio se encuentra encendida o apagada. Contiene los métodos `Radio`, `encender` y `apagar`, como se indica en la Figura 2.27.

- **`Radio`**: Es el método constructor de la clase. Posibilita inicializar los objetos, es decir, permite asignar valores iniciales a los atributos `id`, `descripción` y `estado`.
- **`encender`:** Este método establece el valor del atributo `estado` en `true`, para indicar que la radio en la vivienda está encendida.

- **apagar:** Establece el valor del atributo `estado` en `false`, para indicar que la radio se encuentra apagada.

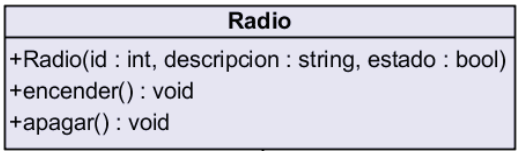

Figura 2.27. Clase Radio

Clase Luminaria

La clase Luminaria almacena el estado de las luces de la vivienda, sea encendido o apagado. Esta clase contiene los métodos `Luminaria`, `encender` y `apagar`, como se indica en la Figura 2.28.

- **Luminaria:** Es el método constructor utilizado para inicializar las variables `id`, `descripción` y `estado` asociadas a una luminaria.
- **encender:** Asigna `true` al atributo `estado`, para indicar que una luminaria está encendida.
- **apagar:** Asigna `false` al atributo `estado`, para indicar que una luminaria está apagada.

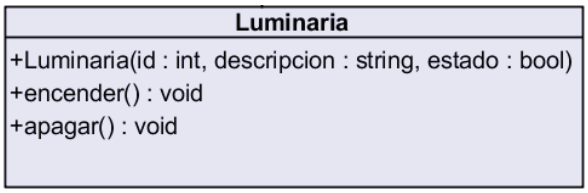

Figura 2.28. Clase Luminaria

Clase Sirena

Esta clase guarda el estado de la sirena, la cual se puede encender para generar una alerta sonora en la vivienda. Como se observa en la Figura 2.29, esta clase contiene los métodos `Sirena, encender` y `apagar`.

- **Sirena:** Este método asigna un valor inicial al `id, descripción` y `estado` de la sirena en la aplicación móvil.
- **encender:** Asigna a la variable `estado` el valor de `true` para indicar que la sirena en la vivienda está encendida.

- **`apagar:`** Asigna a la variable `estado` el valor `false` para indicar que la sirena en la vivienda está apagada.

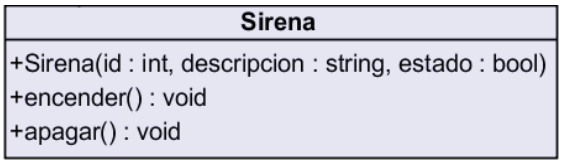

Figura 2.29. Clase Sirena

Clase Dispositivo

La superclase Dispositivo hereda a las subclases Luminaria, Radio y Sirena los atributos `id, descripción` y `estado`. Esta clase contiene un método constructor que recibe 3 parámetros, como se observa en la Figura 2.30. Este constructor es utilizado por los métodos constructores de las subclases para inicializar las variables `id`, `descripción` y `estado`.

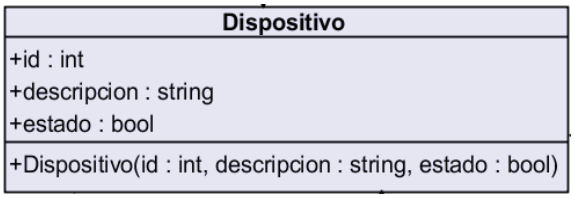

Figura 2.30. Clase Dispositivo

Clase Ventana

Esta clase almacena el estado de una ventana en la vivienda, la cual puede permanecer abierta o cerrada. Como se muestra en la Figura 2.31, esta clase contiene un método llamado `Ventana`, que se encarga de dar un valor inicial a los atributos `id, descripción` y `estado.`

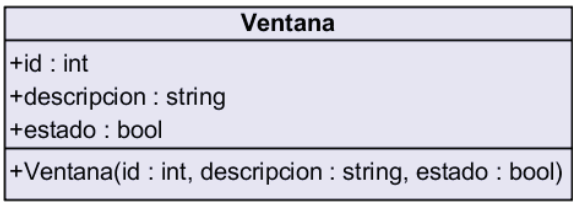

Figura 2.31. Clase Ventana

Clase puerta

Esta clase indica si la puerta en la vivienda permanece cerrada o abierta. Contiene un método llamado `Puerta`, que se encarga de dar un valor inicial a los atributos `id`, `descripción` y `estado`, como se indica en la Figura 2.32.

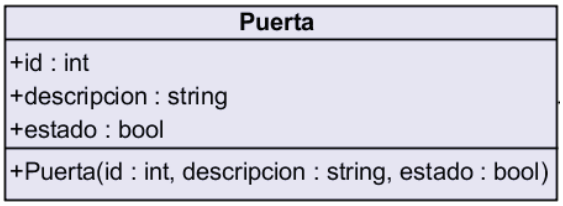

Figura 2.32. Clase puerta

Clase Alarma

Esta clase mantiene el estado de las diferentes funciones del sistema domótico relacionadas a la alarma y generación de alertas en la vivienda. La clase alarma contiene los atributos `estado`, `botonPanico`, `simularPresencia`, `movimientoDetectado` y los métodos `Alarma`, `activar` y `desactivar`, como se indica en la Figura 2.33.

A continuación, se describe los atributos de esta clase:

- **`estado`:** Indica si la alama está activada o desactivada. El usuario puede activar las funciones de simulación de presencia, alertas por detección de movimiento y alertas por apertura de puertas o ventanas, únicamente cuando la alarma está activada.
- **`botonPanico`:** Indica si el botón de pánico está activado o desactivado.
- **`simulaPresencia`:** Indica si la simulación de presencia se encuentra activada o desactivada.
- **`movimientoDetectado`**: Indica si se ha detectado movimiento en la vivienda.

Posteriormente, se describen los métodos de esta clase:

- **`Alarma`**: Este método constructor permite dar valores iniciales a los atributos `estado`, `botonPanico`, `simularPresencia`, `movimientoDetectado`.
- **`activar`:** Este método posibilita al usuario activar la alarma, al dar el valor de `true` a la variable `estado`.
- **`desactivar`**: Este método posibilita al usuario desactivar la alarma, al dar el valor de `false` a la variable `estado`.

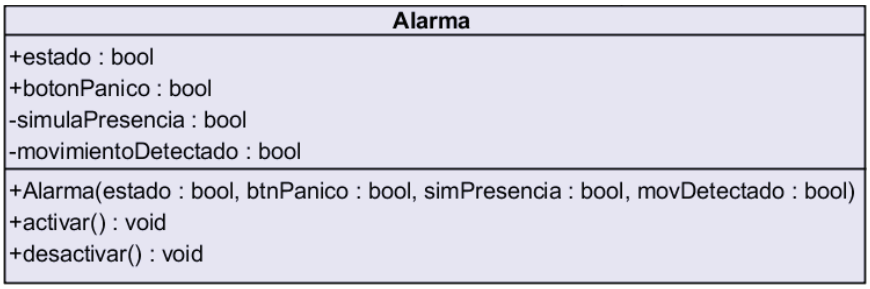

Figura 2.33. Clase Alarma

Clase Controlador

Esta clase almacena el estado de todo el sistema domótico en la aplicación móvil. Contiene instancias de las diferentes clases antes mencionadas para representar a cada uno de los elementos del sistema domótico. Los atributos y métodos de esta clase se muestran en la Figura 2.34.

A continuación, se describen los atributos de la clase Controlador:

- **`conexionEstablecida`**: Indica si la conexión entre la aplicación Android y el controlador Arduino está establecida o cerrada.
- **`tipoConexion`**: Indica si la conexión entre la aplicación Android y el controlador Arduino se realiza a través de Internet o Bluetooth.
- **`luminaria1:`** Instancia de la clase Luminaria que representa una lámpara en la habitación 1 de la vivienda.
- **`luminaria2:`** Representa una lámpara en la habitación 2 de la vivienda.
- **`luminaria3:`** Representa una lámpara en la habitación 3 de la vivienda.
- **`luminaria4`**: Representa una lámpara ubicada en la sala de la vivienda.
- **`luminaria5`**: Representa una lámpara ubicada en el comedor de la vivienda.
- **`luminaria6`**: Representa una lámpara ubicada en el baño de la vivienda.
- **`ventana1:`** Representa una ventana en la habitación 1 de la vivienda.
- **`ventana2:`** Representa una ventana en la habitación 2 de la vivienda.
- **`ventana3`**: Representa una ventana en la habitación 3 de la vivienda.
- **`ventana4:`** Representa una ventana ubicada en la sala de la vivienda.

- **`ventana5:`** Representa una ventana ubicada en el comedor de la vivienda.
- **`puerta1:`** Representa la puerta principal de la vivienda.
- **`radio:`** Instancia de la clase Radio, que representa un dispositivo receptor de radio.
- **`sirena:`** Instancia de la clase Sirena, que representa la bocina empleada al generar alertas en la vivienda.
- **`alarma`**: Instancia de la clase Alarma, que representa la funcionalidad de alarma del sistema domótico.
- **`temperatura:`** Esta variable indica la temperatura en la vivienda.

Posteriormente, se describen los métodos de la clase Controlador:

- **`configurarEstadoSistema`**: Este método se encarga de almacenar el estado del sistema domótico en la aplicación móvil, es decir, configura el valor de la variable `estado` de las instancias dentro de la clase Controlador en base a información recibida de Arduino.

  Esto permite a la aplicación Android conocer el estado del sistema domótico una vez que el usuario ha accedido a la aplicación.
- **`actualizarEstadoSistema`**: Este método actualiza el estado del sistema domótico en la aplicación móvil en base a la información enviada desde el controlador Arduino.
- **`configurarLuminaria`**: Este método es utilizado para configurar el estado de las luminarias en la aplicación Android cuando el usuario accede a la aplicación.
- **`configurarVentana`**: Este método es utilizado para configurar el estado de las ventanas en la aplicación Android cuando el usuario accede a la aplicación.
- **`crearElementos`**: En este método se crean instancias de las clases descritas anteriormente para representar los diferentes elementos del sistema domótico.
- **`accesosAbiertos`**: Este método permite conocer si los accesos en la vivienda, es decir, las ventanas y la puerta, permanecen abiertos o cerrados.

2.1.6. Diseño del Sistema de Control Domótico

En el presente apartado se describe el sensado de variables, control de dispositivos en la vivienda y el mecanismo empleado para transferir información entre el controlador Arduino y la aplicación para Android.

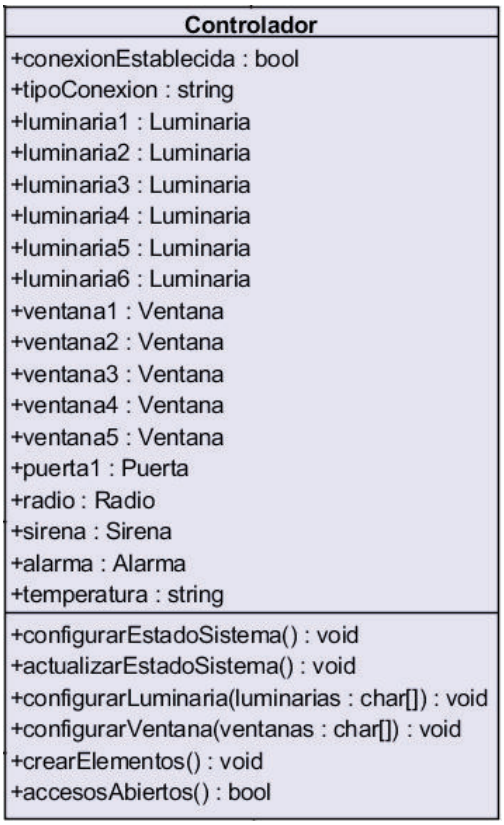

Figura 2.34. Clase Controlador

2.1.6.1. Comunicación dispositivo móvil-controlador

El intercambio de información entre la aplicación móvil y la placa Arduino se realiza a través de Internet o Bluetooth. A continuación, se detalla el método de transferencia de información entre el controlador Arduino y la aplicación móvil de usuario.

Comunicación mediante Internet

El usuario puede gestionar el sistema domótico a través de Internet. Esto posibilita el control de los diferentes dispositivos desde dentro o fuera de la vivienda.

El controlador Arduino actúa como servidor, es decir, como un dispositivo encargado de responder las peticiones de los clientes. La aplicación de Android actúa como cliente, ya que se encarga de iniciar la comunicación con el servidor y enviar peticiones.

Para el intercambio de información entre la aplicación de Android y la placa Arduino se emplean *sockets*. Los *sockets* representan una abstracción de la red de comunicación que permite el envío y recepción de datos entre procesos en una misma máquina o en máquinas diferentes [68].

Los *sockets* Berkeley, conocidos universalmente como *sockets*, hacen referencia a una API[4] que brinda un conjunto de funciones que permite a los programadores incluir la capacidad de comunicación a través de Internet a sus programas. Los *sockets* Berkeley son la API estándar en la industria al crear y utilizar *sockets*. Fueron inicialmente utilizados como una API para el sistema operativo Unix, pero después fueron adoptados por TCP/IP.

La comunicación basada en *sockets* es independiente del lenguaje de programación utilizado para implementarlo. Esto quiere decir, que un programa que implemente *sockets* utilizando el lenguaje Java, como la aplicación de Android de este prototipo, se puede comunicar con un *socket* desarrollado con otro lenguaje de programación.

Para hacer posible la comunicación entre dos dispositivos se crea un *socket* en cada uno de ellos, es decir, un *socket* representa un extremo de la conexión. Se puede identificar a un *socket* a través de los siguientes parámetros: una dirección IP, que permite identificar a un dispositivo o *host*, un número de puerto, que permite identificar a un proceso en ese *host* y un protocolo, por ejemplo, TCP (*Transmission Control Protocol*) o UDP (*User Datagram Protocol*) [69].

Se emplea TCP cuando se requiere intercambiar información de forma confiable. TCP posibilita la recuperación de segmentos que se pierden, dañados, duplicados o recibidos en orden incorrecto. Para ello emplea números de secuencia en cada segmento, requiere que un mensaje de confirmación sea enviado en respuesta al envío de datos y emplea una función conocida como CRC o verificación de redundancia cíclica que permite detectar errores producidos en la transmisión de datos. Al contrario de TCP, UDP no garantiza la entrega de datos ni el manejo de mensajes duplicados, sin embargo, suele ser utilizado por aplicaciones donde la velocidad es más importante que la confiabilidad. Esto se debe principalmente a que UDP no emplea mensajes de confirmación o ACK, lo que posibilita un flujo continuo de paquetes [70].

Con UDP el transmisor no espera a recibir mensajes de confirmación, en su lugar, continua con el envío de los paquetes siguientes. Si algunos paquetes no llegan al receptor, este no espera a que se reenvíen, lo que disminuye la sobrecarga en la transmisión.

Una conexión entre dos dispositivos es identificada de forma única a través de una 5-tupla formada por:

---

[4] API: Una API o *Application Programming Interface* es una especificación que describe estructuras de datos, rutinas, variables, clases, entre otros. Es utilizada por componentes de software como una interfaz para comunicarse entre sí.

- La dirección IP de origen y destino.
- Los puertos origen y destino.
- El protocolo, como, TCP o UDP.

En la Figura 2.35, se describe el proceso de comunicación entre la aplicación móvil y el controlador Arduino empleando las funciones o métodos que la *Berkeley Sockets API* provee.

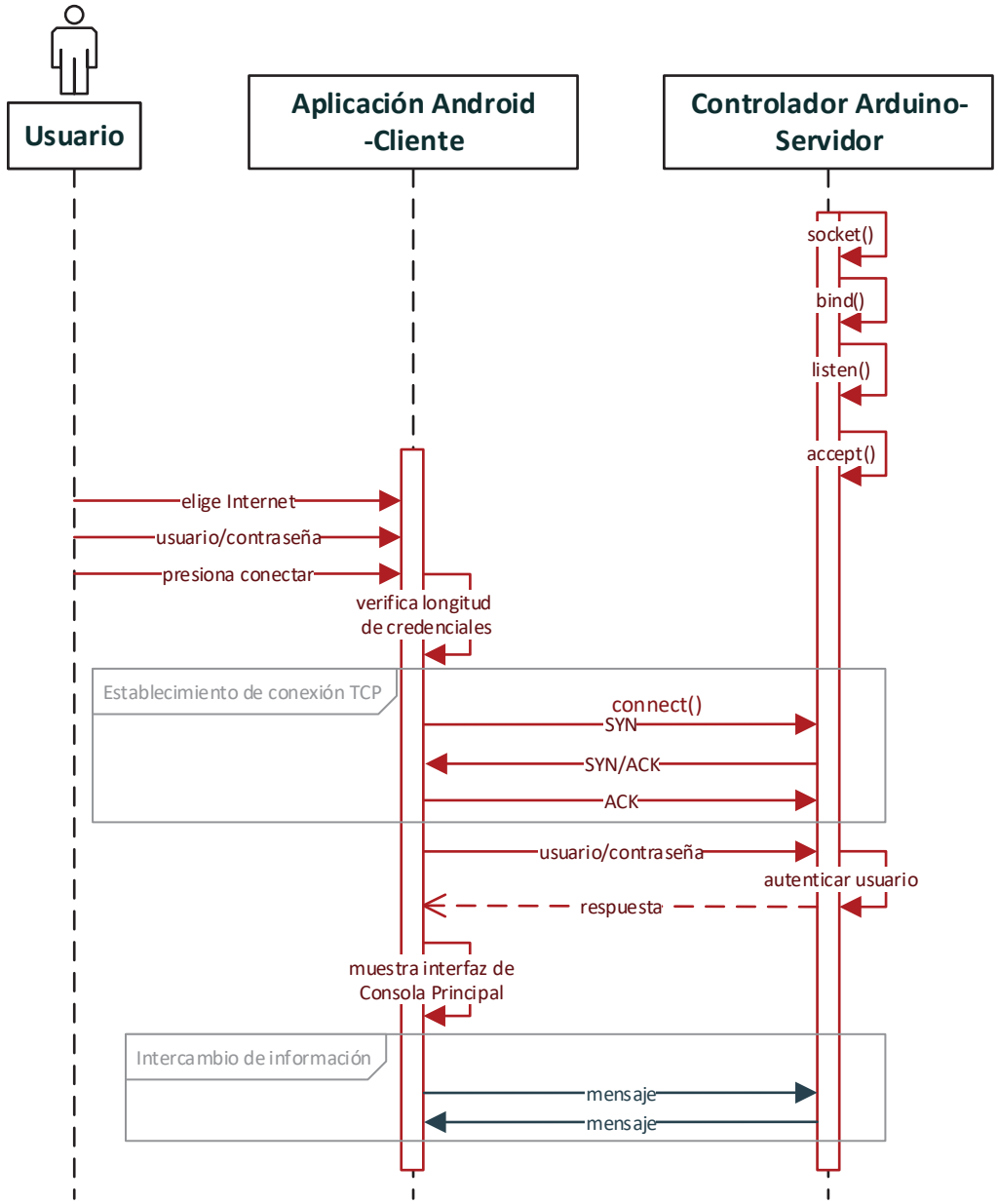

Figura 2.35. Comunicación aplicación móvil-controlador a través de Internet

Para empezar, la aplicación Android (cliente) se conecta al controlador (servidor), se intercambia información, y finalmente se cierra la conexión. Los *sockets* Berkeley definen nombres estándar para las funciones, algunas de las cuales se describen a continuación:

- A través de la función `socket()`, se crea un *socket* en la aplicación Android y el controlador Arduino. Para identificar a cada *socket* se emplea una dirección IP, un número de puerto y un protocolo, en este caso, el protocolo TCP.
- Se emplea `bind()` de lado del servidor para asociar el *socket* a una dirección IP y puerto.
- La función `listen()` se emplea de lado del servidor para colocar al *socket* en estado de escucha.
- Se emplea `connect()` de lado del cliente para asignar al *socket* un puerto libre e intentar establecer una conexión TCP con el otro extremo.
- `accept()` permite al servidor aceptar la petición del cliente de establecer la conexión TCP y crea un nuevo *socket* para manejar dicha conexión con el cliente.
- Las funciones `send()`, `receive()`, `read()` y `write()` permiten enviar y recibir datos a través del *socket*.
- `close()` permite al sistema liberar los recursos asignados al *socket* y cierra la conexión TCP.

Comunicación mediante Bluetooth

El sistema domótico puede ser gestionado desde la aplicación móvil a través de Bluetooth. Bluetooth es una tecnología inalámbrica de corto alcance destinada a reemplazar los cables al conectar diferentes dispositivos electrónicos. La mayoría de dispositivos Bluetooth están diseñados para trabajar en un rango de 10 metros. Bluetooth opera a una frecuencia de 2.4 GHz, en la misma banda de frecuencia que otros protocolos. Una red Bluetooth, conocida como Piconet, emplea un modelo maestro/esclavo para controlar que dispositivos pueden enviar datos. El dispositivo maestro coordina la comunicación en la Piconet, puede enviar datos a cualquiera de los dispositivos esclavos y solicitar datos de los mismos. Los dispositivos esclavos solo pueden transmitir y recibir de su maestro. Es decir, no se pueden comunicar con otros esclavos en la Piconet. Un maestro puede comunicarse con un máximo de 7 dispositivos esclavos.

El controlador Arduino emplea un módulo Bluetooth para comunicarse con el dispositivo móvil. En este contexto, la plataforma Arduino actúa como dispositivo esclavo, mientras que el teléfono inteligente actúa como maestro.

A continuación, se describe el proceso para crear una conexión Bluetooth entre dos dispositivos:

- **Descubrimiento (*Inquiry*)** [71]**:** Permite a un dispositivo conocer la existencia de otros dispositivos que se encuentran en rango o dentro de su zona de cobertura. Un dispositivo envía un paquete de *inquiry* o descubrimiento, y cualquier dispositivo que escuche dicho mensaje y que permita ser descubierto responde con su dirección Bluetooth, nombre, entre otros. Una vez que el proceso de descubrimiento se haya completado, el dispositivo que realiza el proceso de descubrimiento puede elegir establecer una conexión con el dispositivo descubierto a través del procedimiento de *paging*.

- **Conexión (*Paging*):** Para establecer una conexión entre dos dispositivos es necesario que ambos conozcan la dirección Bluetooth del otro, que puede ser obtenida en el proceso de descubrimiento. En el proceso de *paging* ambos dispositivos sincronizan su reloj y frecuencias a utilizar, y se establece una conexión entre los dispositivos.

- Cuando un dispositivo ha completado el proceso de *paging*, entra en modo de conexión establecida, lo que posibilita el envío o recepción de información.

Un dispositivo puede incluir un mecanismo de seguridad conocido como emparejamiento o *pairing*. Una vez que se ha establecido una conexión con un dispositivo remoto por primera vez, una solicitud de emparejamiento se presenta automáticamente al usuario en el dispositivo Android.

Cuando un dispositivo está emparejado, información básica sobre ese dispositivo -como el nombre, dirección MAC, clase- es almacenada en memoria. Al conocer la dirección MAC del dispositivo remoto, se puede iniciar una conexión con el mismo sin la necesidad de realizar el proceso de descubrimiento, asumiendo que el dispositivo permanece dentro de rango.

El emparejamiento permite que dos dispositivos sean conscientes de la existencia del otro, y compartan una clave que posibilite que se autentiquen el uno al otro y establezcan una conexión cifrada. El emparejamiento se realiza con una clave de cifrado conocida como PIN (*Personal Information Number*) [35]. El dispositivo esclavo envía una solicitud de

emparejamiento al dispositivo maestro. Por lo general, el dispositivo maestro requiere la intervención del usuario para ingresar el PIN del dispositivo esclavo [72]. Si el PIN ingresado es correcto, la conexión se establece.

Para que el dispositivo móvil Android intercambie información con el módulo Bluetooth del controlador Arduino, es necesario que ambos dispositivos estén emparejados. En la Figura 2.36, se muestra el procedimiento que permite a la placa Arduino y al dispositivo móvil Android intercambiar información.

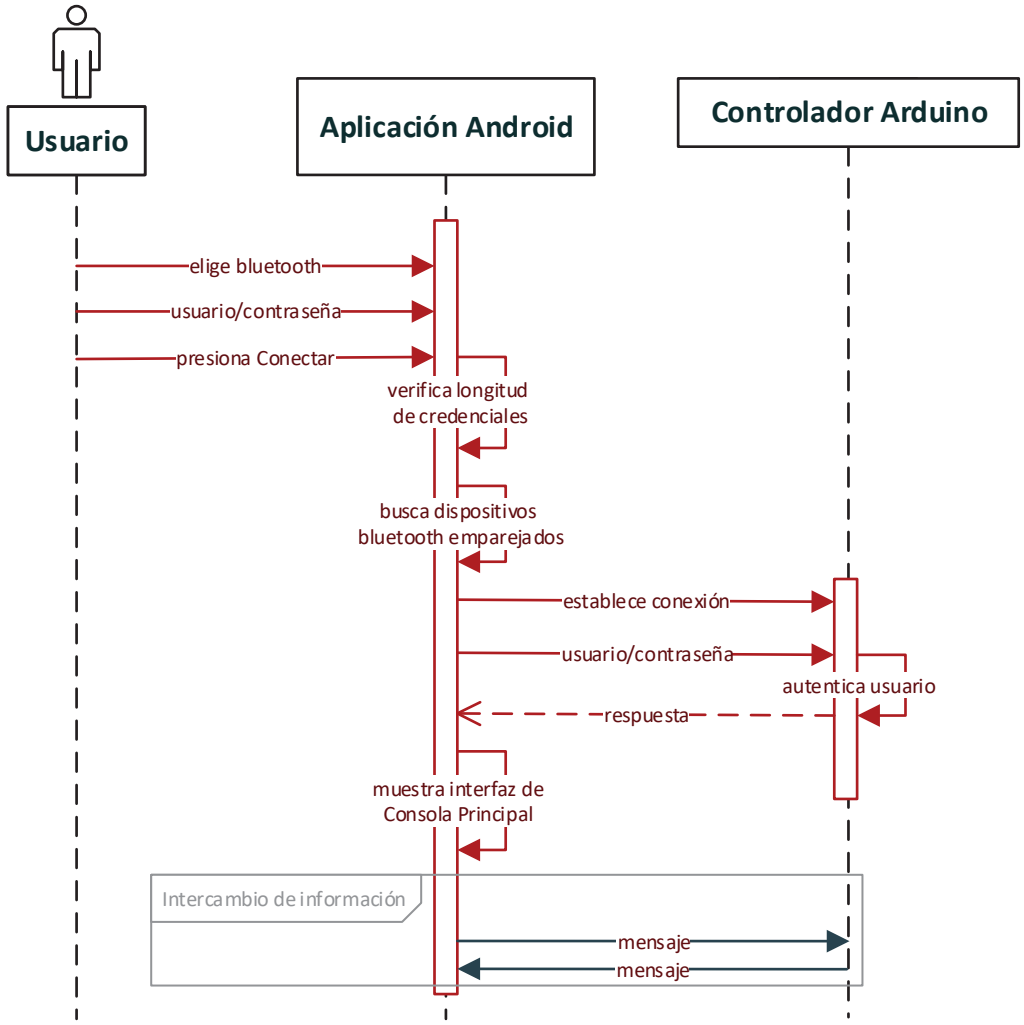

Figura 2.36. Comunicación aplicación móvil-controlador a través de Bluetooth

Intercambio de información

Una vez establecida la conexión entre la aplicación Android y el controlador se procede al intercambio de mensajes entre estos dos dispositivos. El envío de mensajes entre ambos dispositivos permite a la aplicación de Android dar aviso al controlador la acción que el

usuario quiere llevar a cabo, y posibilita que el controlador comunique al usuario cambios en el estado del sistema, como la apertura de una puerta o el encendido de una luminaria en la vivienda.

Se emplean dos tipos de mensajes, los cuales se detallan a continuación:

- **Mensaje de configuración inicial:** Es utilizado únicamente cuando el usuario se autentica con el controlador. Este mensaje enviado desde el controlador Arduino informa a la aplicación de Android el resultado del proceso de autenticación y en caso de que la autenticación haya sido exitosa, indica también el estado de todo el sistema.

  Cuando la autenticación del usuario ha sido exitosa, Arduino envía un mensaje formado por una secuencia de 24 caracteres, como se muestra en la Figura 2.37. Cada carácter es representado por un byte, de acuerdo al esquema de codificación ASCII[5].

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
O	K	#	1	0	0	0	0	1	2	1	.	3	0	0	0	0	0	1	1	1	1	1	0

Figura 2.37. Mensaje de configuración inicial

  El contenido del mensaje de configuración inicial se detalla en la Tabla 2.25.

  Cuando la autenticación del usuario ha sido fallida, el controlador envía un mensaje que indica el fallo a la aplicación móvil. Este mensaje está formado por 5 caracteres que forman la palabra ERROR, donde cada carácter es representado con un byte de acuerdo al esquema de codificación ASCII.

- **Mensaje de actualización:** Un mensaje de actualización o comando indica el estado de un dispositivo o función del sistema domótico. Está formado por 3 caracteres o 3 bytes al emplear la codificación ASCII.

  El primer carácter indica la funcionalidad o tipo de dispositivo al que se hace referencia en el mensaje. El segundo carácter especifica el número de dispositivo, lo que permite diferenciar a un elemento de otro del mismo tipo. El tercer carácter indica el estado del dispositivo o funcionalidad.

[5] ASCII (*American Standard Code for Information Interchange)* es un código que posibilita representar de forma numérica un carácter, empleando para ello 7 bits.

Un ejemplo de comando o mensaje de actualización se muestra en la Figura 2.38.

tipo de elemento	número	estado
0	4	1

Figura 2.38. Mensaje de actualización

Donde el 0 indica que el mensaje hace referencia a una luminaria, el 4 indica que se trata de la luminaria 4, y el 1 indica que esta se encuentra encendida. Por lo tanto, este comando notifica que la luminaria 4 permanece encendida.

Tabla 2.25. Composición del mensaje de configuración inicial

Posición carácter	Descripción
1-3	Caracteres OK# indican autenticación exitosa.
4	Estado de la luminaria en la habitación 1, 1 es encendido y 0 apagado.
5	Estado de la luminaria en la habitación 2, 1 es encendido y 0 apagado.
6	Estado de la luminaria en la habitación 3, 1 es encendido y 0 apagado.
7	Estado de la luminaria en la sala, 1 es encendido y 0 apagado.
8	Estado de la luminaria en el comedor, 1 es encendido y 0 apagado.
9	Estado de la luminaria en el baño, 1 es encendido y 0 apagado.
10-13	Representa la temperatura en la vivienda en °C.
14	Estado de la radio, 1 representa encendido y 0 apagado.
15	Estado de la alarma, 1 representa activado y 0 desactivado.
16	Detección de movimiento, 1 es activado, 0 desactivado.
17	Simulación de presencia, 1 representa activado y 0 desactivado.
18	Estado de la puerta principal, 1 representa abierto y 0 cerrado.
19	Estado de la ventana en la habitación 1, 1 es abierto y 0 cerrado.
20	Estado de la ventana en la habitación 2, 1 es abierto y 0 cerrado.
21	Estado de la ventana en la habitación 3, 1 es abierto y 0 cerrado.
22	Estado de la ventana en la sala, 1 es abierto y 0 cerrado.
23	Estado de la ventana en el comedor, 1 es abierto y 0 cerrado.
24	Estado de la sirena, 1 representa encendido y 0 apagado.

2.1.6.2. Sensado de variables

El prototipo de sistema domótico permite llevar a cabo el sensado del estado de las ventanas, sensado del estado de la puerta, sensado de temperatura y la detección de movimiento en la vivienda. El sensado de variables se especifica en el diagrama de bloques de la Figura 2.39, el cual se describe a continuación:

- El sistema cuenta con diferentes sensores, los cuales, a través de un circuito electrónico, entregan señales eléctricas al controlador Arduino.
- El controlador Arduino recibe las señales generadas por los sensores, las interpreta, y envía un mensaje de actualización a la aplicación móvil, para notificar el estado de los elementos del sistema.
- La aplicación recibe el mensaje de actualización e interpreta su significado. Posteriormente muestra en la interfaz de consola descrita en la Figura 2.19, el estado de los elementos del sistema en base al mensaje recibido.

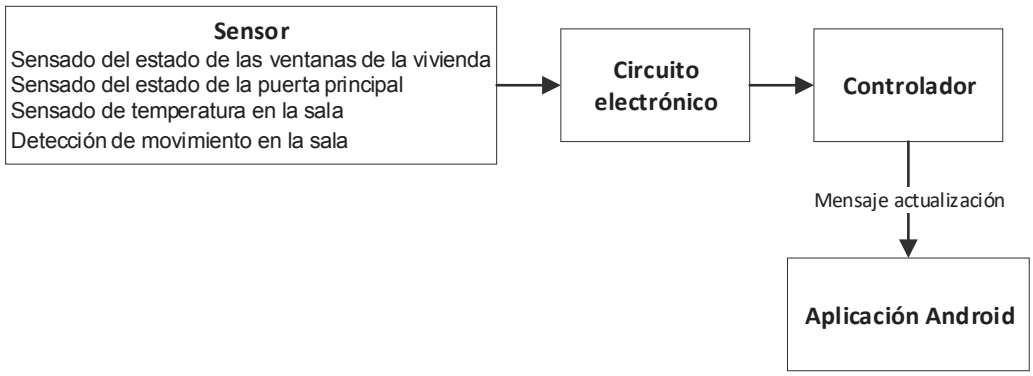

Figura 2.39. Diagrama de sensado de variables

2.1.6.3. Control de dispositivos

El prototipo de sistema domótico posibilita el encendido o apagado de las luminarias y la radio, además de la activación de una sirena mediante el botón de pánico. El control de dispositivos se especifica en el diagrama de bloques de la Figura 2.40 y se describe a continuación:

- El usuario controla la funcionalidad del sistema desde la interfaz de consola de la aplicación móvil. Cuando el usuario activa o desactiva un elemento (como una luminaria) o una función del sistema (como la alarma), la aplicación móvil envía un mensaje de actualización al controlador indicando la acción realizada.

- El controlador recibe el mensaje de actualización, lo interpreta, y genera una determinada salida en uno de sus pines. El circuito electrónico asociado a dicho pin del controlador posibilita enviar una orden a un actuador para encender o apagar determinado dispositivo, como una luminaria, radio, o sirena.

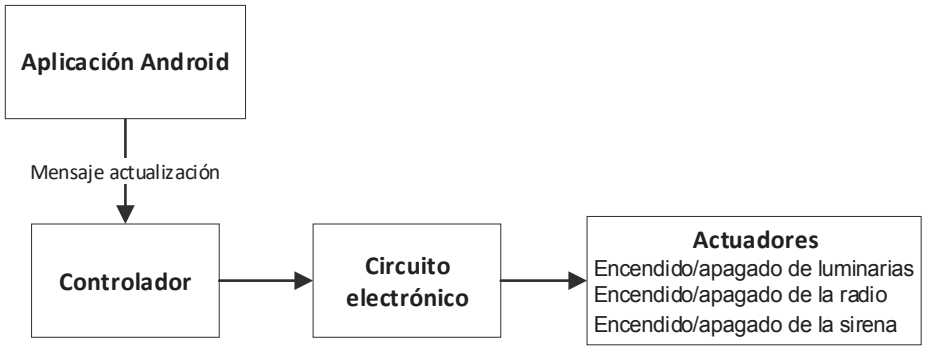

Figura 2.40. Diagrama de control de dispositivos

2.2. Fase de implementación

En esta sección se detalla el proceso de implementación de la aplicación para el controlador Arduino, la implementación de la aplicación móvil y el ensamblaje del prototipo de sistema domótico.

2.2.1. Implementación de la aplicación móvil

A continuación, se describen las diferentes interfaces asociadas a la aplicación móvil Android y la estructura de los archivos que forman parte de la misma. El código fuente de la aplicación se incluye en el Anexo VIII.

2.2.1.1. Pantalla de inicio

La pantalla de inicio se muestra por unos segundos cuando el usuario abre la aplicación en su teléfono.

Es utilizada para dar a conocer el nombre de la aplicación, denominada “Casa Inteligente”, como se observa en la Figura 2.41.

2.2.1.2. Interfaz de conexión

Después de la pantalla de inicio se muestra la interfaz de conexión, que se observa en la Figura 2.42.

Figura 2.41. Pantalla de inicio

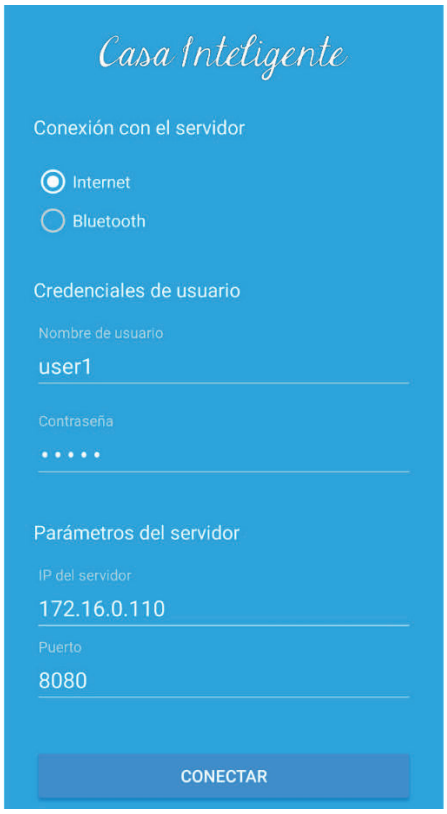

Figura 2.42. Interfaz de conexión

Una vez que se ha ingresado el usuario y contraseña -además de la IP y puerto del servidor al conectarse a través de Internet- se presiona sobre conectar, lo que genera un cuadro de

progreso para indicar que la aplicación está tratando de conectarse al servidor, como se indica en la Figura 2.43.

Figura 2.43. Diálogo de progreso en la interfaz de conexión

Si la autenticación fue fallida, se muestra en la pantalla una notificación *toast* indicando lo sucedido, como se observa en la Figura 2.44.

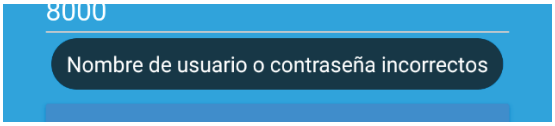

Figura 2.44. Notificación de fallo en la autenticación

Al contrario, si la autenticación fue exitosa, se muestra al usuario la interfaz de consola.

2.2.1.3. Interfaz de consola

Figura 2.45. Interfaz de consola (Parte 1 de 3)

Figura 2.45. Interfaz de consola (Parte 2 de 3)

Figura 2.45. Interfaz de consola (Parte 3 de 3)

Una vez autenticado, el usuario puede gestionar el estado de todo el sistema desde esta interfaz de forma intuitiva.

Los componentes de la interfaz de consola asociados al control de las luces en las diferentes estancias de la vivienda, control de la alarma, simulación de presencia y encendido de la radio, se muestran en la Figura 2.45.

Además, la interfaz indica el estado de la sirena, detección de movimiento, temperatura, y especifica el estado de los accesos en la vivienda, indicando si la puerta y las ventanas permanecen abiertas o cerradas.

El botón flotante incluido en la parte inferior derecha de la interfaz controla la función de botón de pánico.

2.2.1.4. Interfaz de configuración de cámara

Esta interfaz contiene un botón flotante en la esquina inferior derecha que controla la función de botón de pánico y el texto “acceder a la cámara”, como se indica en la Figura 2.46.

Figura 2.46. Interfaz de configuración de cámara

Al presionar sobre el texto se intenta abrir una aplicación externa al sistema que permite configurar parámetros de la cámara IP como la visión nocturna, la captura de fotografías, permite visualizar el video captado por la cámara, entre otros. Si la aplicación externa no se encuentra instalada en el dispositivo se muestra un mensaje indicando que no se puede

realizar esa acción. La aplicación para configurar la cámara IP se puede descargar de la página oficial.[6]

2.2.1.5. Panel lateral de navegación

Las interfaces de consola y la de configuración de la cámara poseen una barra situada en la parte superior de la interfaz, como se muestra en la Figura 2.47. Esta barra de aplicación o *app bar,* contiene un ícono seguido del texto “Consola”. El ícono permite agregar la capacidad de navegación y cambios de vistas, mientras que el texto indica al usuario que se encuentra en la interfaz de gestión del sistema domótico.

Figura 2.47. Barra de aplicación

El panel lateral de navegación que se muestra en la Figura 2.48 permite desplazarse entre las diferentes interfaces o salir de la aplicación.

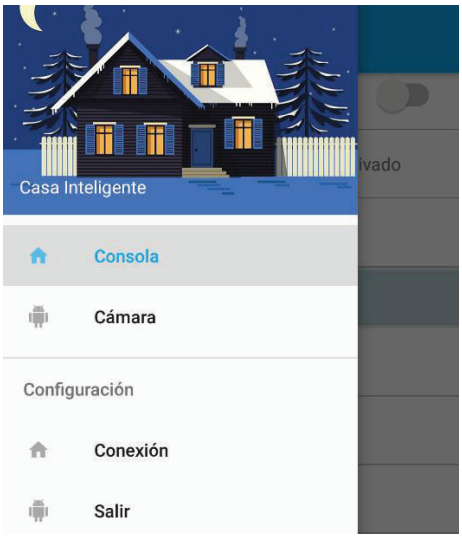

Figura 2.48. Panel lateral de navegación

El panel de navegación permanece oculto por defecto, pero se puede acceder al mismo al presionar sobre el ícono de la *app bar* o al deslizar un dedo desde el borde izquierdo de la pantalla.

[6] Aplicación Android mydlink, https://play.google.com/store/apps/details?id=com.dlink.mydlink&hl=es

2.2.1.6. Notificaciones

Se emplean cuadros de diálogo para notificar al usuario fallos en la conexión con el servidor, operaciones inválidas -como tratar de encender una luminaria al estar activada la simulación de presencia- y para alertar la apertura de una puerta o ventana una vez activada la alarma en la vivienda. Este tipo de notificaciones no utiliza toda la pantalla y requiere la acción del usuario para poder continuar, como se indica en la Figura 2.49.

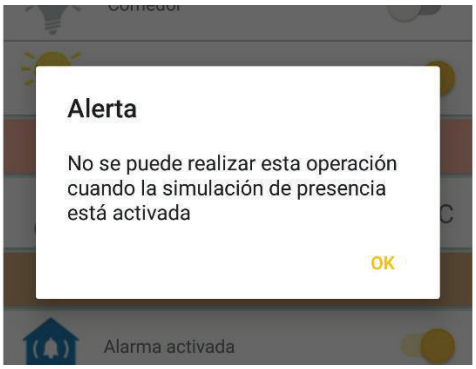

Figura 2.49. Notificación mediante cuadro de diálogo

Se emplea un *toast* para brindar información simple al usuario sobre alguna operación realizada, como al activar el botón de pánico, al presionar el botón *back* de la barra de navegación para cerrar la aplicación, al producirse un fallo en la autenticación, al no ser posible conectar con el servidor, entre otros. Un *toast* solo ocupa el espacio requerido para mostrar el mensaje y desaparece automáticamente después de un tiempo, como se indica en la Figura 2.50.

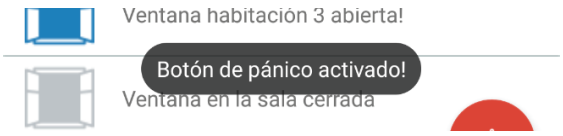

Figura 2.50. Notificación mediante un *toast*

2.2.1.7. Errores de conexión

Cuando el usuario presiona salir en el panel lateral de navegación, la aplicación móvil se cierra, y se produce el cierre de la conexión con Arduino, es decir, se produce un intercambio de mensajes de FIN y ACK entre Arduino y la aplicación móvil para cerrar la conexión TCP.

Al producirse un fallo, como al desconectar un cable de red, desconectar a la placa Arduino de la energía eléctrica de forma abrupta, al perder acceso a Internet en la vivienda, entre

otros, no se produce un cierre de la conexión como en condiciones normales. Este escenario, en donde uno de los extremos sabe que la conexión se perdió, pero el otro extremo cree que la conexión sigue activa, se conoce como *half-open connection* [73]. Para detectar esta anomalía, se requiere enviar o recibir información con el dispositivo al otro extremo de la conexión para verificar si este permanece activo, y por ende, comprobar si la conexión permanece establecida.

Para detectar fallas en la conexión entre el controlador y el dispositivo móvil se implementó un mecanismo similar al utilizado por TCP, conocido como TCP *keep-alive*. Dicho mecanismo se basa en el envío de un segmento TCP - denominado *keep-alive* - de forma periódica al dispositivo en el otro extremo de la conexión. El dispositivo que recibe el mensaje responde con un ACK, lo que indica al dispositivo que envió el *keep-alive* que el otro extremo permanece disponible y la conexión sigue establecida [74].

El intervalo de tiempo para el envío del *keep-alive* depende de la plataforma o aplicación donde se implemente. El RFC 1122 define un intervalo por defecto de 2 horas [75, p. 101], motivo por el cual sistemas operativos como Linux y Windows emplean por defecto ese intervalo [76]. Dicho RFC fue escrito en 1989, por lo que se considera que la utilización de intervalos de tiempo menores es mucho más útil en las redes modernas. Es así, que es común encontrar implementaciones que definen intervalos para el envío de mensajes de *keep-alive* de 45 o 60 segundos [77].

De manera análoga al *keep-alive* de TCP, Arduino envía un mensaje de actualización de forma periódica a la aplicación móvil, que además de indicar la temperatura en la vivienda, es utilizado para verificar que la conexión entre el dispositivo móvil y Arduino permanece establecida. Si la aplicación móvil no recibe el mensaje de actualización de forma periódica, se asume que se ha producido un error en la conexión con Arduino. Por lo tanto, se notifica al usuario lo sucedido, para que intente conectarse nuevamente al controlador, empleando el Internet o Bluetooth, como se indica en la Figura 2.51.

Al igual que en el caso del *keep-alive*, el intervalo de tiempo para el envío de los mensajes de actualización depende de la aplicación en la cual se implementa. Existe un compromiso entre el tiempo que toma detectar una falla en la conexión y el tráfico que se genera, por lo que el intervalo de tiempo más adecuado varía en función de lo que se desea conseguir. Al seleccionar un intervalo de tiempo de un par de segundos para el envío de los mensajes de actualización, se verificó que es posible detectar una falla en la conexión de forma rápida, pero el tráfico que se genera es mayor en comparación al generado al elegir un intervalo de unos minutos. Después de realizar pruebas con intervalos de tiempo en el

orden de los segundos y minutos, se decidió seleccionar un intervalo de 40 segundos, similar al empleado para el envío de mensajes *keep-alive* en algunas implementaciones.

Se puede estimar el tráfico generado por el envío de los mensajes de actualización considerando que el mensaje a enviar está formado por 3 bytes. El mensaje es encapsulado en un segmento TCP, cuya cabecera es de 20 bytes. El segmento TCP es encapsulado en un datagrama IP, cuya cabecera es de 20 bytes. A su vez, el paquete IP es encapsulado en una trama Ethernet, que añade un *overhead* de 18 bytes. De esta manera la trama resultante tiene un tamaño de 61 bytes. Estos 61 bytes o 488 bits son enviados por Arduino cada 40 segundos, por lo que se requiere una conexión a Internet de al menos 488/40 bps o aproximadamente 12 bps, un valor muy por debajo del promedio de velocidad de una conexión fija a Internet en el Ecuador, que para el año 2017 era de 6,2 Mbps [78].

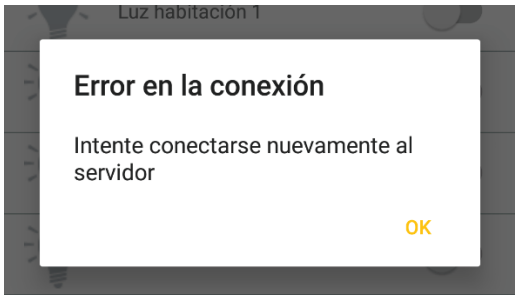

Figura 2.51. Notificación de fallo en la conexión

2.2.1.8. Archivos asociados a las interfaces

Cada ventana de la aplicación está asociada a dos archivos, una clase de extensión cs que se extiende de la clase Activity y un archivo de extensión axml.

La clase Activity se encarga de crear la ventana en la cual se colocan los diferentes componentes gráficos, por tal motivo se dice que cada ventana o interfaz de la aplicación se encuentra representada por una *activity.* La aplicación contiene varias ventanas, lo que quiere decir que contiene diferentes actividades.

En el archivo de extensión axml se define el *layout* o estructura visual de la interfaz de usuario, incluyendo los elementos que aparecen en la pantalla y sus propiedades.

Los archivos asociados a la aplicación móvil se muestran en la Figura 2.52 y se describen en el Anexo IX.

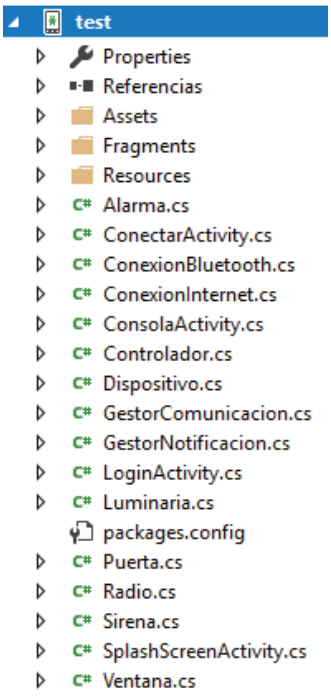

Figura 2.52. Estructura de archivos

2.2.1.9. Comunicación con Arduino

La aplicación móvil para Android fue desarrollada empleando el lenguaje de programación C# y la plataforma Xamarin. El envío y recepción de información en la aplicación se realiza a través de Internet o Bluetooth, como se describe a continuación.

Comunicación mediante Internet

Para el intercambio de información a través de Internet se emplean *sockets*. En el Código 2.3 se muestra el proceso de establecimiento de la conexión con el servidor Arduino, el cual se detalla a continuación. Para empezar, se crea una instancia de la clase `Socket`. Se inicializa la instancia pasando al constructor de la clase `socket` los siguientes parámetros, que describen la forma en que el *socket* envía información:

- **`AddressFamily`:** Especifica el esquema de direccionamiento que la instancia de la clase `socket` puede utilizar. Controla el protocolo utilizado en la capa de red de acuerdo al modelo OSI[7]. Por ejemplo, `InterNetwork` al emplear IP versión 4 o `InterNetworkV6` al emplear IP versión 6. Los parámetros `SocketType` y `ProtocolType` son seleccionados dependiendo del valor de `AddressFamily`.

[7] El modelo OSI (*Open Systems Interconnection*) define las fases por las que atraviesan los datos al ser enviados de un dispositivo a otro sobre una red de comunicaciones.

- **`SocketType`:** Determina el tipo de *socket* asociado a la instancia de la clase `Socket`. En este caso, este parámetro indica de forma implícita el valor de `ProtocolType`. Al seleccionar `stream` como el tipo de *socket* a emplear, el valor de `ProtocolType` es siempre `Tcp`.

- **`ProtocolType`:** Especifica el protocolo empleado por el *socket*. Se selecciona `Tcp` como el protocolo a utilizar en la capa de transporte.

Se emplea la clase `IPEndPoint` para representar uno de los extremos de la conexión, en este caso, el controlador Arduino, que actúa como servidor. La IP y puerto del servidor se pasan como parámetros al constructor de la clase `IPEndPoint`. El método `socket.BeginConnect()` inicia una petición de conexión al dispositivo Arduino. Recibe como parámetros el `IPEndPoint` que representa el dispositivo remoto, un método a ser llamado cuando la operación de `BeginConnect()` se complete, y un objeto que contiene información de la petición.

Se emplea `result.AsyncWaitHandle.WaitOne(5000, true)` para especificar un *timeout* para el intento de conexión, es decir, especificar el tiempo máximo disponible para el establecimiento de la conexón TCP entre el cliente (dispositivo móvil) y el servidor (Arduino). Una vez iniciado el proceso de establecimiento de la conexión se cuenta 5000 ms. Transcurrido ese tiempo, se evalúa si la conexión con el servidor se estableció o no, empleando para ello la propiedad `socket.Connected`. Si la conexión ha sido establecida, se llama al método `socket.EndConnect()` para completar la petición de conexión iniciada con `BeginConnect()`. Si el intento de conexión ha fallado, se llama al método `socket.Close()` para liberar los recursos asociados al *socket* .

```
// Se inicializa el socket
Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
// Especifico IP y puerto del servidor
IPEndPoint servidor = new IPEndPoint(IPAddress.Parse(txtIpServidor.Text),
Int32.Parse(txtPuertoServidor.Text));
// Inicia el proceso de establecimiento de conexión con el servidor
IAsyncResult result = socket.BeginConnect(servidor, null, null);
// Especifico un timeout de 5 segundos para el intento de conexión
bool success = result.AsyncWaitHandle.WaitOne(5000, true);
// Conexión con el servidor establecida
if (socket.Connected) { socket.EndConnect(result);}
// No se puede conectar al servidor
else { socket.Close();}
```

Código 2.3. Establecimiento de la conexión a través de Internet

El envío de información en la aplicación móvil se muestra en el Código 2.4 y se detalla a continuación. El mensaje a enviar es codificado con ASCII y almacenado en la variable

`datos`. La clase `NetworkStream` provee métodos para el envío y recepción de datos en *sockets* de tipo `stream`. Se emplea el método `flujoDatos.Write()` para enviar el mensaje al controlador Arduino en el otro extremo de la conexión. El método `Write()` recibe como parámetros una matriz de bytes llamada `datos`, un entero que representa el índice del elemento de la matriz `datos` desde el cual se realiza el envío, y el número de bytes a enviar.

```
string mensaje = "010";
byte[] datos = Encoding.ASCII.GetBytes(mensaje);
NetworkStream flujoDatos = new NetworkStream(socket);
flujoDatos.Write(datos, 0, datos.Length);
```

Código 2.4. Envío de datos desde la aplicación móvil

La recepción de información en la aplicación móvil se muestra en el Código 2.5. Se declara una matriz de 24 bytes denominada `buffer` para almacenar el mensaje recibido. Se utiliza el método `flujoDatos.Read()` para leer los datos recibidos. El método `Read()` recibe como parámetros una matriz denominada `buffer` para almacenar los datos leídos, el índice en la matriz `buffer` desde donde se empieza a almacenar los datos recibidos, y el número de bytes leídos.

```
byte[] buffer = new byte[24];
int numBytesRecibidos = flujoDatos.Read(buffer, 0, buffer.Length);
string mensaje = Encoding.ASCII.GetString(buffer);
```

Código 2.5. Recepción de datos en la aplicación móvil

Comunicación mediante Bluetooth

El dispositivo Android se debe conectar al módulo Bluetooth de Arduino antes de intercambiar información.

Para establecer una conexión entre ambos dispositivos, se requiere que el dispositivo Android y el módulo Bluetooth estén emparejados. El proceso para verificar que ambos dispositivos están emparejados se describe en el Código 2.6 que se detalla a continuación.

Se declara un objeto del tipo `BluetoothDevice` para representar al dispositivo remoto, es decir, al módulo Bluetooth HC05 de Arduino. Para consultar la lista de dispositivos emparejados se emplea la propiedad `mBluetoothAdapter.BondedDevices`, que retorna una lista de objetos del tipo `BluetoothDevice`, cada uno de los cuales representa un dispositivo emparejado.

Posteriormente se verifica si alguno de los dispositivos emparejados tiene como nombre “CasaDomo”. Si alguno de los dispositivos emparejados posee ese nombre, se trata del módulo Bluetooth HC05.

```
// Se crea un objeto que represente el módulo Bluetooth de Arduino
BluetoothDevice dispositivoBluetooth = null;
// Se obtiene la lista de dispositivos emparejados
ICollection<BluetoothDevice> pairedDevices = mBluetoothAdapter.BondedDevices;
// Verifica si existen dispositivos emparejados
if (pairedDevices.Count > 0) {
   // En este lazo se verifica el nombre de cada dispositivo emparejado
foreach (BluetoothDevice device in pairedDevices) {
   String deviceName = device.Name;
   // Se crea el objeto que hace referencia al módulo HC05
   if (deviceName == "CasaDomo")
      dispositivoBluetooth = device;
   }
}
```

Código 2.6. Verificación de dispositivos Bluetooth emparejados

En el Código 2.7 se muestra el proceso de conexión entre el módulo Bluetooth y el dispositivo Android, como se describe a continuación.

Una vez que se ha obtenido el objeto `BluetoothDevice` que represente al dispositivo remoto, se puede iniciar la conexión. Se emplea el método `CreateRfcommSocketToServiceRecord()` para inicializar un objeto `BluetoothSocket`, que permite al dispositivo Android conectarse al módulo Bluetooth. Dicho método recibe como parámetro el UUID (*Universally Unique Identifier*), el cual identifica el servicio brindado por un dispositivo Bluetooth. Es utilizado para identificar el servicio al cual desea conectarse un dispositivo. Una vez que se ha inicializado el objeto `miSocket`, se llama al método mi `miSocket.Connect()` para iniciar la conexión. El método `Connect()` retorna sin excepciones cuando la conexión ha sido establecida. Si el método retorna una excepción, no se pudo conectar con el dispositivo remoto.

```
Java.Util.UUID miUUID=Java.Util.UUID.FromString("00001101-0000-1000-8000-
00805F9B34FB");
BluetoothSocket miSocket=dispositivoBluetooth.CreateRfcommSocketToServiceRecord
(miUUID);
try {
   // Conectamos con el dispositivo remoto a través del socket
   miSocket.Connect();
}
catch (Java.IO.IOException connectException) {
   // No se puede conectar, se cierra el socket
   miSocket.Close();
}
```

Código 2.7. Establecimiento de la conexión por Bluetooth

Una vez que el dispositivo Android y el módulo Bluetooth de Arduino están conectados, se puede iniciar la transferencia de información.

En el Código 2.8 se observa el proceso de envío de información en el dispositivo Android utilizando Bluetooth. Es necesario crear un objeto del tipo `Stream` llamado `flujoSalida`, el cual brinda los métodos necesarios para el envío de información. Después, se define el mensaje a enviar, el cual es codificado con ASCII y almacenado en una matriz denominada datos. Se emplea el método `Write()` para enviar el mensaje. Este método recibe como parámetros una matriz de bytes con el mensaje a enviar, un entero que representa el índice de la matriz `datos` desde donde se inicia el envío, y un entero que representa el número de bytes a enviar.

```
Stream flujoSalida = miSocketB.OutputStream;
string mensaje = "611";
byte[] datos = Encoding.ASCII.GetBytes(mensaje);
flujoSalida.Write(datos, 0, datos.Length);
```

Código 2.8. Envío de datos por Bluetooth en la aplicación móvil

En el Código 2.9 se muestra el proceso de recepción de información en el dispositivo Android mediante Bluetooth. Se crea un objeto llamado `flujoEntrada`, el cual es utilizado para la recepción de información. Posteriormente, se define una matriz de 24 bytes para almacenar el mensaje recibido. Se usa el método `Read()` para leer los datos recibidos. Este método recibe como parámetros una matriz llamada `buffer` en donde se ubican los bytes leídos, un entero que representa el índice de la matriz `buffer` desde donde se empieza a almacenar el mensaje recibido, y un entero que representa el número de bytes leídos. Se emplea el método `Encoding.ASCII.GetString()` para convertir los bytes recibidos en una secuencia de caracteres ASCII.

```
Stream flujoEntrada = miSocketB.InputStream;
byte[] buffer = new byte[24];
int numBytesRecibidos = flujoEntrada.Read(buffer, 0, buffer.Length);
string mensajeServidor = Encoding.ASCII.GetString(buffer);
```

Código 2.9. Recepción de datos por Bluetooth en la aplicación móvil

Para evitar el bloqueo del hilo principal que gestiona la interfaz de usuario se emplean hilos trabajadores, desde los cuales se realiza el envío y recepción de información como se describió anteriormente, tanto al emplear *sockets* para la comunicación a través de Internet, como al utilizar Bluetooth.

2.2.2. Implementación del programa de Arduino

La placa Arduino Mega 2560 es la encargada de brindar al sistema domótico la capacidad de sensado de variables, control de dispositivos y comunicación con el cliente. El código fuente del programa para Arduino se incluye en el Anexo X.

2.2.2.1. Estructura del programa para la plataforma Arduino Mega 2560

La plataforma Arduino actúa como la unidad central o cerebro del sistema domótico. La Figura 2.53 describe a través de un diagrama el proceso por el cual el controlador Arduino lleva a cabo las funciones de encendido/apagado de luminarias y radio, medición de temperatura, sensado de apertura/cierre de puertas y ventanas, sensado de movimiento, la simulación de presencia, botón de pánico y comunicación con la aplicación móvil para Android.

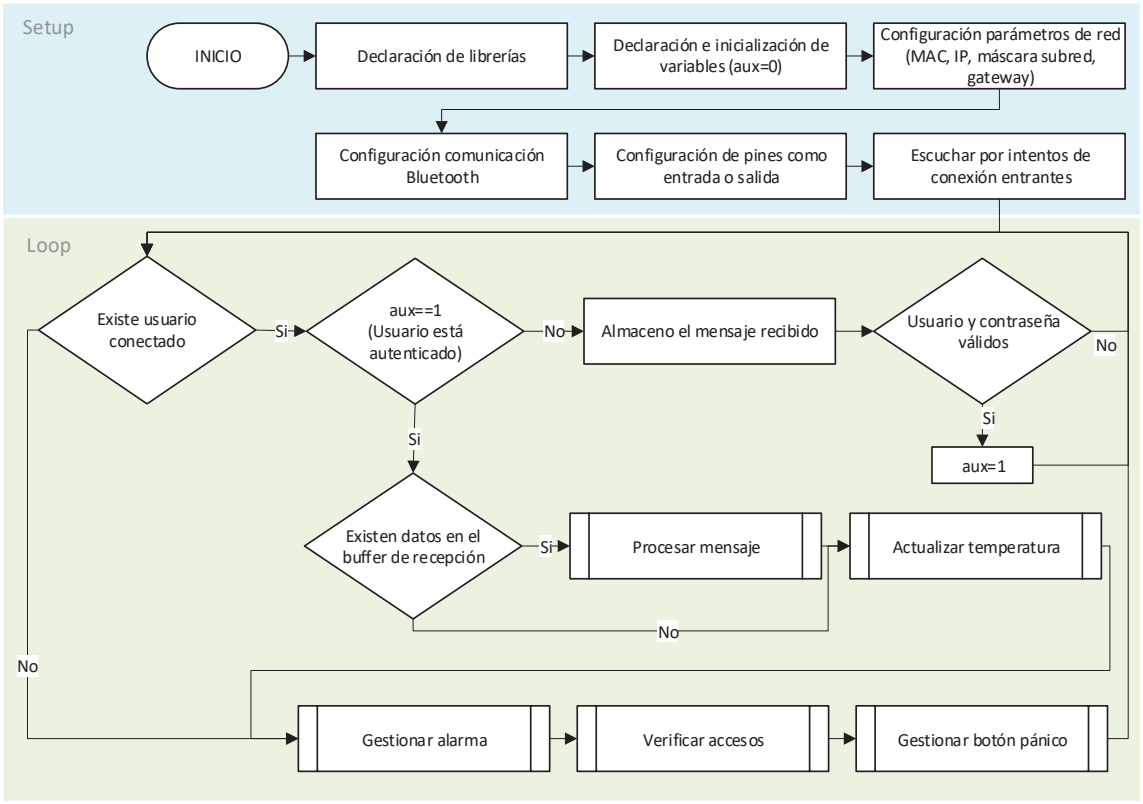

Figura 2.53. Estructura programa Arduino Mega 2560

A continuación, se describen los subprocesos que permiten al controlador llevar a cabo las funciones de monitoreo y control.

Subproceso Procesar Mensaje

Interpreta el mensaje recibido y ejecuta una acción de acuerdo al contenido de dicho mensaje. Para ello se analiza el significado de cada uno de los 3 caracteres que forman el mensaje de actualización. En respuesta a ese mensaje, se genera un 1L o un 0L en un determinado pin de salida del controlador, lo que posibilita mediante un circuito, encender o apagar un dispositivo. La estructura del subproceso se describe en la Figura 2.54.

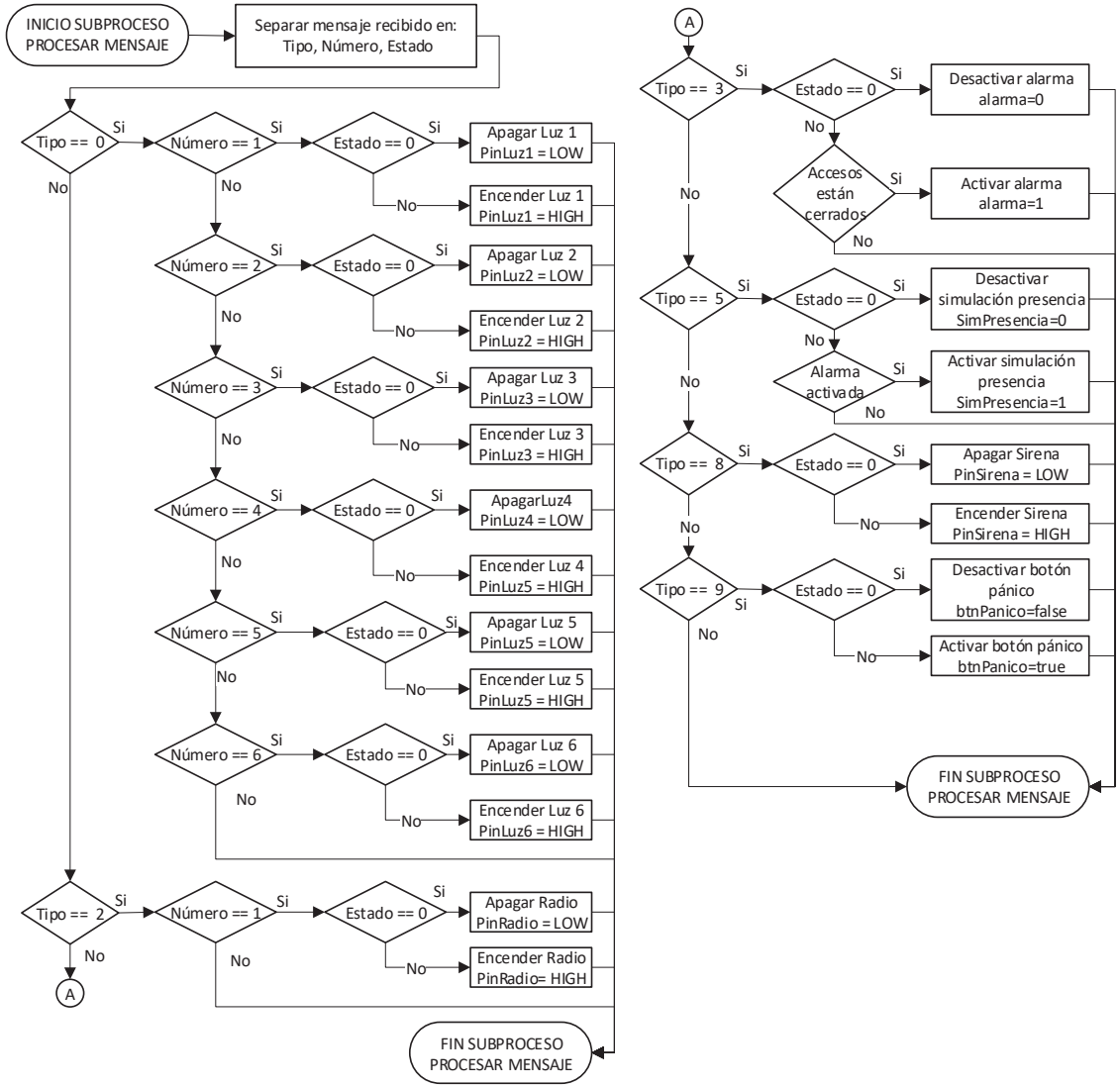

Figura 2.54. Subproceso procesar mensaje

Subproceso Actualizar Temperatura

A continuación, se detalla el mecanismo para notificar al usuario la temperatura en la vivienda a través de mensajes de actualización.

Para empezar, se verifica si el tiempo transcurrido desde la última actualización de temperatura es igual o mayor al valor dado por la variable `intervalo` (40000 ms). Si el tiempo transcurrido es menor, no se genera ningún mensaje. Si el tiempo trascurrido es mayor o igual, se almacena en la variable `lectura` el valor entregado por el sensor de temperatura en el pin analógico correspondiente.

Se procede a calcular la temperatura en °C utilizando el valor `lectura` obtenido del sensor. De acuerdo con la hoja de datos del sensor LM35 [59], se puede calcular la temperatura a

partir del voltaje generado por el sensor a través de la siguiente función lineal: $T = \frac{Vsensor\ (mV)}{10\ mV}$ °C. Además, dado que el convertidor análogo-digital de Arduino maneja 10 bits, puede asignar uno de entre 1024 posibles valores. Considerando como voltaje de referencia 5 V (5000 mV), es posible obtener el voltaje en el pin analógico de Arduino de la siguiente manera: $Vsensor = \frac{\text{lectura}}{1024} x\ 5000\ mV$. Por lo tanto, es posible obtener la temperatura al realizar la siguiente operación: $T = \frac{\text{lectura}}{1024} x\ 500$ (°C).

Una vez obtenida la temperatura, se envía un mensaje al dispositivo móvil del usuario con la temperatura calculada. La función `millis()` retorna el tiempo en milisegundos desde que la placa Arduino empezó a ejecutar el programa, valor que es asignado a la variable `Tiempo Actual`. Se asigna a `Tiempo Previo` el valor de `Tiempo Actual` después de enviar un mensaje de actualización de temperatura, lo que ocasiona que se empiece a contar nuevamente hasta que el tiempo transcurrido sea igual o mayor al `intervalo`. Esto genera el envío de un nuevo mensaje de actualización de temperatura cada 40 segundos. La estructura del subproceso descrito se observa en la Figura 2.55.

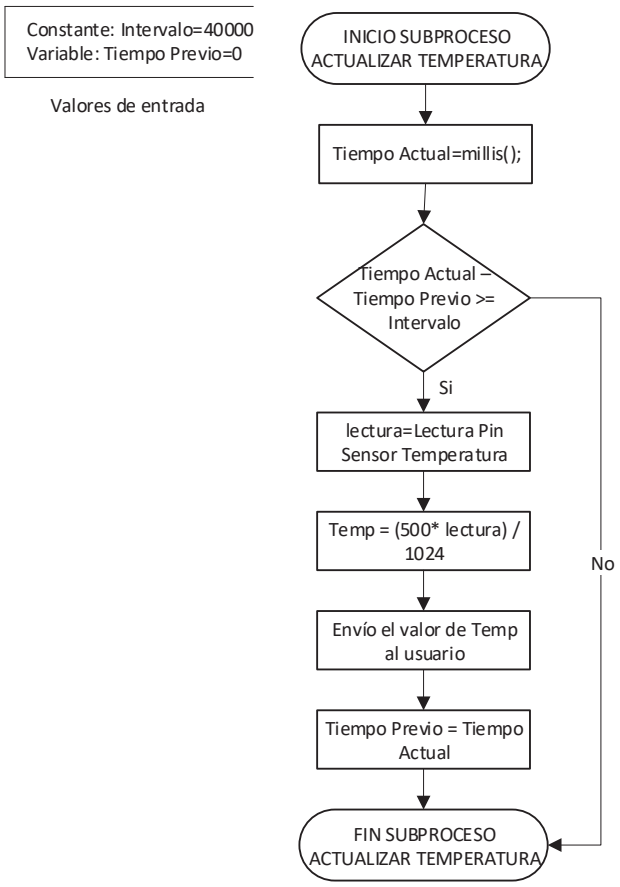

Figura 2.55. Subproceso actualizar temperatura

Subproceso Gestionar Alarma

El subproceso para gestionar la función de alarma se describe en la Figura 2.56.

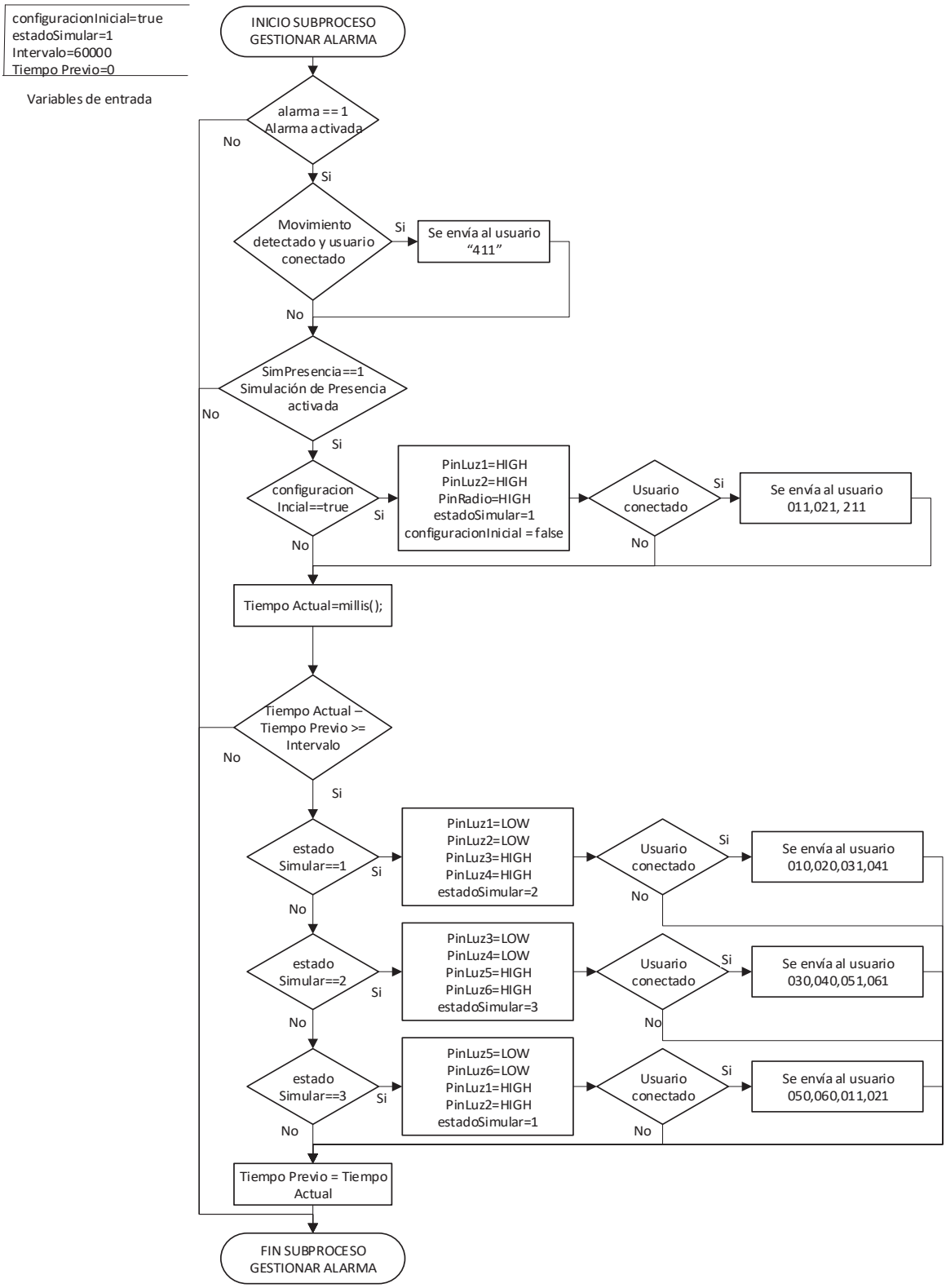

Figura 2.56. Subproceso gestionar alarma

Para que las funciones de detección de movimiento y simulación de presencia puedan ser activadas, es necesario que la alarma esté activada.

Al detectarse movimiento se envía una notificación al usuario, a través del mensaje de actualización 411. Si la alarma no está activada, no se notifica al usuario.

Si la simulación de presencia está activada en la vivienda, el sistema enciende y apaga las luminarias en la vivienda de forma automática, alternando entre 3 estados posibles, como se muestra en la Figura 2.57:

- **Estado 1:** Se encienden las luminarias de la habitación 1 y habitación 2 por un lapso de un minuto. Para ello se colocan en HIGH los pines de Arduino `PinLuz1` y `PinLuz2` respectivamente. Se notifica al usuario los elementos activados mediante el envío de los mensajes 011 y 021.
- **Estado 2:** Se encienden las luminarias de la habitación 3 y de la sala durante un minuto. Para ello se colocan en HIGH los pines de Arduino `PinLuz3` y `PinLuz4` respectivamente. Se notifica los elementos activados mediante el envío de los mensajes 031 y 041.
- **Estado 3:** Se encienden las luminarias del comedor y del baño durante un minuto. Para ello se colocan en HIGH los pines de Arduino `PinLuz5` y `PinLuz6` respectivamente. Se notifica los elementos activados mediante el envío de los mensajes 051 y 061.

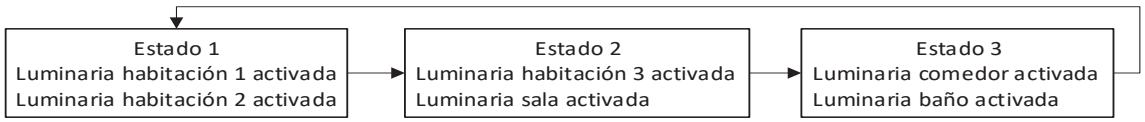

Figura 2.57. Diagrama de simulación de presencia

Por lo tanto, al activar la simulación de presencia, el sistema pasa al Estado 1, le sigue el Estado 2, y después el Estado 3. Transcurrido un minuto el sistema regresa al Estado 1 y el ciclo se repite hasta que el usuario desactive la simulación de presencia. El tomacorriente de la radio, asociado al pin `PinRadio` en el controlador Arduino, permanece activo en los 3 estados. Se mide si el tiempo transcurrido es mayor o igual a un minuto empleando las variables `Tiempo Previo`, `Tiempo Actual` e `intervalo`.

Mientras la simulación de presencia permanezca activada, las luces se encienden o apagan de forma automática, por lo que el usuario no puede cambiar su estado desde la aplicación móvil. Al desactivar la simulación de presencia, el usuario recupera el control de

las luminarias. Si la función de alarma es desactivada, se desactivan las funciones de detección de movimiento, alertas por apertura de puertas o ventanas y la simulación de presencia.

Subproceso Gestionar Botón de Pánico

El subproceso para gestionar el botón de pánico se muestra en la Figura 2.58.

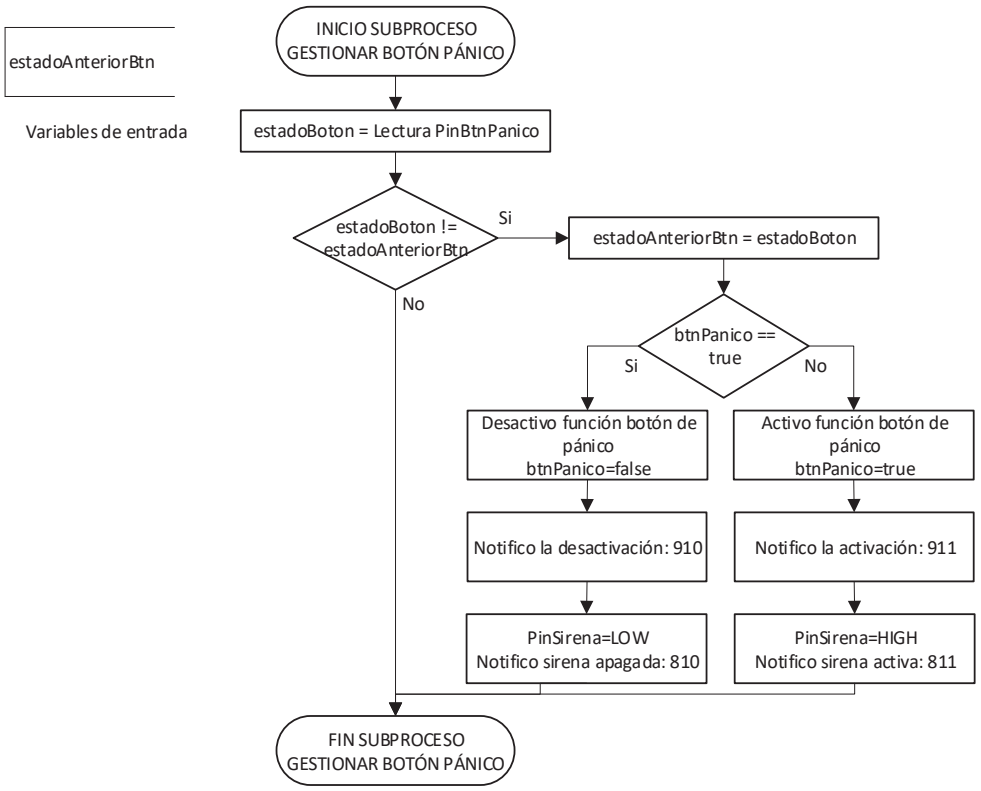

Figura 2.58. Subproceso gestionar botón de pánico

Se activa o desactiva la función de botón de pánico cuando un usuario presiona un pulsador. Para empezar, es necesario conocer el estado del pulsador, es decir, verificar si está presionado. Para ello se realiza la lectura del pin de Arduino denominado `PinBtnPanico`. Si el pulsador está presionado `estadoBoton` es 0, caso contrario tiene el valor de 1. Se compara el estado actual del pulsador con el estado anterior registrado, si estos son diferentes, se sabe que el pulsador ha sido presionado y se procede a almacenar el nuevo estado del botón. Después de determinar que el pulsador fue presionado, se verifica si la función de botón de pánico esta activada o desactivada. Dado el caso que la función de botón de pánico esté activada, al presionar el pulsador esta es desactivada. Una vez desactivada se envía un mensaje con el código 910 a la aplicación móvil para notificar lo sucedido, además, se apaga la sirena y se notifica al usuario con el código 810. Si la

función de botón de pánico se activa, se notifica al usuario con el código 911, se enciende la sirena y se envía un mensaje con el código 811 a la aplicación móvil.

Subproceso Verificar Accesos

El monitoreo del estado de la puerta y las ventanas se describe en la Figura 2.59.

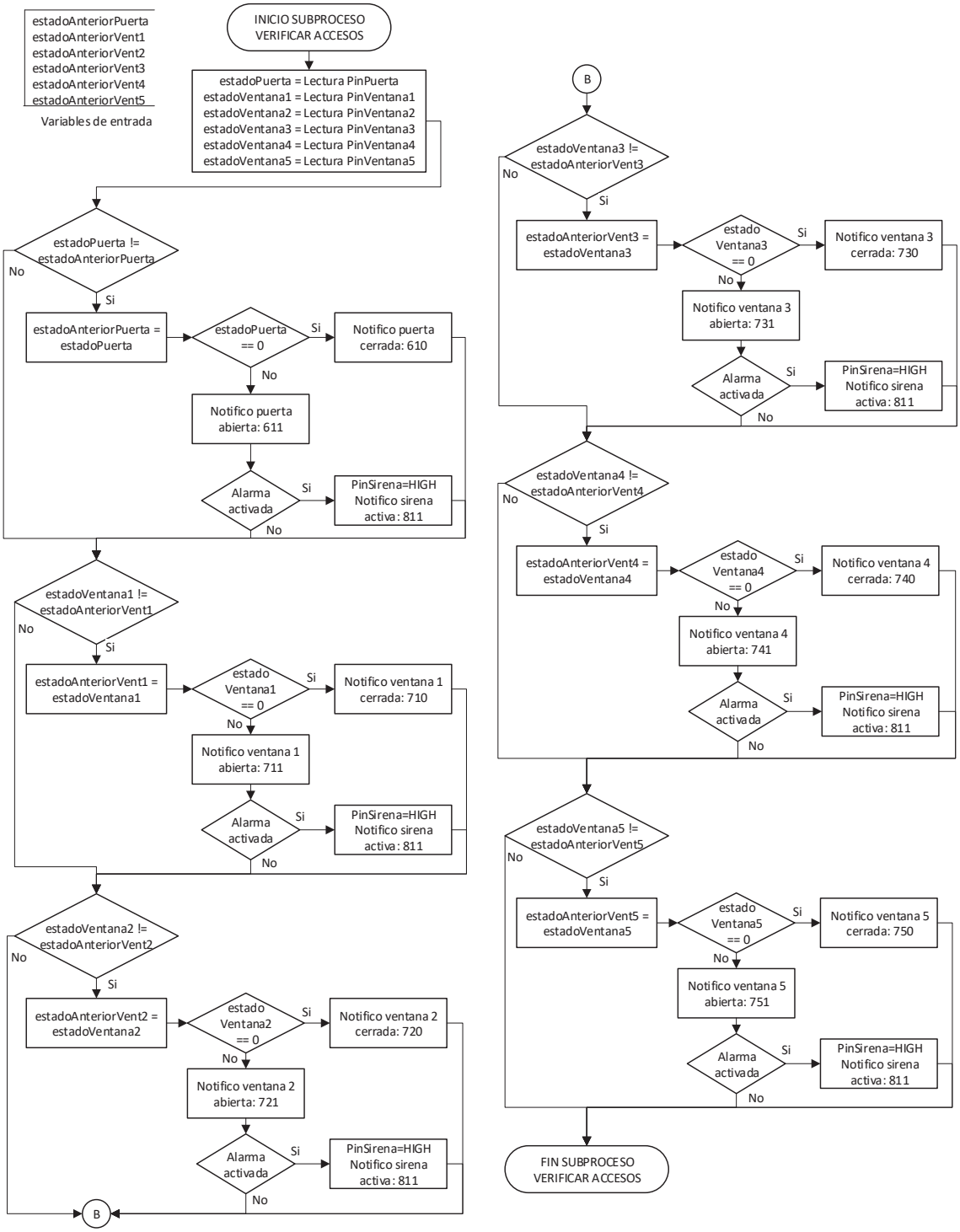

Figura 2.59. Subproceso verificar accesos

Para conocer si una puerta o ventana está abierta o cerrada se realiza la lectura de los pines de Arduino asociados a los sensores magnéticos. Si una puerta/ventana que permanecía abierta es cerrada, o al contrario, si estando cerrada es abierta, se envía un mensaje de actualización a la aplicación móvil para notificar al usuario.

Si la alarma está activada al momento en que un acceso -puerta o ventana- se abre, se genera una alerta sonora en la vivienda empleando una sirena.

2.2.2.2. Comunicación con la aplicación móvil

A continuación, se describe el proceso de envío y recepción de información con la aplicación móvil para Android.

Comunicación mediante Internet

Para el intercambio de información a través de Internet, se emplean las clases `EthernetServer`, `EthernetClient` y `Ethernet`.

Como se muestra en el Código 2.10, se definen la dirección MAC, dirección IP, la dirección del *gateway* y la máscara de subred para Arduino. Se crea una instancia de la clase `EthernetServer` llamada `server`, y se pasa como parámetro al constructor de dicha clase el número de puerto en el cual el controlador escuchará por peticiones de conexión entrantes.

Para asociar la placa Arduino a los parámetros de red, se emplea el método `begin` de la clase `Ethernet`. Posteriormente, se emplea una llamada al método `server.begin()` para empezar a escuchar por peticiones de conexión en el puerto 8000.

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 172, 16, 0, 5 };
byte gateway[] = { 172, 16, 0, 1 };
byte subnet[] = { 255, 255, 255, 0 };
EthernetServer server(8000);
Ethernet.begin(mac, ip, gateway, gateway, subnet);
server.begin();
```

Código 2.10. Configuración de parámetros de red en Arduino

Una vez que un cliente se ha conectado al servidor Arduino, se emplea el método `server.available()`, el cual retorna un *socket* asociado a la conexión con ese cliente.

El *socket* es almacenado en un objeto llamado `client` del tipo `EthernetClient`. Para el envío de información al cliente, se emplea el método `client.print("mensaje")`, como se muestra en el Código 2.11.

```
EthernetClient client;
client = server.available();
client.print("mensaje");
```

Código 2.11. Envío de datos desde Arduino

La recepción de información se muestra en el Código 2.12. Es necesario declarar una variable del tipo `String` llamada `miMensaje`, la cual almacenará el comando enviado desde la aplicación móvil.

El método `client.available()` retorna el número de bytes disponibles para lectura, lo que permite leer los datos recibidos mientras existan datos en el *buffer*.

El lazo se emplea para leer todos los bytes presentes en *buffer* de recepción. Dentro del lazo se utiliza el método `client.read()` para leer el siguiente byte enviado por el cliente.

Finalmente, se almacena el mensaje recibido en `miMensaje`.

```
String miMensaje = "";
while (client.available())
{
  char c = client.read();
  miMensaje = miMensaje + c;
}
```

Código 2.12. Recepción de datos en Arduino

Comunicación mediante Bluetooth

Para el envío y recepción de información a través de Bluetooth con Arduino, se emplea un puerto serial, por el cual se envían y reciben los bits de datos de forma secuencial. Para ello, se especifica la tasa de transmisión en 57600 bps empleando el método `begin`, como se observa en el Código 2.13.

Para el envío de información se utiliza el método `print`, pasando como parámetro a este método la secuencia de caracteres a enviar.

```
Serial.begin(57600);
Serial.print("611");
```

Código 2.13. Envío de datos por Bluetooth en Arduino

El proceso de recepción de datos en el controlador se detalla en el Código 2.14. El método `available()` retorna el número de bytes por leer en el *buffer* de Arduino. Mientras existan datos disponibles en la memoria se utiliza el método `read()` para leer los bytes y almacenarlos en la variable auxiliar `c`.

Cuando todos los bytes han sido leídos de *buffer*, el mensaje recibido es accesible a través de la variable `miMensaje`.

```
String miMensaje = "";
while (Serial.available())
{
  char c = Serial.read();
  miMensaje = miMensaje + c;
}
```

Código 2.14. Recepción de datos por Bluetooth en Arduino

2.2.3. Implementación del Sistema de Control Domótico

En esta sección se detalla la implementación de los componentes del sistema domótico que permiten realizar el monitoreo de los elementos y controlar los diferentes dispositivos y funciones del sistema.

2.2.3.1. Implementación del Control de Luminarias

El prototipo de sistema domótico permite controlar el encendido y apagado de las luminarias en la vivienda. El proceso de control de luminarias se observa en la Figura 2.60, y se describe a continuación:

- Se activa o desactiva la luminaria utilizando la interfaz de usuario de la aplicación para Android.
- La aplicación envía un mensaje de actualización -a través de Internet o Bluetooth- para indicar la acción realizada por el usuario. El contenido del mensaje de actualización varía dependiendo de la luminaria a controlar, como se muestra en la Tabla 2.26.
- Arduino recibe e interpreta el comando. En respuesta al mensaje recibido, se genera una señal HIGH o LOW en un pin de la placa. Esta señal controla un circuito que realiza el encendido o apagado de la luminaria.

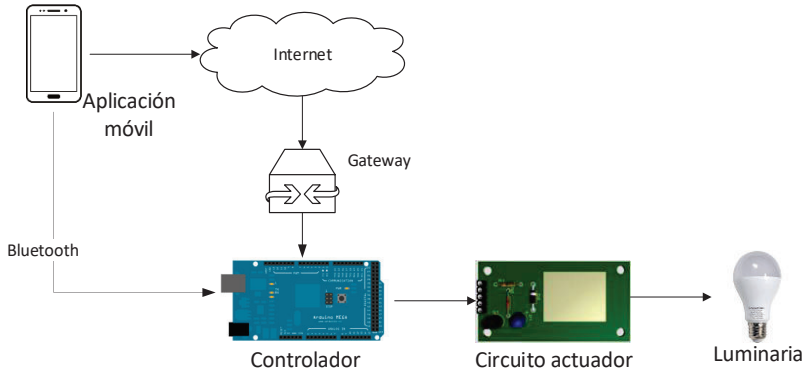

Figura 2.60. Diagrama de control de luminarias

El proceso para encender o apagar la radio es similar al de las luces. Para encender la radio se activa el tomacorriente asociado.

Comandos para el control de luminarias

Se emplea un comando o mensaje de actualización para indicar el estado de las luminarias y la radio en la vivienda. Al tratarse de las luminarias, el mensaje está formado por 3 dígitos: [0][1-6][0-1], como se describe en la Tabla 2.26. El primer carácter indica que el elemento sobre el cual hace referencia el mensaje de actualización es una luminaria, el segundo carácter indica el número de luminaria, y el tercer carácter indica el estado -encendido o apagado- de dicha luminaria. En el caso de la radio, el mensaje de actualización es [2][1][0-1].

Tabla 2.26. Comandos para control de luminarias

Comando			Descripción
Elemento	**Número**	**Estado**	
0	1	0	Luz habitación 1 apagada
		1	Luz habitación 1 encendida
	2	0	Luz habitación 2 apagada
		1	Luz habitación 2 encendida
	3	0	Luz habitación 2 apagada
		1	Luz habitación 2 encendida
	4	0	Luz sala apagada
		1	Luz sala encendida
	5	0	Luz comedor apagada
		1	Luz comedor encendida
	6	0	Luz baño apagada
		1	Luz baño encendida
2	1	0	Radio apagada
		1	Radio encendida

Cuando el valor de estado es 0, Arduino genera una señal de LOW en un pin. Si el estado es 1, Arduino genera una señal de HIGH en dicho pin.

Pines para el control de luminarias

Las luminarias y la radio se encienden o apagan dependiendo del voltaje generado por Arduino. Cuando se genera 5 V en un pin, el dispositivo se enciende. Al colocar 0 V en el

pin, el dispositivo se apaga. En la Tabla 2.27 se detallan los pines de la placa Arduino Mega 2560 utilizados para controlar las luces y la radio.

Tabla 2.27. Pines para encendido/apagado de luces y radio

Número pin	Tipo	Descripción
22	Salida	Controla luminaria de la habitación 1
23	Salida	Controla luminaria de la habitación 2
24	Salida	Controla luminaria de la habitación 3
25	Salida	Controla luminaria de la sala
26	Salida	Controla luminaria del comedor
27	Salida	Controla luminaria del baño
28	Salida	Controla el tomacorriente de la radio

Circuito actuador

Arduino genera señales de 5 V y 0 V, por lo que es necesaria la utilización de un circuito electrónico para activar o desactivar dispositivos que funcionan con 120 V. A continuación, se detalla el diseño del circuito empleado para encender o apagar las luminarias, y activar o desactivar el tomacorriente asociado a la radio.

La Figura 2.61 muestra el esquema del circuito electrónico actuador, que se describe a continuación:

- Se emplea un relé para activar o desactivar las luces y el tomacorriente. Se seleccionó un relé de 5 V, con una resistencia en la bobina de 70 Ω.
- El relé es controlado a través de un transistor, que es utilizado como un interruptor *ON/OFF*. El uso del transistor es necesario debido a que la placa de Arduino puede entregar o recibir hasta 40 mA, lo cual no es suficiente para activar el relé. Se empleará un transistor 2N3904.
- Cuando Arduino genera una señal de 0 V a la base del transistor, este actúa como un interruptor abierto, impidiendo el paso de corriente a través de la bobina, lo que causa que el interruptor del relé permanezca en la misma posición y la luminaria esté desactivada. Si Arduino genera una señal de 5 V, el transistor actúa como un interruptor cerrado, lo que permite el paso de corriente por la bobina del relé. Esto causa un cambio en la posición del interruptor del relé, lo que enciende la luminaria.

- Para visualizar el paso de corriente por la bobina del relé se emplea un led en serie a una resistencia.
- Cuando se desactiva el relé al interrumpir el paso de corriente, se genera un pico de voltaje en la bobina que puede dañar al transistor. Por lo tanto, se emplea un diodo para cortocircuitar la bobina y disipar el pico, protegiendo al transistor. Se utiliza un diodo 1N4007.

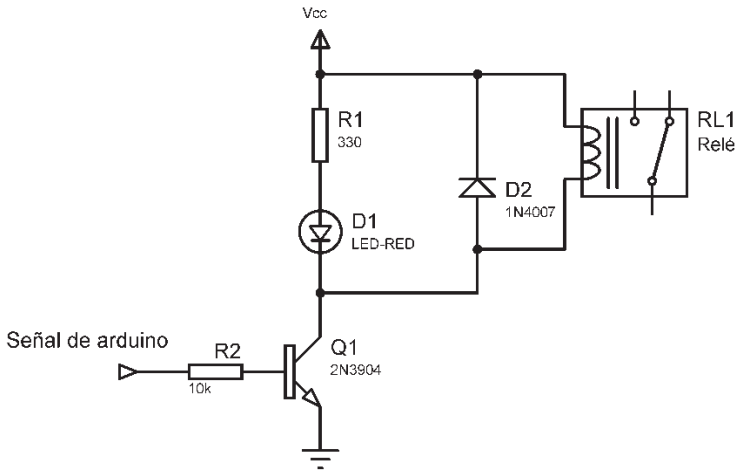

Figura 2.61. Circuito electrónico actuador

A continuación, se presentan las consideraciones a tomar en cuenta en el diseño del circuito actuador:

- La relación entre la corriente de colector I_C y la corriente de base I_B para el transistor es:

$$\beta=150$$

- En el caso de la bobina del relé:

$$V_{bobina} = 5\ V$$

$$R_{bobina} = 70\ \Omega$$

- La placa Arduino cumple con las siguientes especificaciones:

$$I_{pin}max = 40\ mA$$

$$V_{high} = 5\ V$$

- Por lo tanto, la corriente en la bobina se puede calcular de la siguiente manera:

$$I_{bobina} = I_C = \frac{V_{bobina}}{R_{bobina}} = \frac{5\ V}{70\ \Omega} = 71{,}4\ mA$$

- Para conocer si el relé está activado o desactivado se emplea un led y una resistencia R_1 en serie:

$$I_{led} = 10\ mA$$

$$V_{led} = 2\ V$$

$$R_1 = \frac{V_{CC} - V_{led}}{I_{led}} = \frac{5\ V - 2\ V}{10\ mA} = 300\ \Omega$$

$$R_1 = 330\ \Omega$$

- La corriente en la base del colector se obtiene así:

$$\beta = \frac{I_C}{I_B}\ ,\ I_B = \frac{I_C}{\beta} = \frac{71{,}4\ mA}{150} = 0{,}48\ mA$$

- La resistencia R_2 en la base del transistor se puede calcular como se muestra a continuación:

$$V_{BE} = 0{,}6\ V$$

$$R_B = R_2 = \frac{V_{high} - V_{BE}}{I_B} = \frac{5\ V - 0{,}6\ V}{0{,}48\ mA} = 9166{,}67\ \Omega = 9{,}17\ \text{K}\Omega$$

$$R_2 = 10\ K\Omega$$

Circuito de iluminación

La orden de encendido o apagado de luminarias se envía desde la interfaz de usuario. Una vez que Arduino recibe el mensaje, enciende o apaga las luminarias a través de un circuito que consta principalmente de un relé. El circuito eléctrico para el encendido y apagado de las luminarias en la vivienda se muestra en la Figura 2.62 y se describe a continuación:

- Una terminal de cada luminaria se conecta al pin NA de un relé, mientras que el otro terminal de cada luminaria se conecta a la línea de neutro.
- El pin C de cada relé se conecta a la línea de fase.
- Cuando el relé no está energizado, el interruptor permanece en la posición NC, es decir, la luminaria permanece apagada. Cuando circula corriente por la bobina del relé, el interruptor cambia de NC a NA, lo que enciende la luminaria.

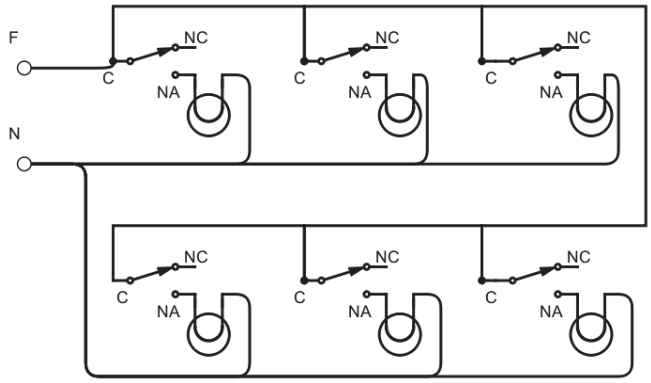

Figura 2.62. Esquema de conexión de luminarias

En Figura 2.63 la se detallan las conexiones a realizar en el circuito actuador empleado para el control de una luminaria.

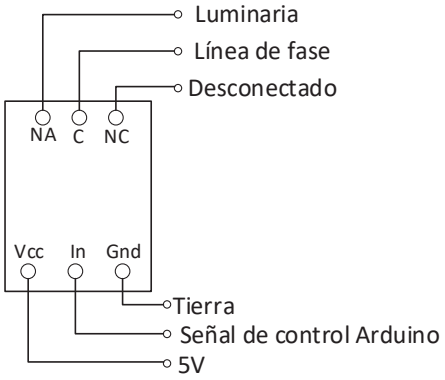

Figura 2.63. Esquema de conexión del circuito actuador

Para el tomacorriente, una de sus terminales se conecta al pin NA del relé, mientras que el otro terminal, se conecta a la línea de neutro, como se indica en la Figura 2.64.

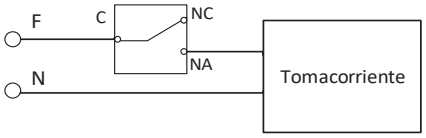

Figura 2.64. Esquema de conexión para tomacorriente

2.2.3.2. Implementación del Sensado de apertura de puertas y ventanas

El prototipo de sistema domótico posibilita monitorear desde la aplicación móvil el estado de los accesos en la vivienda, es decir, permite conocer si una puerta o ventana se encuentra abierta o cerrada. El proceso de sensado de apertura o cierre de una puerta o ventana se muestra en la Figura 2.65, y se detalla continuación:

- El contacto magnético presente en el marco de la puerta o ventana genera, mediante un circuito electrónico, una señal de 0 V o 5 V que es recibida en un pin de Arduino.
- La placa Arduino identifica el estado del acceso asociado a ese pin en base a la señal recibida. Al detectar que una puerta o ventana ha sido abierta o cerrada, se envía un mensaje de actualización a la aplicación móvil para notificar el cambio de estado.
- La aplicación móvil recibe el comando y muestra el cambio de estado de la puerta o ventana en la interfaz de consola de la aplicación móvil.

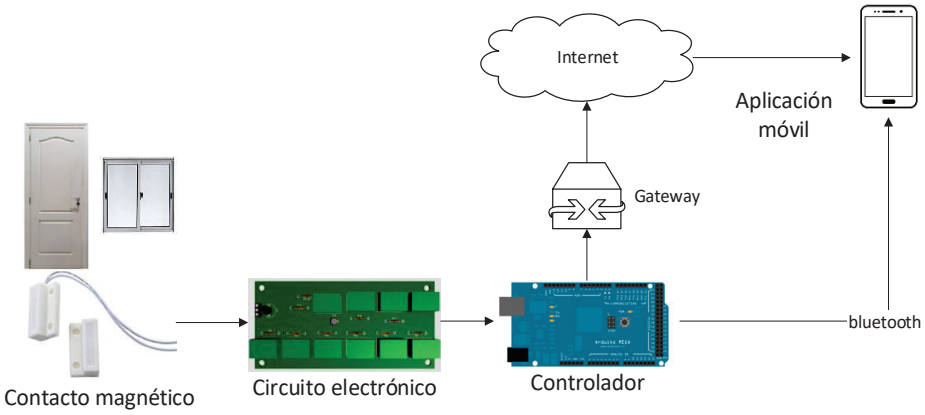

Figura 2.65. Diagrama de sensado de puertas y ventanas

Comandos para sensado de puertas y ventanas

Cuando una puerta o ventana se abre o cierra, la placa Arduino envía un mensaje de actualización a la aplicación móvil. El mensaje consta de 3 caracteres, como se observa en la Tabla 2.28, y permite notificar el estado -abierto o cerrado- de una puerta o ventana.

Al notificar el estado de la puerta, el mensaje se compone de los siguientes caracteres: [6][1][0-1]. El primer carácter indica que el mensaje hace referencia a una puerta, el segundo carácter indica que se trata de la puerta principal de la vivienda, y el tercer carácter muestra si esa puerta se encuentra abierta o cerrada.

El mensaje empleado al notificar el estado de las ventanas es: [7][1-5][0-1]. El primer carácter indica que el mensaje hace referencia a una ventana, el segundo carácter permite diferenciar a una ventana de las otras, y el tercer carácter muestra si esa ventana está abierta o cerrada.

Tabla 2.28. Comandos para sensado de puertas y ventanas

Comando			Descripción
Elemento	**Número**	**Estado**	
7	1	0	Ventana habitación 1 cerrada
		1	Ventana habitación 1 abierta
	2	0	Ventana habitación 2 cerrada
		1	Ventana habitación 2 abierta
	3	0	Ventana habitación 3 cerrada
		1	Ventana habitación 3 abierta
	4	0	Ventana sala cerrada
		1	Ventana sala abierta
	5	0	Ventana comedor cerrada
		1	Ventana comedor abierta
6	1	0	Puerta cerrada
		1	Puerta abierta

Pines para sensado de puertas y ventanas

La placa Arduino determina si un acceso está abierto o cerrado en base al valor entregado por el circuito electrónico del contacto magnético. Si Arduino recibe en un pin un 1L, se interpreta que la puerta o ventana permanece abierta. Al contrario, si la placa Arduino recibe 0L en un pin, la puerta o ventana permanece cerrada. En la Tabla 2.29 se muestran los pines de la placa Arduino Mega 2560 utilizados para monitorear el estado de las puertas y ventanas.

Tabla 2.29. Pines para el sensado de puertas y ventanas

Número pin	Tipo	Descripción
33	Entrada	Contacto magnético de la puerta
34	Entrada	Contacto magnético de la ventana en la habitación 1
35	Entrada	Contacto magnético de la ventana en la habitación 2
36	Entrada	Contacto magnético de la ventana en la habitación 3
37	Entrada	Contacto magnético de la ventana en la sala
38	Entrada	Contacto magnético de la ventana en el comedor

Circuito para sensado de puertas y ventanas

El contacto magnético está formado por una parte fija y una móvil. La parte fija se ubica en el marco de la puerta o ventana, y la parte móvil se ubica sobre la puerta o ventana. Los contactos magnéticos junto a un circuito electrónico son utilizados para conocer si un acceso en la vivienda está abierto o cerrado.

El circuito electrónico para el sensado de apertura de puertas y ventanas, tomado de [79], se basa en una resistencia de *pull-up*, como se muestra en la Figura 2.66 y se describe a continuación:

- Cuando una puerta o ventana está cerrada, ambas partes del contacto magnético se juntan. El interruptor presente en la parte fija del contacto magnético se cierra, lo que permite el paso de corriente. En este caso el pin de entrada de Arduino recibe 0 V.
- Cuando una puerta o ventana se abre, ambas partes del contacto magnético se separan. El interruptor del contacto magnético se abre, lo que impide el paso de corriente. En este caso la placa Arduino recibe una señal de 5 V.
- Se emplea una resistencia de *pull-up* de 10 KΩ para asegurar que el pin permanezca en estado HIGH o 1L mientras el interruptor del contacto magnético está abierto.

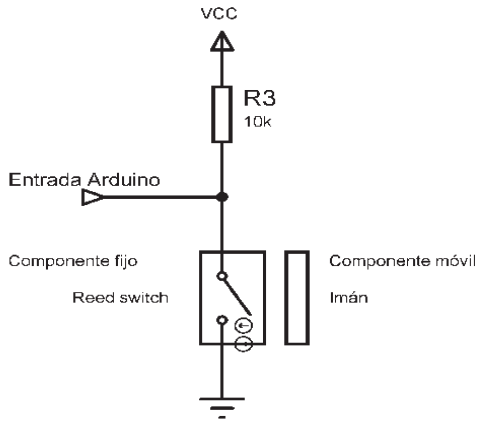

Figura 2.66. Circuito para sensado de puertas y ventanas

2.2.3.3. Implementación de la Detección de Movimiento

El prototipo de sistema domótico posibilita el sensado de presencia en la estancia de la sala de la vivienda. El proceso de sensado de presencia se muestra en la Figura 2.67 y se detalla a continuación:

- Se utiliza un sensor de movimiento PIR que genera señales de 5 V o 0 V, que son recibidas en un pin de Arduino.
- Al detectarse movimiento, el sensor genera una señal de 5 V por un tiempo determinado, que puede ser modificado a través de un potenciómetro presente en la placa del sensor. Transcurrido ese tiempo el sensor genera una señal de 0 V.
- La placa Arduino determina que se ha detectado movimiento al detectar un cambio de 0 V a 5 V en la señal recibida del sensor. Si la función de alarma del sistema domótico se encuentra activada, se envía un mensaje de actualización a la aplicación móvil para notificar lo sucedido.
- La aplicación móvil recibe el mensaje y genera una notificación para alertar al usuario.

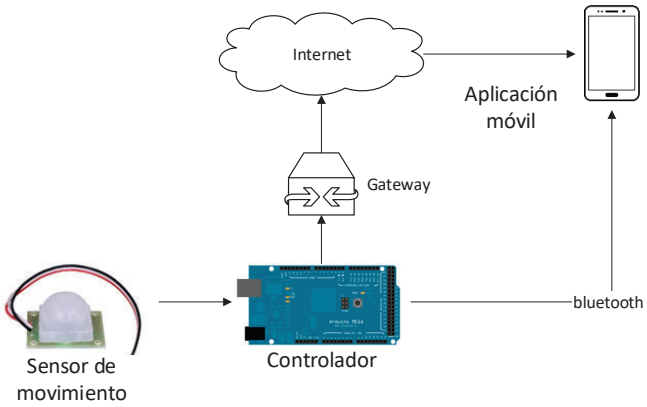

Figura 2.67. Diagrama de detección de movimiento

Comando para la detección de movimiento

Al detectar presencia, Arduino envía un mensaje de actualización a la aplicación móvil. Este mensaje está formado por 3 caracteres, como se muestra en la Tabla 2.30.

El primer y segundo caracteres indican que el mensaje hace referencia a la detección de movimiento, el tercer carácter indica la ubicación del sensor en la vivienda.

Tabla 2.30. Comando para la detección de movimiento

Comando	Descripción
411	Presencia detectada en la sala

Pin para la detección de movimiento

Al detectar movimiento el sensor genera un 1L en el pin de entrada de la placa. El pin de la placa Arduino Mega 2560 en el cual se recibe la señal del sensor se muestra en la Tabla 2.31.

Tabla 2.31. Pin para la detección de movimiento

Número pin	Tipo	Descripción
32	Entrada	Recibe datos del sensor de movimiento

2.2.3.4. Implementación del Control de la Sirena

El sistema domótico posibilita la generación de alertas sonoras a través de una sirena. El proceso de control de la sirena se observa en la Figura 2.68 y se describe a continuación:

- La sirena es activada al presionar el botón de pánico o al abrir una de las puertas o ventanas de la vivienda mientras la alarma está activada.
- Para controlar la sirena la placa Arduino coloca uno de sus pines en estado 1L o 0L. Esta señal es recibida por un circuito electrónico actuador que se encarga de encender o apagar la sirena.
- Cuando la sirena ha sido encendida o apagada, Arduino envía un mensaje de actualización a la aplicación móvil para notificar el estado de la sirena.

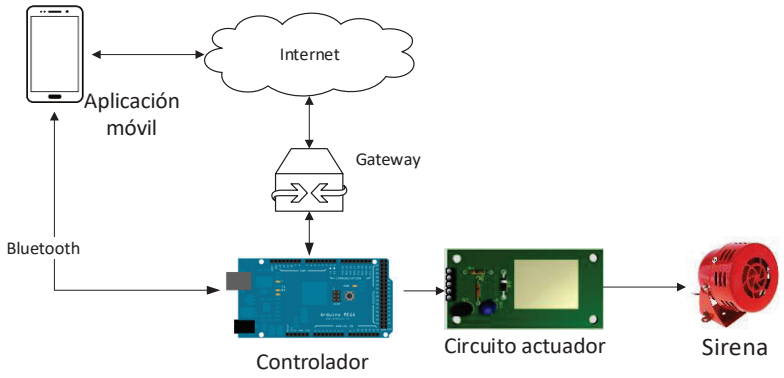

Figura 2.68. Diagrama de control de la sirena

Comandos para control de la sirena

Cuando la sirena es encendida o apagada, se genera un mensaje para notificar su estado actual. El mensaje está formado por 3 caracteres, como se muestra en la Tabla 2.32. El primer carácter indica que el mensaje trata sobre la sirena. El segundo carácter indica el

número de la sirena, en caso de requerir la instalación de otra sirena en la vivienda. El tercer carácter indica si el dispositivo se encuentra encendido o apagado.

Tabla 2.32. Comandos para control de sirena

Comando			Descripción
Elemento	Número	Estado	
8	1	0	Sirena apagada
8	1	1	Sirena encendida

Pin para control de la sirena

Para controlar la sirena Arduino genera una señal de 5 V o 0 V en uno de sus pines. El pin de la placa Arduino Mega 2560 empleado para el control de la sirena se muestra en la Tabla 2.33.

Tabla 2.33. Pin para control de sirena

Número pin	Tipo	Descripción
29	Salida	Utilizado al encender/apagar la sirena

Circuito actuador

Se emplea un circuito electrónico que permite encender o apagar la sirena dependiendo de una señal generada por Arduino. El circuito electrónico actuador se presenta en la sección 2.2.3.1

2.2.3.5. Implementación de la Simulación de Presencia

El sistema domótico posibilita simular presencia en la vivienda mediante la activación de las luminarias de forma automática. A continuación, se detalla el funcionamiento de la función de simulación de presencia:

- Se activa la simulación de presencia desde la interfaz de consola de la aplicación móvil.
- La aplicación móvil verifica el estado de la alarma. Si la alarma está activada, la aplicación móvil envía un mensaje de actualización al controlador Arduino para notificar la activación de la función de simulación de presencia.

- Arduino recibe el mensaje y se encarga de encender o apagar las luminarias en la vivienda sin la intervención del usuario. La activación y desactivación de luminarias se realiza de acuerdo con lo descrito en la sección 2.2.2.1.
- Arduino envía un mensaje para notificar a la aplicación móvil cada vez que una luminaria se enciende o apaga. Por lo tanto, se puede visualizar desde la interfaz de usuario el estado de las luminarias mientras la simulación de presencia está activada.

Comandos para la simulación de presencia

Al activar la función de simulación de presencia, la aplicación móvil notifica al controlador a través de un mensaje de actualización o comando, como se describe en la Tabla 2.34. El mensaje está formado por 3 caracteres, los dos primeros indican que el mensaje hace referencia a la simulación de presencia, el tercer carácter indica el estado de la función de simulación de presencia.

Tabla 2.34. Comandos para la simulación de presencia

Comando	Descripción
510	Simulación de presencia activada
511	Simulación de presencia desactivada

2.2.3.6. Implementación del Botón de Pánico

El sistema domótico cuenta con un botón de pánico que genera una alerta sonora mediante la activación de una sirena utilizada para persuadir al intruso a dejar la vivienda. El usuario activa el botón de pánico mediante un pulsador presente en la vivienda o a través de la aplicación móvil. El funcionamiento del botón de pánico se muestra en la Figura 2.69 y se describe a continuación:

- **Control del botón de pánico empleando pulsador:** Se utiliza un pulsador y un circuito electrónico para generar una señal de 5 V o 0 V, que es recibida en uno de los pines de Arduino.

  Arduino interpreta los valores recibidos y genera una señal en uno de sus pines para encender o apagar la sirena.

  Si el usuario se encuentra conectado al controlador, Arduino envía un mensaje a la aplicación móvil para notificar la activación o desactivación del botón de pánico.

La aplicación móvil interpreta el mensaje recibido y muestra la información en la interfaz de usuario.

- **Control del botón de pánico mediante la aplicación móvil:** El usuario activa o desactiva el botón de pánico desde la aplicación móvil, lo que genera el envío de un mensaje al controlador indicando la acción realizada.

  En función del mensaje recibido, el controlador configura la salida en el pin conectado al circuito actuador, lo que permite encender o apagar la sirena.

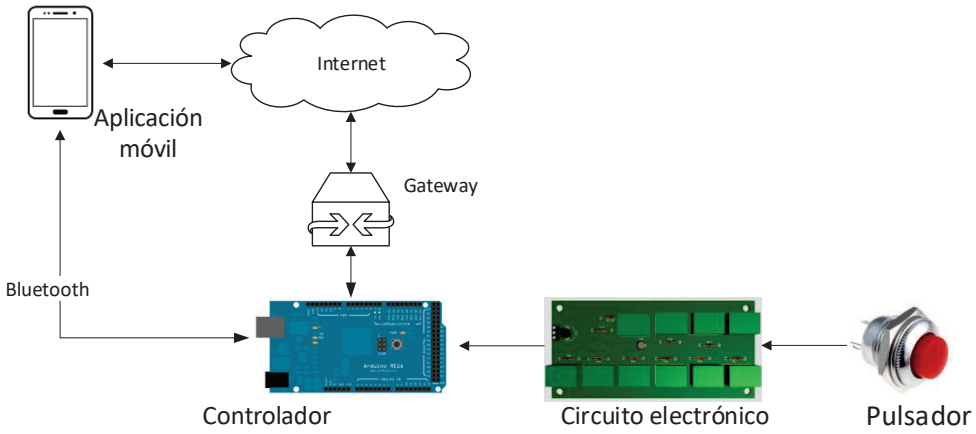

Figura 2.69. Diagrama de gestión del botón de pánico

Comandos para el botón de pánico

Cuando la función de botón de pánico es activada o desactivada, se produce el envío de un mensaje para notificar lo sucedido. El mensaje está formado por 3 caracteres, como se observa en la Tabla 2.35.

El primer carácter indica que el mensaje hace referencia al botón de pánico, el segundo carácter indica un número asociado al pulsador, lo que permite diferenciar a un pulsador de otro en caso de instalar otros pulsadores en la vivienda, el tercer carácter indica el estado del botón de pánico.

Tabla 2.35. Comandos para el botón de pánico

Comando			Descripción
Elemento	Número	Estado	
9	1	0	Botón de pánico desactivado
9	1	1	Botón de pánico activado

Pin para el botón de pánico

El circuito del pulsador envía una señal a un pin de Arduino para indicar que el botón fue presionado. El pin de la placa Arduino Mega 2560 empleado para gestionar el botón de pánico se muestra en la Tabla 2.36.

Tabla 2.36. Pin para el botón de pánico

Número pin	Tipo	Descripción
39	Entrada	Recibe datos del circuito asociado al botón de pánico

Circuito para botón de pánico

Un botón se conecta al microcontrolador y permite generar un determinado nivel lógico cuando se encuentra presionado, cerrado o activo, y el nivel lógico opuesto al permanecer abierto, inactivo o sin presionar.

El circuito electrónico para el botón de pánico, tomado de [80], se muestra en la Figura 2.70 y se describe a continuación:

- Se emplea una resistencia R01 de *pull-up* de 10 KΩ para asegurar que el pin permanezca en estado HIGH o 1L mientras el botón permanezca abierto.
- Al presionar el botón el pin recibe un 0L, lo que activa la función de botón de pánico.
- Al presionar un pulsador común, dos piezas de metal entran en contacto una con la otra. Si estas dos pequeñas piezas de metal no son perfectamente planas o están perfectamente alineadas, pueden hacer contacto y separarse unas cuantas veces antes de pasar a un estado fijo.
- Un microcontrolador es capaz de comprobar el estado de un pulsador varios millones de veces por segundo, por lo tanto, al presionar el pulsador, parecería que fue presionado varias veces en intervalos de tiempo muy pequeños, lo que se conoce como rebote.
- Para asegurarse que la presión del botón sea percibida como una única acción por el microcontrolador se utiliza un circuito anti rebote.
- La R02 de 1 KΩ y C1 de 0,1 uF forman un circuito anti rebote común.

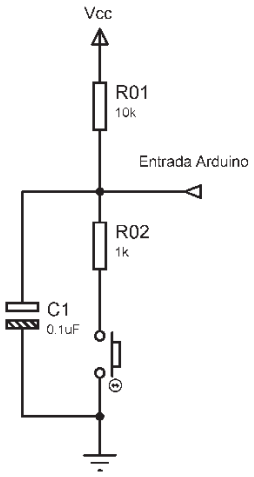

Figura 2.70. Circuito electrónico para el botón de pánico

2.2.3.7. Implementación del Sensado de Temperatura

El prototipo de sistema domótico permite conocer la temperatura en la vivienda. El sensado de temperatura se muestra en la Figura 2.71 y se detalla a continuación:

- El sensor de temperatura LM35 genera una señal eléctrica en la entrada de un pin de Arduino.
- Arduino calcula la temperatura en base a la señal entregada por el sensor. Envía a la aplicación móvil un mensaje de actualización con el valor de la temperatura.
- La aplicación recibe el mensaje y muestra la temperatura en la interfaz de consola de la aplicación móvil.

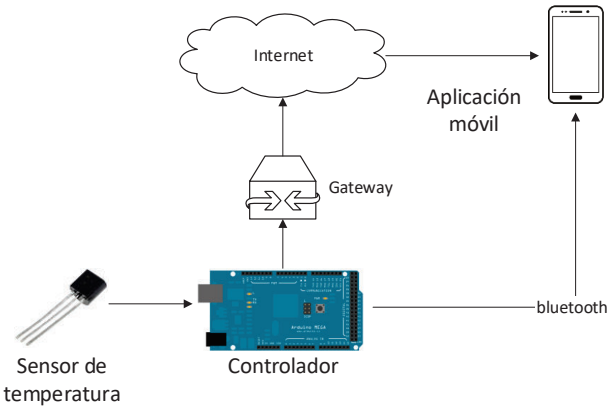

Figura 2.71. Diagrama de sensado de temperatura

Comando para sensado de temperatura

Arduino envía un mensaje a la aplicación móvil indicando la temperatura en la vivienda. El mensaje está formado por 6 caracteres, como se muestra en la Tabla 2.37.

El primer carácter indica que el mensaje contiene la temperatura, el segundo carácter identifica la ubicación del sensor de temperatura en la vivienda, los 4 caracteres restantes indican el valor de temperatura.

Tabla 2.37. Comando para sensado de temperatura

Comando	Descripción
t121.9	La temperatura en la sala de la vivienda es de 21.9 °C

Pin para sensado de temperatura

Arduino recibe la señal del sensor de temperatura en un pin analógico. La Tabla 2.38 muestra el pin de la placa Arduino Mega 2560 empleado para el sensado de temperatura.

Tabla 2.38. Pin para sensado de temperatura

Número pin	Tipo	Descripción
A8	Pin analógico	Recibe datos del sensor de temperatura

2.2.3.8. Implementación del Monitoreo con Cámara de video

La cámara de video es configurada para que al detectar movimiento capture una imagen y la envíe al correo electrónico especificado por el usuario.

La Figura 2.72 muestra el esquema de conexión de la cámara.

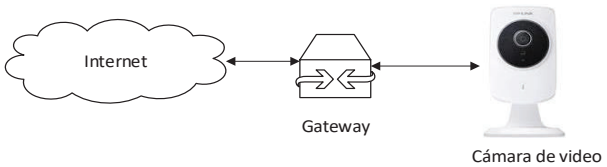

Figura 2.72. Conexión cámara de video

La configuración de la cámara se incluye en el Anexo XI.

2.2.4. Implementación de Tarjetas Electrónicas

En esta sección se describe la implementación de los circuitos electrónicos empleados para controlar cada una de las funciones del prototipo de sistema domótico. Se elaboraron tarjetas electrónicas para realizar las siguientes tareas:

- **Tarjeta actuadora *On/Off*:** Estas tarjetas electrónicas son utilizadas como interruptores para controlar un dispositivo desde la interfaz de usuario de la aplicación móvil. Permiten encender o apagar las luminarias, activar un tomacorriente y encender una sirena.
- **Tarjeta para sensado de puertas y ventanas:** En función del estado del sensor magnético, genera un determinado nivel lógico (1L o 0L) para indicar al controlador Arduino si una puerta o ventana se encuentra abierta.
- **Tarjeta para el control del botón de pánico:** En función de un pulsador, genera un 1L o 0L para indicar al controlador Arduino si el botón de pánico es presionado.
- **Fuente de poder:** Esta tarjeta electrónica permite alimentar al controlador, sensores, actuadores y tarjetas electrónicas empleadas en el prototipo.

2.2.4.1. Fabricación de Tarjetas Electrónicas

A continuación, se detalla el proceso de fabricación de las tarjetas electrónicas:

- Se elaboran los esquemas de los circuitos a implementar, como se indica en la Figura 2.73.

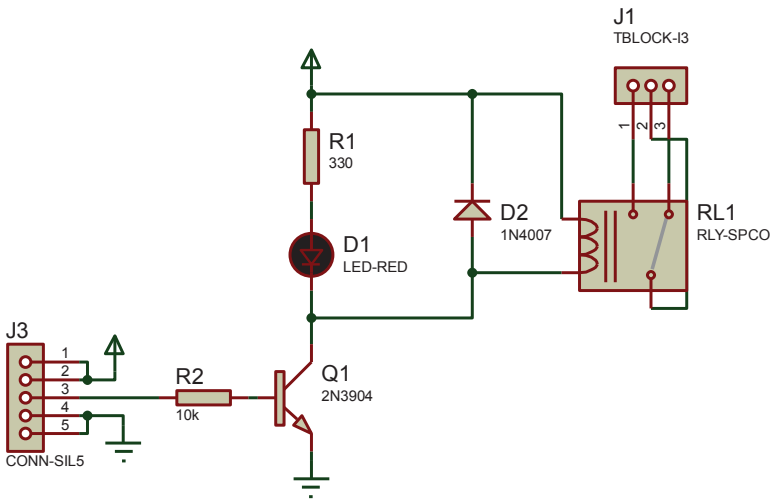

Figura 2.73. Tarjeta actuadora *On/Off*

Los esquemas de los circuitos con todos los componentes electrónicos que estarán presentes en la placa se realizan mediante la aplicación ISIS del programa Proteus.

Los esquemas de todas las tarjetas electrónicas se incluyen en el Anexo XII.

- Se elaboran las pistas con la aplicación Ares del programa Proteus, ver Figura 2.74. Para ello, se sitúan los componentes electrónicos sobre una placa virtual, verificando que la descripción de los elementos empleados en Proteus concuerde con los elementos físicos.

  Las pistas de todas las placas electrónicas se incluyen en el Anexo XIII.

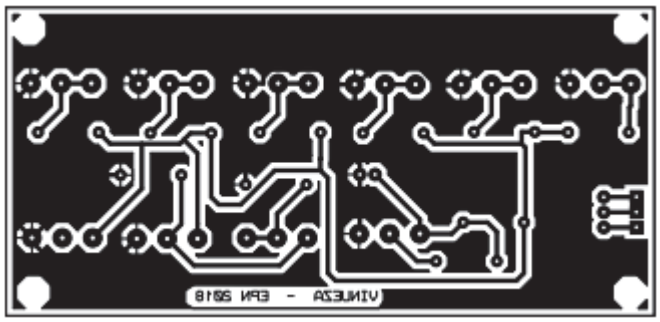

Figura 2.74. Pistas de circuito electrónico

- Se verifica la ubicación de los elementos en la placa a través de una imagen en 3D de la tarjeta electrónica, ver Figura 2.75.

Figura 2.75. Vista en 3D de circuito electrónico

- Se imprime las pistas generadas en papel termotransferible, se recorta la baquelita y se transfiere el tóner a la baquelita mediante planchado, ver Figura 2.76. Se coloca la placa de circuito impreso sobre una sustancia que retira el cobre restante de la baquelita, obteniendo las pistas del circuito (Figura 2.77).

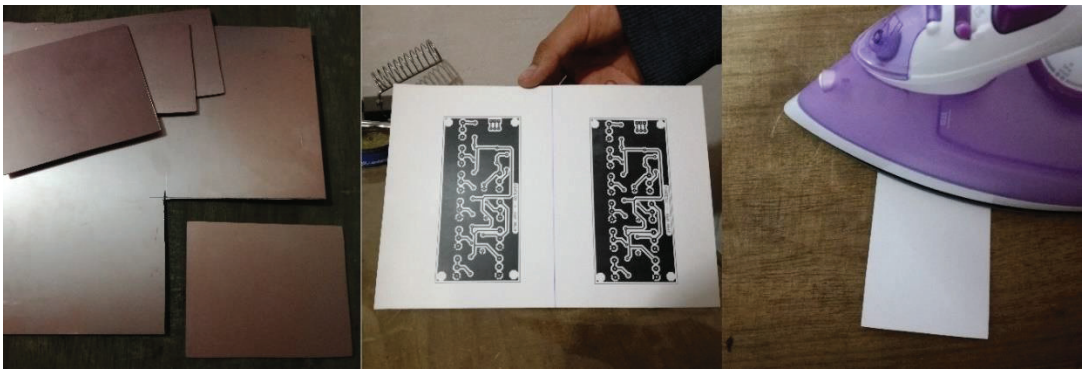

Figura 2.76. Proceso de transferencia de tóner a baquelita

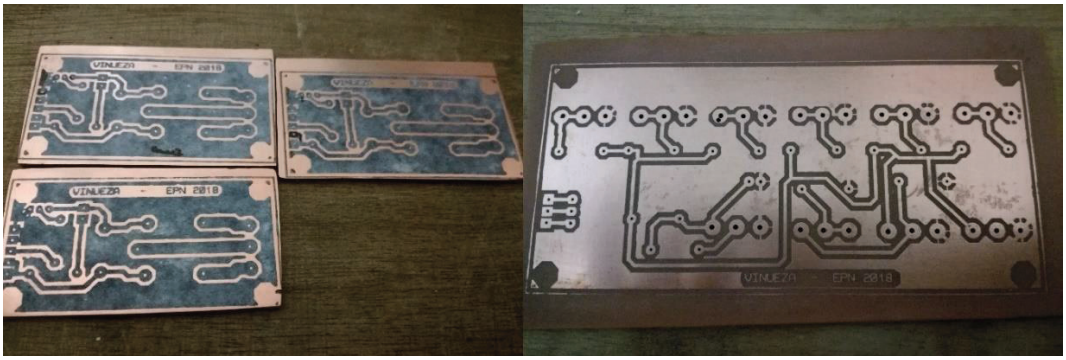

Figura 2.77. Placa de circuito electrónico

- Se realizan las perforaciones en la placa para colocar los componentes del circuito, ver Figura 2.78.

Figura 2.78. Proceso de perforación de placa

- Se procede a soldar los componentes electrónicos en la placa, ver Figura 2.79. Finalmente se obtiene una placa electrónica, como muestra la Figura 2.80. Las hojas técnicas de los principales circuitos integrados utilizados en las placas electrónicas se incluyen en los Anexos XIV, XV.

Figura 2.79. Proceso de soldado de componentes electrónicos

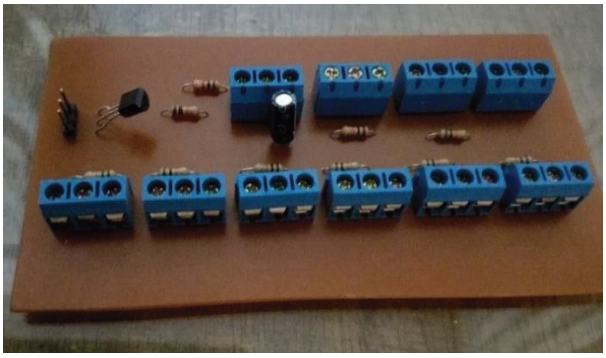

Figura 2.80. Placa electrónica terminada

3. RESULTADOS Y DISCUSIÓN

En este apartado se presentan los resultados de las pruebas realizadas a los componentes del sistema domótico para verificar la funcionalidad del prototipo. El manual de usuario se incluye en el Anexo XVI. El costo referencial del prototipo de sistema domótico se muestra en el Anexo XVII.

3.1. Tablero principal

El tablero principal del prototipo se muestra en la Figura 3.1. Los sensores y los circuitos actuadores distribuidos en la vivienda se conectan a las diferentes placas del tablero principal, como se muestra en la Figura 3.2.

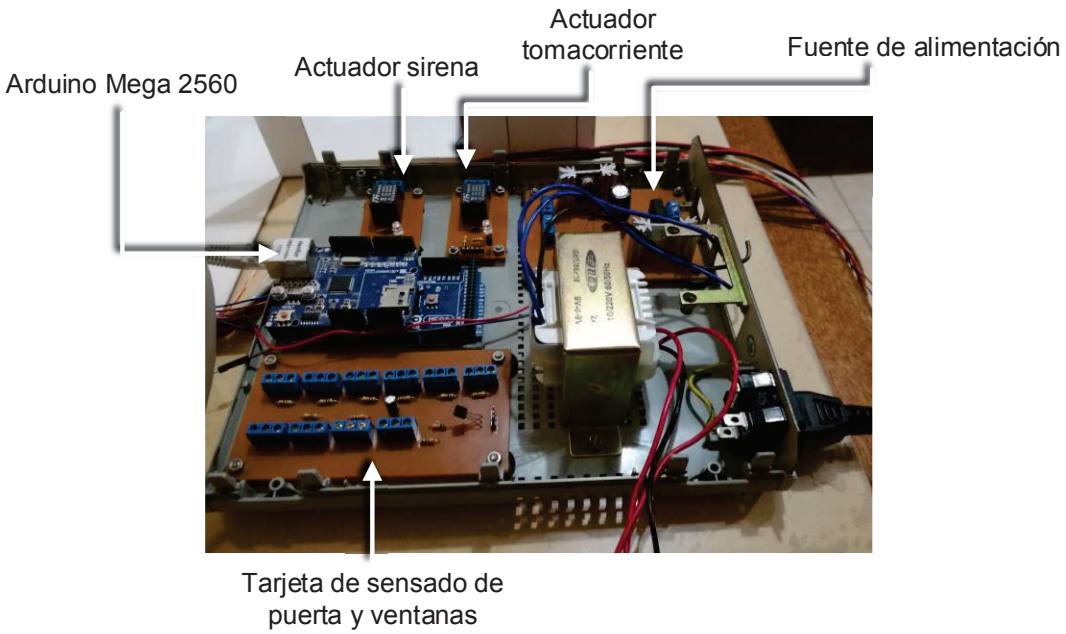

Figura 3.1. Tablero principal del prototipo

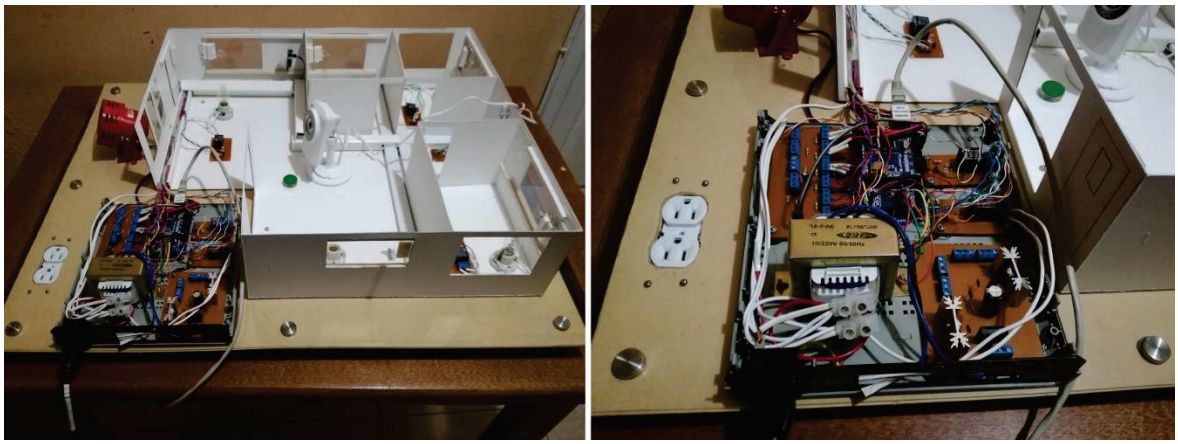

Figura 3.2. Conexiones en el tablero principal

El tablero principal contiene una placa Arduino Mega 2560 conectada a un Ethernet *shield* y a un módulo Bluetooth HC05. Incluye una placa para la conexión de los sensores magnéticos, sensor de temperatura, sensor de presencia, botón de pánico, y las placas actuadoras *on/off*. Además, el tablero contiene la placa de la fuente de alimentación de todo el sistema.

3.2. Comunicación controlador - aplicación móvil

Para comprobar que los mensajes enviados por el controlador Arduino son recibidos por la aplicación Android, se accede a la interfaz de conexión de la aplicación.

Se selecciona Internet como método de conexión, se ingresa el usuario y contraseña, seguidos de la dirección IP 172.16.0.5 y puerto 8000 del controlador Arduino, como se observa en la Figura 3.3.

Al presionar conectar, las credenciales son enviadas a la placa Arduino.

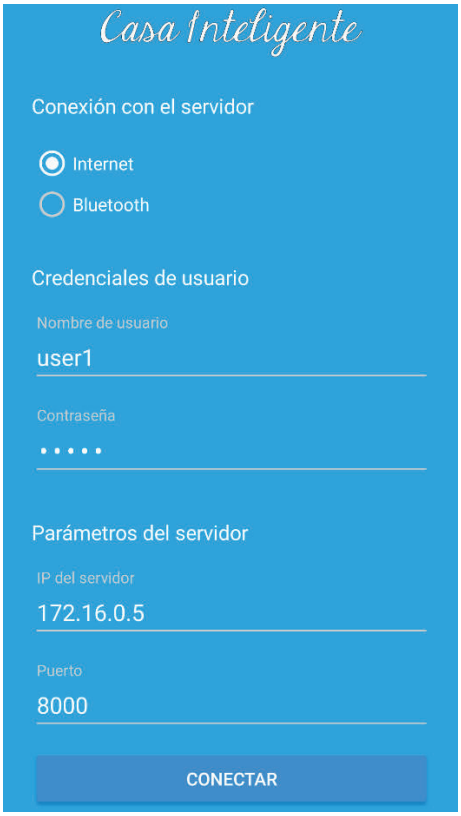

Figura 3.3. Parámetros para conexión con Arduino a través de Internet

Para visualizar las acciones realizadas por el controlador Arduino se emplea el monitor serial del IDE de Arduino. Como se muestra en la Figura 3.4, la placa Arduino recibe el mensaje de la aplicación móvil y genera una respuesta.

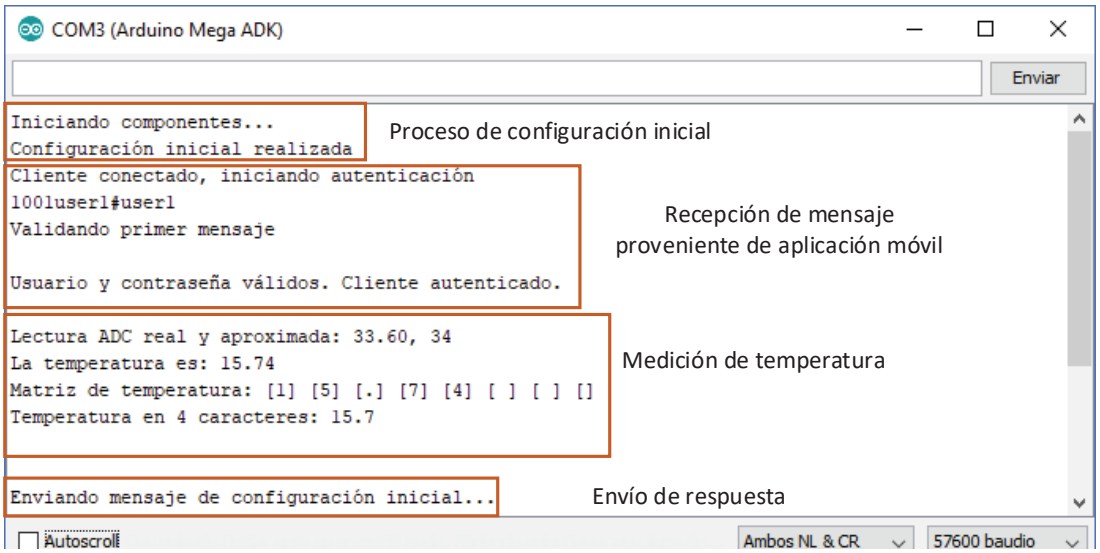

Figura 3.4. Pruebas de envío y recepción de mensajes en la placa Arduino

La aplicación Android genera *logs* para mantener un registro de los mensajes recibidos y errores producidos durante la ejecución. Para visualizar los *logs* generados por la aplicación se emplea el Logcat de Android. Esta es una de las herramientas de monitoreo de Android que posibilita acceder a los *logs* del sistema o la aplicación en tiempo real.

Se conecta el dispositivo móvil al computador y se accede a la ventana Logcat desde Visual Studio, donde se visualiza el mensaje recibido por la aplicación Android, como se indica en la Figura 3.5.

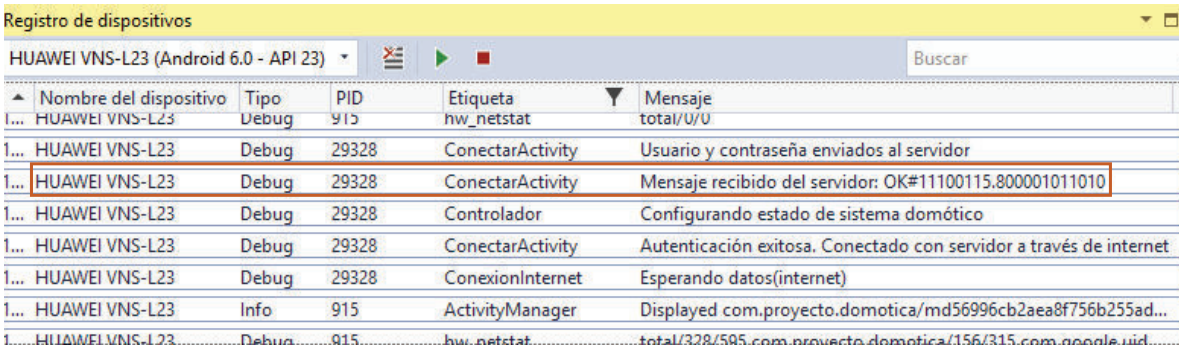

Figura 3.5. *Logs* generados por la aplicación móvil

La ventana permite seleccionar un dispositivo virtual ejecutándose en el emulador o un dispositivo físico para monitorear, en este caso se selecciona un dispositivo físico con Android 6.0 utilizado para las pruebas.

La columna tipo muestra la prioridad de los mensajes de *log*. El PID permite identificar los registros asociados a la aplicación móvil, lo que puede ser utilizado para filtrar todos los registros asociados a la aplicación. La etiqueta se utiliza para identificar la fuente del

mensaje de *log*, es decir, indica la clase o actividad desde donde se genera. La columna mensaje muestra el contenido del *log*.

El mensaje recibido corresponde al mensaje de configuración inicial, que muestra que la autenticación fue exitosa e indica a la aplicación móvil el estado actual de los elementos del sistema domótico. En las pruebas realizadas se obtuvieron resultados similares al conectarse a la placa Arduino mediante Bluetooth.

Una vez comprobado que la aplicación móvil se puede comunicar con el controlador Arduino a través de Internet o Bluetooth, se procede a verificar la función de notificación de errores de conexión.

Para ello se desconecta a la placa Arduino de la energía eléctrica, lo que produce una alerta en la interfaz de consola de la aplicación móvil, en la que se notifica que se ha producido un error en la conexión, como se muestra en la Figura 3.6.

Figura 3.6. Pruebas de notificación de errores de conexión

3.3. Encendido y apagado de luminarias

Para realizar las pruebas de encendido y apagado de luminarias se accede a la interfaz de consola de la aplicación móvil.

Se activan las luminarias empleando los botones de la interfaz, como se muestra en la Figura 3.7.

El encendido o apagado de cada luminaria se lleva a cabo mediante un circuito actuador que se conecta a la placa Arduino en el tablero principal. La conexión de los terminales del actuador se muestra en la Figura 3.8.

Figura 3.7. Encendido de luminarias desde la interfaz de usuario

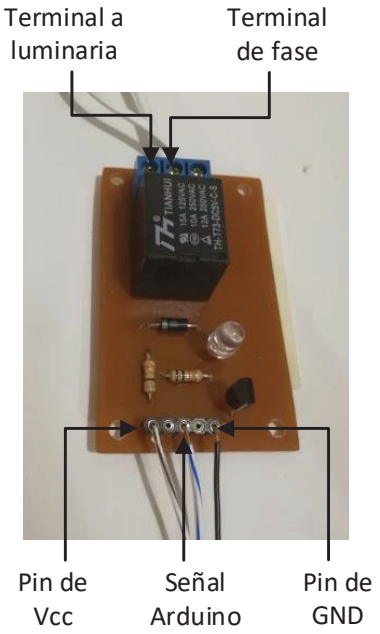

Figura 3.8. Terminales del circuito actuador

El circuito actuador incluye un led que indica el estado de la luminaria. El led se enciende cuando Arduino envía una señal al circuito para activar la luminaria, como se muestra en la Figura 3.9. Esto es utilizado para verificar si una luminaria permanece encendida o apagada.

Se utiliza el mismo circuito actuador para permitir o bloquear el paso de corriente en el tomacorriente que se emplea al encender la radio.

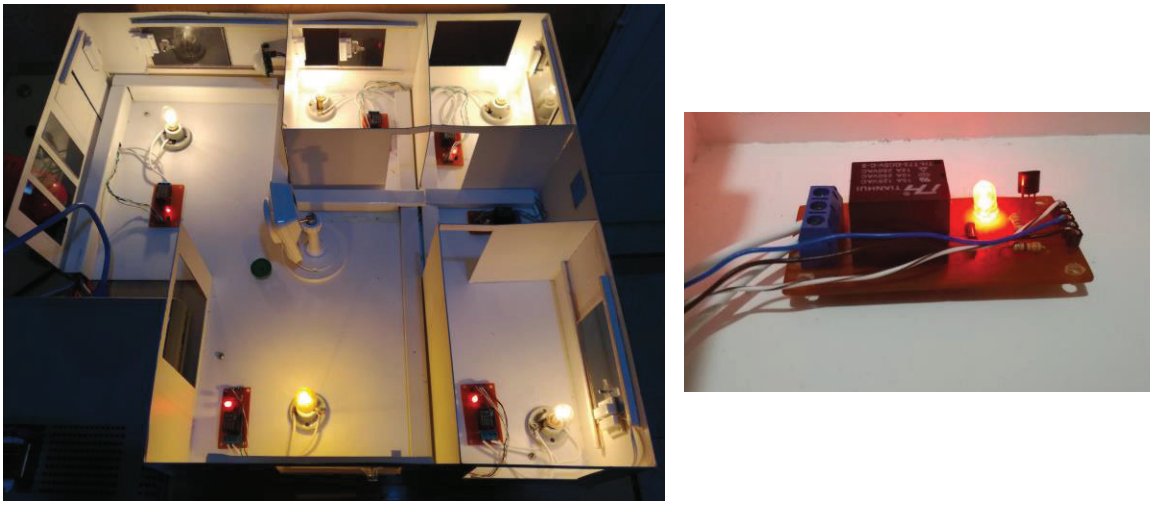

Figura 3.9. Pruebas de encendido y apagado de luminarias

3.4. Sensado de apertura de puerta y ventanas

Para detectar el estado de una puerta o ventana se emplean sensores magnéticos, los cuales se conectan a una tarjeta electrónica presente en el tablero principal, como se muestra en la Figura 3.10.

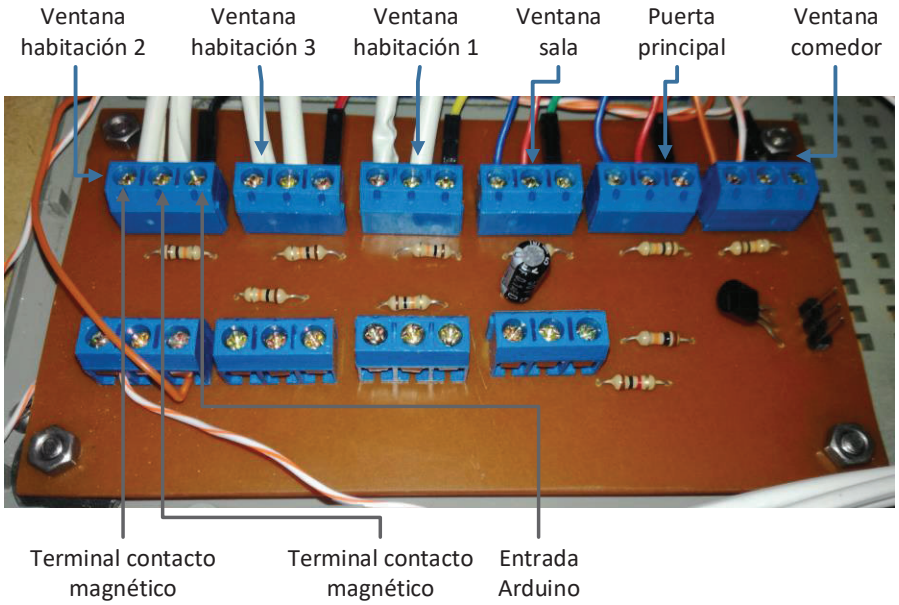

Figura 3.10. Conexión de contactos magnéticos en el tablero principal

Al abrir una ventana o puerta, la aplicación móvil muestra una imagen que indica que dicho acceso permanece abierto. Cuando el acceso se cierra, la aplicación muestra una imagen indicando su estado actual, ver Figura 3.11.

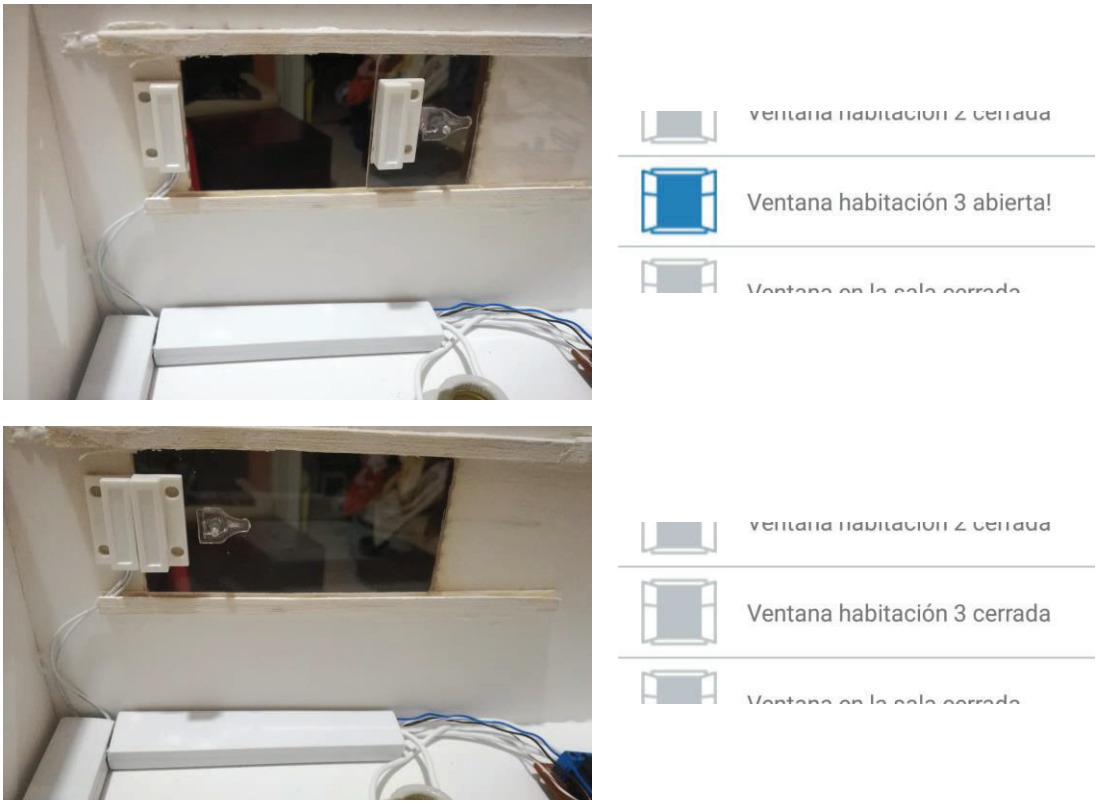

Figura 3.11. Pruebas del sensado de apertura/cierre de puerta y ventanas

En las pruebas realizadas se constató que mientras la alarma permanece activada, la apertura de una puerta o ventana causa una alerta sonora.

3.5. Detección de movimiento

El sensor de movimiento se conecta a la placa Arduino en el tablero principal. En la Figura 3.12 se muestra las conexiones realizadas en el sensor.

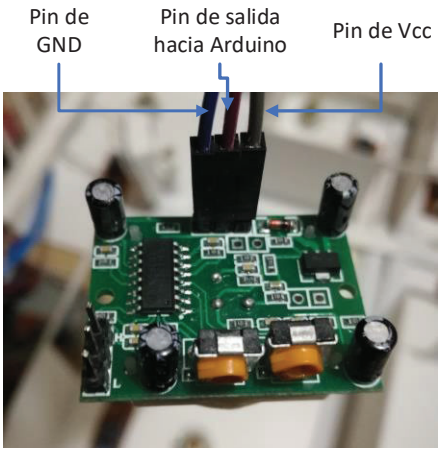

Figura 3.12. Conexión de sensor de movimiento

Para verificar esta funcionalidad del sistema, se activa la alarma en la vivienda, lo que habilita la función de detección de movimiento, como se muestra en la Figura 3.13.

Figura 3.13. Activación de la función de alarma

Se genera movimiento en la estancia de la sala donde está ubicado el sensor, lo que produce una notificación de alerta en la interfaz de usuario, como se observa en la Figura 3.14.

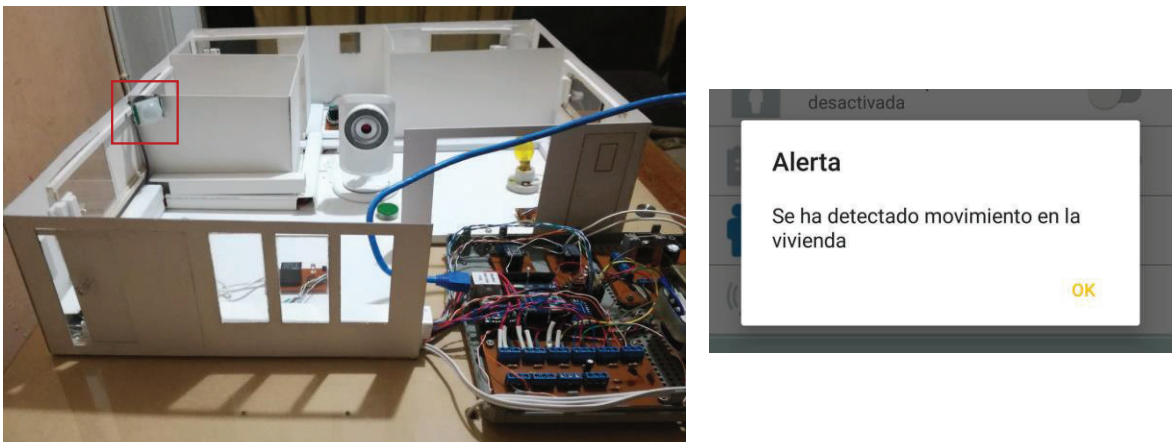

Figura 3.14. Pruebas de detección de movimiento

3.6. Simulación de presencia

Para verificar la funcionalidad del sistema, se accede a la interfaz de consola de la aplicación móvil y se activa la alarma, ya que es un requisito antes de activar la simulación de presencia.

Se activa la simulación de presencia al presionar el botón correspondiente de la interfaz de usuario y se comprueba que las luces de la vivienda se encienden y apagan sin la intervención de usuario, como se muestra en la Figura 3.15.

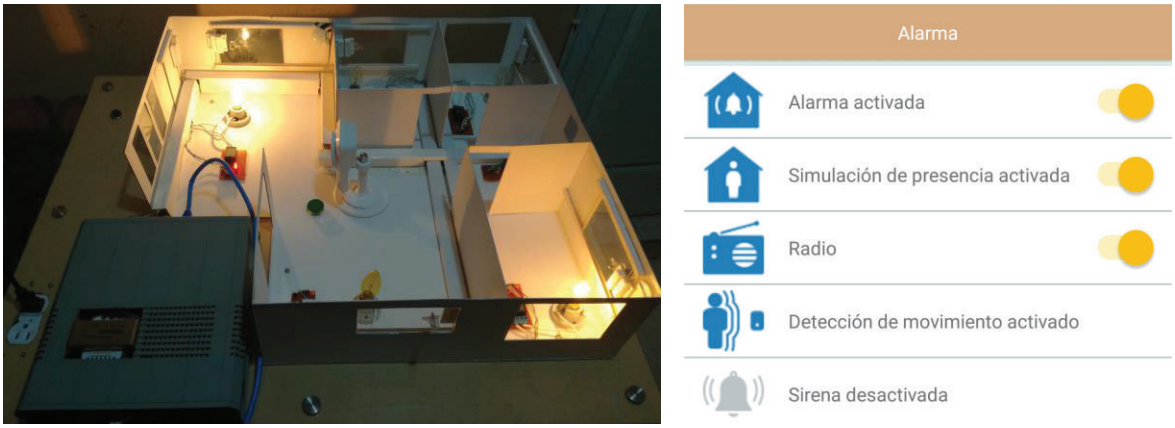

Figura 3.15. Pruebas de simulación de presencia

3.7. Botón de pánico

Para comprobar la funcionalidad del sistema se accede a la interfaz de consola de la aplicación móvil y se activa la función de botón de pánico presionando sobre el botón flotante, lo que genera una alerta sonora mediante la activación de la sirena, como se muestra en la Figura 3.16.

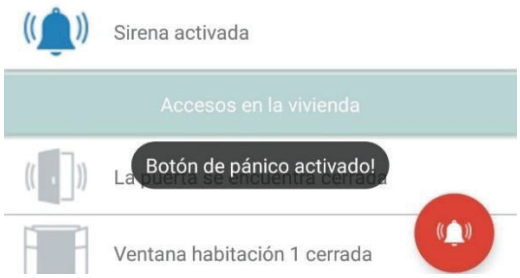

Figura 3.16. Pruebas de botón de pánico desde interfaz de usuario

Adicionalmente, se comprueba que la función de botón de pánico se puede activar al presionar el pulsador, como se muestra en la Figura 3.17.

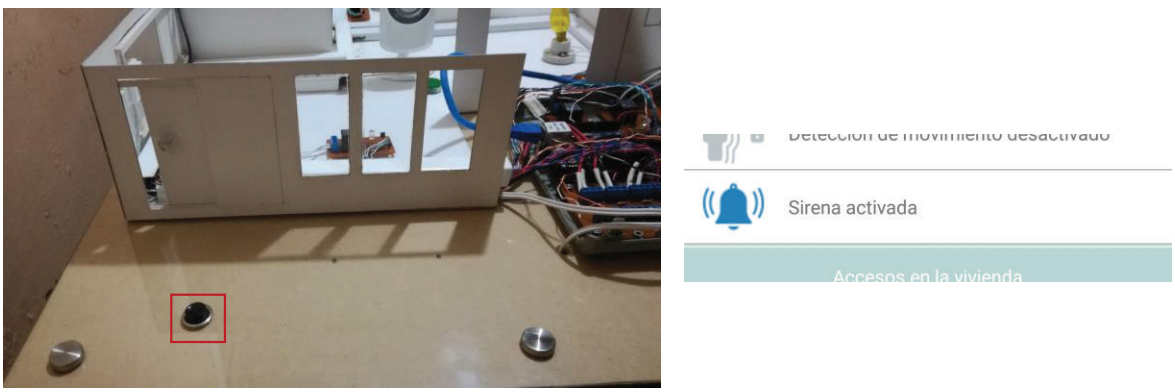

Figura 3.17. Pruebas de botón de pánico con pulsador

El pulsador empleado se conecta a una placa en el tablero principal desde la cual se envía una señal hacia Arduino, como se muestra en la Figura 3.18.

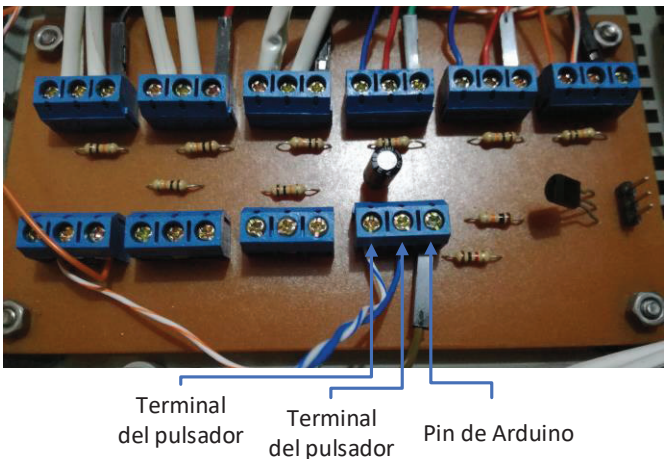

Figura 3.18. Conexión del pulsador en el tablero principal

3.8. Sensado de temperatura

El sensor de temperatura se ubica en una placa del tablero principal, desde la cual se conecta al controlador Arduino. En la Figura 3.19 se muestran las conexiones realizadas en el sensor de temperatura.

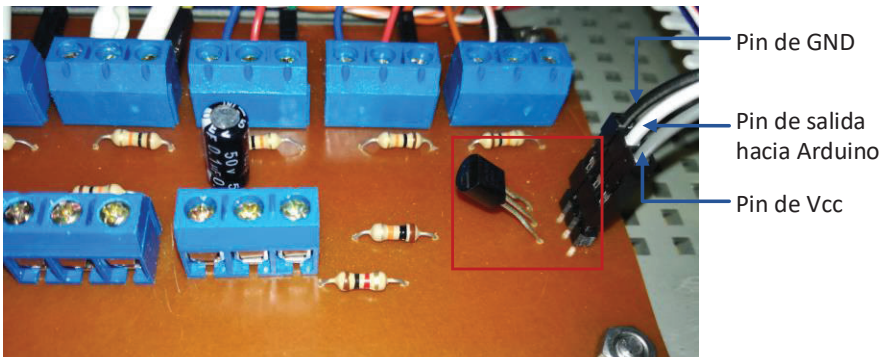

Figura 3.19. Conexión del sensor de temperatura

Para verificar la funcionalidad de sensado de temperatura se accede a la interfaz de consola. Se comprueba que la aplicación muestra el valor de temperatura en °C obtenido por el sensor, como se indica en la Figura 3.20.

Figura 3.20. Pruebas de sensado de temperatura

3.9. Monitoreo con cámara de video

Para comprobar la funcionalidad de monitoreo con la cámara, se mueve un objeto en la estancia de la sala de la vivienda donde se ubica la cámara de video. Al detectarse presencia, se genera una notificación para informar al usuario lo ocurrido a través de la aplicación móvil, como se muestra en la Figura 3.21.

Figura 3.21. Notificación de detección de movimiento

Adicionalmente, al detectarse movimiento, la cámara toma una fotografía y envía la imagen capturada al correo electrónico configurado previamente. Para acceder a la imagen se emplea una aplicación para correo electrónico desde el dispositivo móvil, como se muestra en la Figura 3.22. La imagen capturada por la cámara IP se muestra en la Figura 3.23.

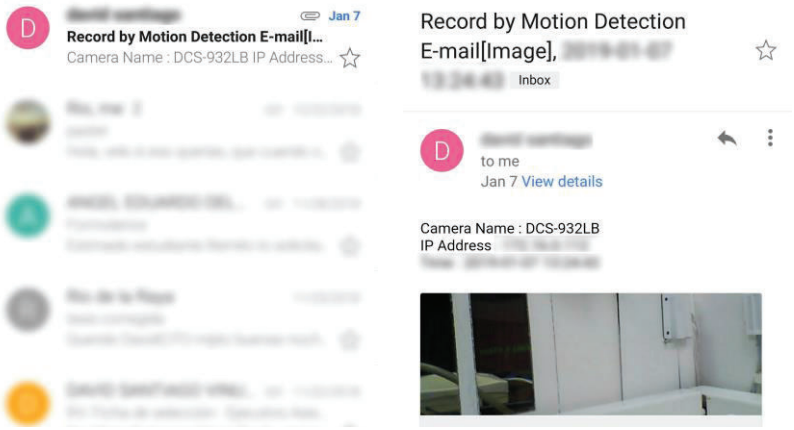

Figura 3.22. Acceso a la imagen capturada desde el correo electrónico

Figura 3.23. Pruebas de monitoreo con la cámara IP

Es posible acceder a la configuración de la cámara de video empleando una aplicación externa al sistema. Para ello, se desplaza a la interfaz de configuración de cámara y se presiona sobre el texto “Acceder a la cámara”, como se indica en la Figura 3.24.

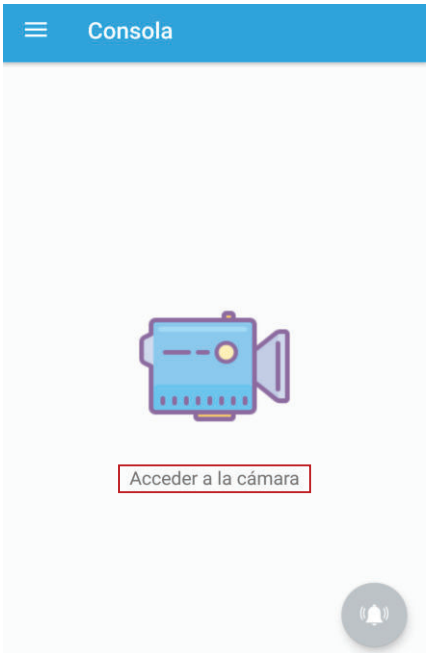

Figura 3.24. Pruebas de acceso a la configuración de la cámara

A continuación, se abre una aplicación externa al sistema que permite configurar ciertos parámetros de la cámara. La Figura 3.25 muestra algunas interfaces de la aplicación móvil para la cámara IP.

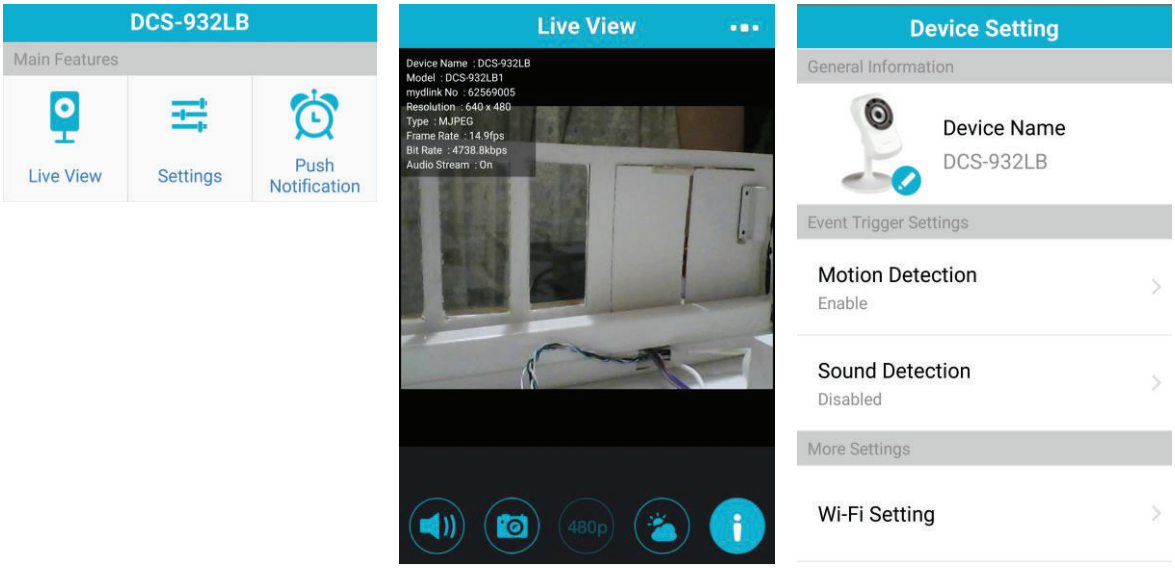

Figura 3.25. Aplicación externa para configuración de la cámara

La ubicación de la cámara se indica en la Figura 3.26.

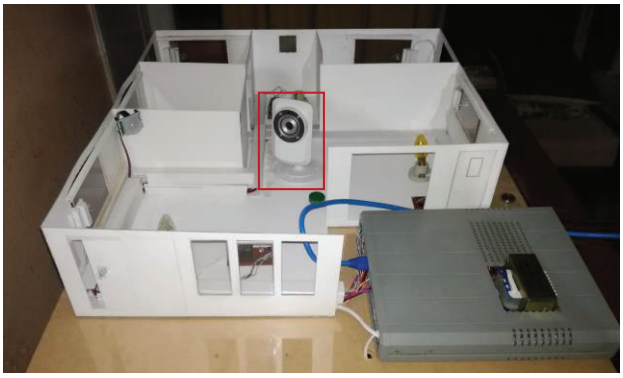

Figura 3.26. Ubicación de la cámara de video

4. CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones y recomendaciones en base a los resultados obtenidos en el desarrollo del prototipo de sistema domótico.

4.1. Conclusiones

- Se desarrolló un prototipo de sistema domótico basado en la plataforma Arduino que genera alertas de seguridad y gestiona equipos e instalaciones, mejorando la calidad de vida de las personas.
- El presente trabajo contribuye a satisfacer la necesidad que existe en el Ecuador de contar con un sistema domótico en las viviendas, que comprometa un uso más eficiente de la tecnología y permita centralizar la gestión de la vivienda.
- La generación de alertas al detectar movimiento, el monitoreo de la apertura de puertas o ventanas y vigilancia mediante una cámara, son funciones del prototipo de sistema domótico que proporcionan mayor seguridad a los habitantes de la vivienda y contribuyen a una mejor calidad de vida.
- La función de simulación de presencia controla las luces de forma automática, lo que genera el envío de mensajes desde el controlador Arduino para notificar el estado de cada luminaria. De esta manera el usuario puede visualizar en todo momento el estado de cada luminaria.
- Se implementó un mecanismo para generar una alerta sonora en el hogar mediante un botón de pánico, que puede ser activado desde la interfaz de usuario de la aplicación móvil o a través de un pulsador presente en la vivienda.
- El controlador Arduino genera notificaciones cada vez que una puerta o ventana se abre o cierra. Esto posibilita al usuario conocer el estado de los accesos de la vivienda en cualquier momento, aun al permanecer fuera del hogar.
- Se desarrolló una aplicación para el sistema operativo Android que posibilita la gestión del prototipo de sistema domótico a través de Bluetooth o Internet, mediante una interfaz gráfica de fácil comprensión.
- Al activar la alarma en la vivienda, las funciones de monitoreo de los accesos y simulación de presencia son habilitadas. Esta funcionalidad del prototipo de sistema domótico se sigue ejecutando incluso cuando no existe un usuario que gestione el sistema desde la aplicación móvil.

- La gestión centralizada de las luminarias y tomacorriente, además del monitoreo de la temperatura a través del prototipo de sistema domótico, brindan un aumento del confort a los habitantes de la vivienda.
- El prototipo posibilita conocer el estado de las luminarias del hogar, ya sea encendido o apagado. Esto permite al usuario apagar las luces al encontrarse al interior o fuera de la vivienda. Esta optimización del consumo de energía eléctrica redunda en un ahorro económico para el usuario.
- La placa Arduino es considerada hardware de fuente abierta lo que posibilita su libre utilización en el desarrollo del prototipo de sistema domótico y permite obtener un prototipo de bajo costo.
- Se desarrolló un prototipo de bajo costo que permite solventar el desafío económico que supone para los usuarios la instalación de un sistema domótico, facilitando el acceso a nuevas tecnologías aplicadas al hogar.
- Se desarrolló un mecanismo para notificar al usuario al producirse fallas en la conexión entre el controlador Arduino y la aplicación móvil, lo que posibilita volver a establecer una conexión entre ambos dispositivos empleando Bluetooth o el Internet.
- La plataforma Arduino posee los recursos necesarios para actuar como la unidad de control del prototipo de sistema domótico, además de facilitar la programación del microcontrolador, lo que la convierte en la plataforma de desarrollo que mejor se adapta a este trabajo.
- La aplicación móvil basada en Android recibe mensajes de actualización del estado de los elementos y envía comandos que son interpretados por el microcontrolador, lo que permite controlar el prototipo de sistema domótico de forma remota y evita la utilización de dispositivos dedicados para su gestión.
- Para el desarrollo de la aplicación Android se emplea el lenguaje C# debido a su facilidad de uso y a la posibilidad de emplear un entorno de desarrollo conocido.
- El prototipo de sistema domótico es gestionado a través de una aplicación para dispositivo móvil que facilita al usuario, incluyendo personas con limitaciones físicas que presentan discapacidad para caminar, el acceso a las funcionalidades del sistema.

4.2. Recomendaciones

- Verificar que los circuitos impresos sean transferidos de forma apropiada del papel termotransferible a la baquelita para asegurar que las pistas de los circuitos sean correctas y que la tarjeta electrónica funcione de manera adecuada.
- Elaborar las tarjetas electrónicas en lugar de adquirir tarjetas ya fabricadas para facilitar la solución de posibles fallas en el hardware y adquirir mayor conocimiento sobre el proceso de elaboración de placas de circuito impreso.
- Incrementar el número de cámaras posibilitando la captura de fotografías en diferentes zonas de la vivienda.
- Contar con un sistema de respaldo de energía para asegurar el funcionamiento del prototipo de sistema domótico al producirse fallas en el suministro eléctrico.
- Instalar contactos magnéticos en todas las puertas y ventanas, a fin de mejorar el monitoreo de los accesos de la vivienda cuando esta permanece deshabitada.
- Mejorar el mecanismo de sensado de presencia incrementando el número de sensores para abarcar cada una de las estancias de la vivienda.
- Configurar el reenvío de puertos en el *gateway* de la vivienda, a fin de permitir el establecimiento de la conexión entre el dispositivo móvil y el controlador ubicado dentro de la red privada del hogar; posibilitando la gestión del prototipo de sistema domótico a través de Internet.
- Mejorar el mecanismo de simulación de presencia, facilitando al usuario la selección de los dispositivos a ser controlados de forma automática en el hogar.
- Mejorar el mecanismo de vigilancia por cámara que posibilita la captura de fotografías en la vivienda, brindando la capacidad de visualizar las imágenes capturadas desde la interfaz de usuario de la aplicación móvil.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] SANDETEL, «Estado del arte de las TIC aplicadas a la edificación inteligente,» Octubre 2011. [En línea]. Available: http://www.juntadeandalucia.es/export/drupaljda/Las_TIC_aplicadas_edificacion_inteligente.pdf. [Último acceso: Mayo 2018].

[2] A. Nogales Escudero, La domótica como solución del futuro, Madrid: Fundación de la Energía de la Comunidad de Madrid, 2007.

[3] Instituto Cerda, «La vivienda domótica: ahorro, confort, seguridad y comunicaciones,» 2000. [En línea]. Available: http://www.ramonmillan.com/documentos/bibliografia/GuiaViviendaDomotica_InstitutCerda.pdf. [Último acceso: Octubre 2017].

[4] S. Merino Cordoba y F. Guzmán Navarro, Domótica: Gestión de la energía y gestión técnica de edificios, Madrid: RA-MA, 2016.

[5] E. Palomeque Vallejo, «Diagnóstico sobre seguridad ciudadana en el Ecuador,» de *Seguridad ciudadana ¿espejismo o realidad?*, F. Carrión, Ed., Quito, FLACSO, 2002, pp. 235-237.

[6] R. J. Millán Tejedor, «La vivienda domótica,» *PC World,* nº 200, pp. 196-198, 2003.

[7] J. M. Huidobro Moya y R. J. Millán Tejedor, Domótica: Edificios Inteligentes, México D.F.: Limusa, 2007.

[8] L. Redolfi, Domótica, Buenos Aires: Fox Andina, 2013.

[9] L. Molina González, Instalaciones domóticas, España: McGraw-Hill, 2010.

[10] M. Casa y A. Rodríguez, Instalaciones Domóticas, España: Marcombo, 2015.

[11] Telefónica, «Libro Blanco del Hogar Digital y las Infraestructuras Comunes de Telecomunicaciones,» España, 2003.

[12] O. Kyas, How to smart home, Alemania: Key Concept Press, 2013.

[13] Trerice, «Electronic temperature sensors: Design & Operation,» [En línea]. Available: http://www.trerice.com/pdfs/2015/Temperature/ElectronicTemperatureSensorsDesignOperation_138_139.pdf. [Último acceso: Octubre 2017].

[14] Automatización Integral de edificios, Universidad de Oviedo, «Sensores para domótica e inmótica,» [En línea]. Available: http://isa.uniovi.es/docencia/AutomEdificios/transparencias/sensores.pdf. [Último acceso: Noviembre 2017].

[15] Gems Sensors & Controls, «What is a reed switch?,» [En línea]. Available: http://docs-europe.electrocomponents.com/webdocs/00b8/0900766b800b81b0.pdf. [Último acceso: Noviembre 2017].

[16] Fargo Controls, «Operating Principles for Magnetic Sensors,» [En línea]. Available: http://www.fargocontrols.com/pdf/sensors/Fargo_mag.pdf. [Último acceso: Noviembre 2017].

[17] M. Schwartz, Arduino Home Automation Projects, Birmingham: Packt Publishing , 2014.

[18] Ministerio de Educación de España, «El relé,» [En línea]. Available: http://platea.pntic.mec.es/~pcastela/tecno/documentos/apuntes/rele.pdf. [Último acceso: Noviembre 2017].

[19] M. Banzi, Getting Started with Arduino, Segunda ed., USA: O'Reilly, 2011.

[20] B. Evans, Beginning Arduino Programming: Writing Code for the Most Popular Microcontroller Board in the World, Apress, 2011, pp. 1-16.

[21] Arduino, «Arduino Uno Rev 3,» [En línea]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Último acceso: Septiembre 2017].

[22] F. Valdés Pérez y R. Pallás Areny, «Introducción a los microcontroladores,» de *Microcontroladores: Fundamentos y aplicaciones con PIC*, Barcelona, Marcombo, 2007, pp. 11-17.

[23] D. Novas Peña, «Microcontroladores: Arquitectura, programación y aplicación,» [En línea]. Available:

https://www.aiu.edu/applications/DocumentLibraryManager/upload/Despradel%20Novas%20Pe%C3%B1a.pdf. [Último acceso: Diciembre 2017].

[24] Arduino, «Bootloader Development,» [En línea]. Available: https://www.arduino.cc/en/Hacking/Bootloader. [Último acceso: Noviembre 2017].

[25] Atmel, «AVR Libc Reference Manual: Toolchain Overview,» [En línea]. Available: https://www.microchip.com/webdoc/AVRLibcReferenceManual/overview_1overview_avr-libc.html. [Último acceso: Diciembre 2017].

[26] Arduino, «Arduino Software IDE,» [En línea]. Available: https://www.arduino.cc/en/Guide/Environment. [Último acceso: Diciembre 2017].

[27] J. Linares Pellicer, «Gráficos por computador: Introducción a processing,» [En línea]. Available: http://users.dsic.upv.es/~jlinares/grafics/processing_spa_1.pdf. [Último acceso: Noviembre 2017].

[28] Arduino, «Compare board specs,» [En línea]. Available: https://www.arduino.cc/en/products.compare. [Último acceso: Enero 2019].

[29] Arduino, «Arduino Mega 2560,» [En línea]. Available: https://www.arduino.cc/en/Main/arduinoBoardMega2560/. [Último acceso: Octubre 2017].

[30] Arduino, «Arduino Mega 2560 Rev3,» [En línea]. Available: https://store.arduino.cc/usa/arduino-mega-2560-rev3. [Último acceso: Octubre 2017].

[31] Adafruit, «Arduino memories,» [En línea]. Available: https://learn.adafruit.com/memories-of-an-arduino/arduino-memories. [Último acceso: Octubre 2017].

[32] Arduino, «Serial Peripheral Interface Library,» [En línea]. Available: https://www.arduino.cc/en/Reference/SPI. [Último acceso: Octubre 2017].

[33] Mouser, «Arduino Ethernet Shield,» [En línea]. Available: http://www.mouser.com/catalog/specsheets/A000056_DATASHEET.pdf. [Último acceso: Noviembre 2017].

[34] Arduino, «Arduino Ethernet Shield V1,» [En línea]. Available: https://www.arduino.cc/en/Main/ArduinoEthernetShieldV1. [Último acceso: Noviembre 2017].

[35] Sparkfun, «Bluetooth Basics,» [En línea]. Available: https://learn.sparkfun.com/tutorials/bluetooth-basics. [Último acceso: Diciembre 2017].

[36] The Linux Information Project, «AT Command Set Definition,» [En línea]. Available: http://www.linfo.org/at_command_set.html. [Último acceso: Diciembre 2017].

[37] Wavesen, «HC-06,» [En línea]. Available: https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf. [Último acceso: Diciembre 2017].

[38] Android, «Arquitectura de la plataforma,» [En línea]. Available: https://developer.android.com/guide/platform/index.html. [Último acceso: Noviembre 2017].

[39] Universidad Carlos III de Madrid, «Arquitectura Android,» [En línea]. Available: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android. [Último acceso: Noviembre 2017].

[40] AndroidPub, «Closer look at Android Runtime,» [En línea]. Available: https://android.jlelse.eu/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924. [Último acceso: Diciembre 2017].

[41] Android, «Aspectos fundamentales de la aplicación,» [En línea]. Available: https://developer.android.com/guide/components/fundamentals.html#Components. [Último acceso: Noviembre 2017].

[42] Android, «Starting an activity,» [En línea]. Available: https://stuff.mit.edu/afs/sipb/project/android/docs/training/basics/activity-lifecycle/starting.html. [Último acceso: Diciembre 2017].

[43] Android, «Activity,» [En línea]. Available: https://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle. [Último acceso: Diciembre 2017].

[44] Android, «Broadcast,» [En línea]. Available: https://developer.android.com/guide/components/broadcasts.html. [Último acceso: Diciembre 2017].

[45] Android, «Servicios,» [En línea]. Available: https://developer.android.com/guide/components/services.html. [Último acceso: Diciembre 2017].

[46] Android, «Content provider basics,» [En línea]. Available: https://developer.android.com/guide/topics/providers/content-provider-basics.html. [Último acceso: Diciembre 2017].

[47] Android, «App Manifest Overview,» [En línea]. Available: https://developer.android.com/guide/topics/manifest/manifest-intro.html. [Último acceso: Diciembre 2017].

[48] Android, «<uses-sdk>,» [En línea]. Available: https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels. [Último acceso: Diciembre 2017].

[49] Android, «Distribution dashboard,» [En línea]. Available: https://developer.android.com/about/dashboards/#Platform. [Último acceso: Octubre 2018].

[50] INEC, «Tecnologías de la Información y Comunicación,» [En línea]. Available: http://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Sociales/TIC/2017/Tics%202017_270718.pdf. [Último acceso: Agosto 2018].

[51] eNeo Laboratories, «Libro Blanco del Hogar Conectado,» 2003. [En línea]. Available: https://www.ramonmillan.com/documentos/bibliografia/LibroBlancoHogarConectado_eNeo.pdf. [Último acceso: Junio 2018].

[52] Agencia de Regulación y Control de Electricidad, «Consumo Anual Per Cápita,» [En línea]. Available: http://www.regulacionelectrica.gob.ec/estadistica-del-sector-electrico/recaudacion-anual/. [Último acceso: Agosto 2018].

[53] Agencia de Regulación y Control de Electricidad, «Estadística Anual y Multianual del Sector Eléctrico Ecuatoriano,» 2016. [En línea]. Available:

http://www.regulacionelectrica.gob.ec/wp-content/uploads/downloads/2017/08/Estad%C3%ADstica-anual-y-multianual-sector-el%C3%A9ctrico-2016.pdf. [Último acceso: Agosto 2018].

[54] Gartner, Inc, «Gartner Says Worldwide Sales of Smartphones Returned to Growth in First Quarter of 2018,» 29 Mayo 2018. [En línea]. Available: https://web.archive.org/web/20180829072934/https://www.gartner.com/newsroom/id/3876865. [Último acceso: Enero 2019].

[55] Requirements Quest, «Nonfunctional requirement examples,» [En línea]. Available: https://requirementsquest.com/wp-content/uploads/2017/01/Nonfunctional-Requirement-EXAMPLES.pdf. [Último acceso: Julio 2018].

[56] S. L. Henry, S. Abou-Zahra y J. Brewer, «The Role of Accessibility in a Universal Web,» 2014.

[57] Sparkfun, «Switch Basics,» [En línea]. Available: https://learn.sparkfun.com/tutorials/switch-basics/poles-and-throws-open-and-closed. [Último acceso: Julio 2018].

[58] Components101, «5V 5Pin Relay,» [En línea]. Available: https://components101.com/5v-relay-pinout-working-datasheet. [Último acceso: Julio 2018].

[59] Texas Instruments, «LM35 Precision Centigrade Temperature Sensors,» [En línea]. Available: http://www.ti.com/lit/ds/symlink/lm35.pdf. [Último acceso: Agosto 2018].

[60] D-Link, «Wireless N IR Home Network Camera,» [En línea]. Available: ftp://ftp.dlinkla.com/pub/DCS-932L/DCS-932_932L_A1_Datasheet_01(W).pdf. [Último acceso: Agosto 2018].

[61] UNAM, «Capacitores,» [En línea]. Available: http://depa.fquim.unam.mx/amyd/archivero/Antologia-Edo-Sol_32006.pdf. [Último acceso: Agosto 2018].

[62] NPTEL, «Light emitting diodes,» [En línea]. Available: https://nptel.ac.in/courses/113106062/Lec16.pdf. [Último acceso: Agosto 2018].

[63] WikiLibros, «Electrónica de comunicaciones/Resistores,» [En línea]. Available: https://es.wikibooks.org/wiki/Electr%C3%B3nica_de_comunicaciones/Resistores. [Último acceso: Agosto 2018].

[64] B. Evans, Arduino Programming Notebook, California, 2007.

[65] Oracle, «Java Platform, Standard Edition 8,» [En línea]. Available: http://www.oracle.com/technetwork/java/javase/jdk-8-readme-2095712.html. [Último acceso: Julio 2018].

[66] Android, «Get the Android SDK,» [En línea]. Available: https://stuff.mit.edu/afs/sipb/project/android/docs/sdk/index.html. [Último acceso: Julio 2018].

[67] Microsoft, «Visual Studio and Xamarin,» [En línea]. Available: https://msdn.microsoft.com/en-us/library/mt299001.aspx. [Último acceso: Julio 2018].

[68] N. Shetty, «Socket Programming,» 2007. [En línea]. Available: http://inst.eecs.berkeley.edu/~ee122/sp07/Socket%20Programming.pdf. [Último acceso: Junio 2018].

[69] P. Fatourou, «Introduction to Sockets Programming in C using TCP/IP,» 2012. [En línea]. Available: https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf. [Último acceso: Junio 2018].

[70] National Instruments, «When to Use UDP Instead of TCP,» 2017. [En línea]. [Último acceso: Junio 2018].

[71] Universidad de Rostock, «Bluetooth Baseband,» [En línea]. Available: https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/baseband.html. [Último acceso: Julio 2018].

[72] CCM Benchmark, «Bluetooth-How it works,» [En línea]. Available: https://ccm.net/contents/69-bluetooth-how-it-works. [Último acceso: Julio 2018].

[73] University of Southern California, «Transmission Control Protocol,» 1981. [En línea]. Available: https://tools.ietf.org/html/rfc793#page-33.

[74] The Linux Documentation Project, «TCP keepalive overview,» [En línea]. Available: http://tldp.org/HOWTO/TCP-Keepalive-HOWTO/overview.html. [Último acceso: Enero 2019].

[75] R. Braden, «Requirements for Internet Hosts -- Communication Layers,» 1989. [En línea]. Available: https://tools.ietf.org/html/rfc1122#page-101. [Último acceso: Enero 2019].

[76] Veritas , «TCP Keepalive Best Practices - detecting network drops and preventing idle socket timeout,» 2013. [En línea]. Available: https://www.veritas.com/support/en_US/article.TECH202675. [Último acceso: Enero 2019].

[77] Lantronix, «TCP Keepalives explained,» 2009. [En línea]. Available: http://ltxfaq.custhelp.com/app/answers/detail/a_id/1512/~/tcp-keepalives-explained. [Último acceso: Enero 2019].

[78] El Telégrafo, «La velocidad del internet en Ecuador supera el promedio,» 2017. [En línea]. Available: https://www.eltelegrafo.com.ec/noticias/economia/4/la-velocidad-del-internet-en-ecuador-supera-el-promedio. [Último acceso: Enero 2019].

[79] Electronics Tutorials, «Pull-up Resistors,» [En línea]. Available: https://www.electronics-tutorials.ws/logic/pull-up-resistor.html. [Último acceso: Enero 2019].

[80] E. Williams, «Embed with Elliot: Debounce your Noisy Buttons,» Diciembre 2015. [En línea]. Available: https://hackaday.com/2015/12/09/embed-with-elliot-debounce-your-noisy-buttons-part-i/. [Último acceso: Enero 2019].

[81] Electronics Hub, «Understanding 7805 IC Voltage Regulator,» 2018. [En línea]. Available: https://www.electronicshub.org/understanding-7805-ic-voltage-regulator/. [Último acceso: Enero 2019].

6. ANEXOS

En esta sección se incluyen los anexos como información complementaria al presente proyecto. Esta sección se distribuye de la siguiente forma:

ANEXO I. Características Arduino Mega 2560

ANEXO II. Características Ethernet *shield*

ANEXO III. Características módulo Bluetooth HC05

ANEXO IV. Características relé polo simple doble salida

ANEXO V. Características sensor PIR HC-SR501

ANEXO VI. Características sensor LM35

ANEXO VII. Características cámara D-Link 932L

ANEXO VIII. Código de la aplicación para Android

ANEXO IX. Estructura de archivos

ANEXO X. Código de la aplicación para la placa Arduino

ANEXO XI. Configuración de cámara de video

ANEXO XII. Circuitos electrónicos implementados

ANEXO XIII. Pistas de las tarjetas electrónicas

ANEXO XIV. Características circuito integrado LM78S05

ANEXO XV. Características circuito integrado KBL400

ANEXO XVI. Manual de usuario

ANEXO XVII. Costo referencial del prototipo

ANEXO I

CARACTERÍSTICAS ARDUINO MEGA 2560

El archivo se encuentra en el CD adjunto.

ANEXO II

CARACTERÍSTICAS ETHERNET SHIELD

El archivo se encuentra en el CD adjunto.

ANEXO III

CARACTERÍSTICAS MÓDULO BLUETOOTH HC05

El archivo se encuentra en el CD adjunto.

ANEXO IV

CARACTERÍSTICAS RELÉ POLO SIMPLE DOBLE SALIDA

El archivo se encuentra en el CD adjunto.

ANEXO V

CARACTERÍSTICAS SENSOR PIR HC-SR501

El archivo se encuentra en el CD adjunto.

ANEXO VI

CARACTERÍSTICAS SENSOR LM35

El archivo se encuentra en el CD adjunto.

ANEXO VII

CARACTERÍSTICAS CÁMARA D-LINK 932L

El archivo se encuentra en el CD adjunto.

ANEXO VIII

CÓDIGO DE LA APLICACIÓN PARA ANDROID

Los archivos se encuentran en el CD adjunto.

ANEXO IX

ESTRUCTURA DE ARCHIVOS

A continuación, se describe la estructura de archivos de la aplicación móvil para Android. El código fuente se adjunta en el Anexo VIII.

- **Carpeta *packages*:** Contiene las librerías de soporte empleadas en la aplicación.
- **Carpeta *test*:** Contiene el código fuente de la aplicación.

A continuación, se muestra el contenido de la carpeta *test*.

- **Carpeta *Fragments*:** Esta carpeta almacena los fragmentos utilizados en la actividad ConsolaActivity. Un fragmento representa parte de la interfaz de usuario. Se pueden combinar varios fragmentos en una pantalla para crear una UI con múltiples vistas y es posible reutilizar un fragmento en múltiples actividades. Esta carpeta contiene los archivos Fragment1.cs y Fragment2.cs.
    - **Fragment1.cs:** Muestra la Interfaz de consola de la aplicación Android, que se observa en la sección 2.2.1.3.
    - **Fragment2.cs:** Muestra la Interfaz de configuración de cámara de la aplicación Android, que se observa en la sección 2.2.1.4
- **Carpeta *Properties*:** Contiene los archivos AndroidManifest.xml y AssemblyInfo.cs.
    - **AndroidManifest.xml*:*** Brinda información sobre la aplicación al sistema Android para que la aplicación pueda ejecutarse. Se indica que la aplicación requiere permisos para usar Internet y Bluetooth. Además, se especifican los parámetos minSdkVersion en 19 y targetSdkVersion en 25.
    - **AssemblyInfo.cs*:*** Este archivo puede ser utilizado para añadir permisos a la aplicación en lugar de utilizar el archivo AndroidManifest.xml.
- **Carpeta *Resources*:** Los recursos hacen referencia a los archivos que sin ser código fuente, componen la aplicación. El código fuente y los recursos son compilados y empaquetados en forma de un APK para su distribución e instalación. Esta carpeta contiene los siguientes ficheros:
    - **Carpeta anim:** Contiene archivos XML que definen animaciones. Está formada por los siguientes ficheros:

- ***push_left_in.xml***: Permite animar la transición al cambiar a otra interfaz.
- ***push_left_out.xml***: Permite animar la transición al regresar a la interfaz anterior.

- **Carpeta *drawable*:** Contiene los recursos gráficos utilizados por la aplicación. Está formado por los siguientes archivos:
  - **alarma.png**: Imagen utilizada en la interfaz de consola para representar la función de alarma del sistema domótico.
  - **alarma_sirena**: Imagen utilizada en la interfaz de consola para representar la sirena.
  - **camara1.png**: Imagen utilizada en la interfaz de configuración de cámara para representar una cámara de video.
  - **casa1.png**: Imagen empleada en la interfaz de consola para representar la simulación de presencia.
  - **casa2.png*:*** Imagen utilizada en la cabecera del panel lateral de navegación para representar una vivienda.
  - **foco.png*:*** Imagen utilizada en la interfaz de consola para representar una luminaria.
  - **movimiento*.png:*** Imagen utilizada en la interfaz de consola al representar la función de detección de movimiento.
  - **puerta.png*:*** Imagen usada en la interfaz de consola al representar una puerta.
  - **radio.png*:*** Imagen usada en la interfaz de consola al representar una radio.
  - **splash_logo.png*:*** Imagen usada en la pantalla de inicio y la interfaz de conexión para representar el logo de la aplicación móvil.
  - **termometro.png**: Imagen utilizada en la interfaz de consola para indicar la temperatura en la vivienda.
  - **ventana.png**: Imagen usada en la interfaz de consola al representar una ventana.

- La densidad de pantalla suele representarse típicamente en puntos por pulgada (dpi) y representa el número de píxeles en un área dada dentro de la pantalla. Para que la aplicación muestre imágenes con buena calidad en dispositivos con diferentes densidades de píxeles, se provee múltiples versiones de cada imagen, correspondiente al ícono de la aplicación y a íconos para el panel lateral de navegación.
  - **Carpeta *drawable-mdpi*:** Contiene íconos para pantallas de densidad media (~160dpi).
  - **Carpeta *drawable-hdpi:*** Contiene íconos para pantallas de alta densidad (~240dpi).
  - **Carpeta *drawable-xhdpi*:** Contiene íconos para pantallas de densidad extra alta (~320dpi).
  - **Carpeta *drawable-xxhdpi*:** Contiene íconos para pantallas de densidad extra-extra alta (~480dpi).
  - **Carpeta *drawable-xxxhdpi:*** Contiene íconos para pantallas de densidad extra-extra-extra alta (~640dpi).
- **Carpeta *layout:*** Contiene archivos XML que describen el diseño de la interfaz de usuario.
  - **conectarActivity.axml**: Define el diseño de la interfaz de conexión.
  - **consolaActivity.axml**: Define el diseño de interfaz de la actividad ConsolaActivity.
  - **fragment1.axml**: Define el diseño de la interfaz de consola.
  - **fragment2.axml*:*** Define el diseño de la interfaz de configuración de cámara.
  - **nav_header.axml**: Define el diseño de la cabecera del panel lateral de navegación.
  - **splashScreenActivity.axml*:*** Define el diseño de la pantalla de inicio.
  - **toolbar.axml*:*** Define el diseño de la *app bar* o barra de la aplicación.

- **Carpeta *menu*:** Contiene archivos XML empleados para describir menús de la aplicación.
  - **nav_menu.xml*:*** Define el diseño del menú del panel lateral de navegación, empleado para desplazarse entre las diferentes interfaces.
- **Carpeta *values*:** Contiene archivos XML que almacenan valores simples, como listas de *strings*, colores, entre otros. Estos recursos son utilizados cuando el dispositivo que está ejecutando la aplicación posee una versión previa a Android 5.0 (API nivel 21).
  - **colors.xml**: Define una lista de colores empleando los valores RGB (*Red, Green, and Blue*) de un color.
  - **dimens.xml**: Establece el margen de los componentes denominados *card views*, empleados en la interfaz de consola.
  - **strings.xml*:*** Almacena todas las constantes de *strings* empleadas en la aplicación. En este archivo se define el nombre de la aplicación.
  - **styles.xml*:*** Define el formato y la vista la interfaz de usuario.
- **Carpeta values-v21**: Contiene recursos que son utilizados cuando el dispositivo que está ejecutando la aplicación posee Android 5.0 o una versión posterior.
  - **dimens.xml**: Establece el margen de los *card views* de la interfaz de consola, en dispositivos con API nivel 21 o posterior.
  - **styles.xml**: Define el formato y la vista de la interfaz de usuario para dispositivos con API nivel 21 o versiones superiores, haciendo uso de *material design*. *Material design* es una guía para el diseño de interacción y movimientos en la interfaz visual de Android. Comprende un conjunto de pautas, componentes gráficos y estilos para crear aplicaciones que den al usuario una mayor sensación de naturalidad.

    Se emplea el fichero styles.xml para animar las transiciones entre interfaces, modificar la barra de estado, entre otros.

- **Alarma.cs**: Define la clase Alarma, como se describe en la sección 2.1.5.3.

- **Camara.cs*:*** Define la clase Camara.
- **ConectarActivity.cs*:*** Define la actividad ConectarActivity, asociada a la interfaz de conexión.
- **ConexionBluetooth.cs**: Define la clase ConexionBluetooth.
- **ConexionInternet.cs**: Define la clase ConexionInternet.
- **ConsolaActivity.cs*:*** Define la actividad ConsolaActivity, asociada a la interfaz de consola.
- **Controlador.cs**: Define la clase Controlador.
- **Dispositivo.cs*:*** Define la clase Dispositivo.
- **GestorComunicacion.cs**: Define la clase GestorComunicación.
- **GestorNotificacion.cs**: Define la clase GestorNotificación.
- **Luminaria.cs*:*** Define la clase Luminaria.
- **Puerta.cs**: Define la clase Puerta.
- **Radio.cs*:*** Define la clase Radio.
- **Sirena.cs**: Define la clase Sirena.
- **SplashScreenActivity.cs**: Define la actividad SplashScreenActivity, asociada a la pantalla de inicio.
- **Ventana.cs*:*** Define la clase Ventana.

ANEXO X

CÓDIGO DE LA APLICACIÓN PARA LA PLACA ARDUINO

El archivo se encuentra en el CD adjunto.

ANEXO XI

CONFIGURACIÓN DE CÁMARA DE VIDEO

Es necesario configurar la cámara de video para capturar imágenes al detectar movimiento. A continuación, se muestra el proceso de configuración de la cámara D-LINK 932L:

- Se coloca en el browser la dirección IP asignada por defecto a la cámara. Para ello se ingresa la dirección 192.168.0.20.
- En la pantalla que se muestra se introduce el usuario y contraseña de la cámara, se ingresa *admin* para ambos valores, ver Figura XI.1.

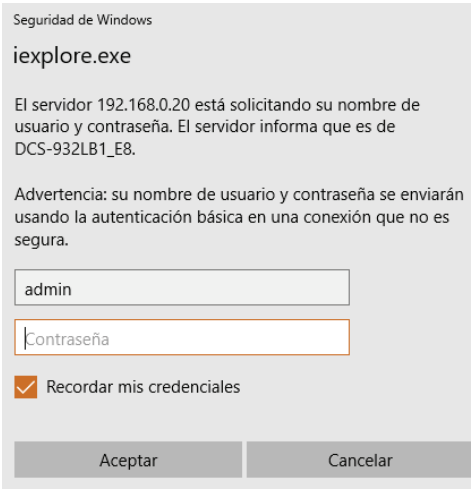

Figura XI.1 Acceso a la configuración de la cámara

- Se accede a la pestaña de configuración, donde es necesario seleccionar la opción “Detección de movimiento”. Se procede a activar la opción de captura de fotografías, para lo cual se selecciona la opción “Activar”, ver Figura XI.2.

Figura XI.2 Activación de la detección de movimiento

- Se procede a configurar el correo al cual se enviarán las imágenes capturadas. Para ello se accede a la opción “Correo”, donde se ingresa la dirección de un servidor SMTP y su número de puerto, como se muestra en la Figura XI.3.

Figura XI.3 Configuración del envío por correo electrónico

- Es necesario incluir la dirección de correo electrónico del remitente, desde la cual se enviará las imágenes capturadas por la cámara. Posteriormente, se ingresa la dirección de correo del destinatario, en la que se recibirá las imágenes, como se muestra en la Figura XI.4.

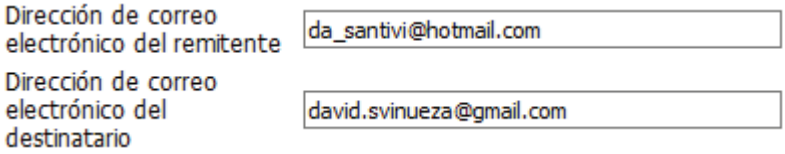

Figura XI.4 Configuración de remitente y destinatario

- Se procede a configurar el nombre de usuario y contraseña de la cuenta de correo electrónico del remitente, ver Figura XI.5.

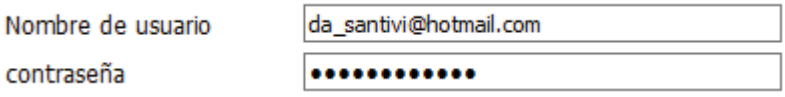

Figura XI.5 Configuración de usuario y contraseña de remitente

- Finalmente, se habilita la opción para el envío de imágenes capturadas al correo electrónico, como se muestra en la Figura XI.6.

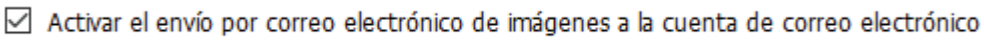

Figura XI.6 Activación del envío de correos electrónicos

ANEXO XII

CIRCUITOS ELECTRÓNICOS IMPLEMENTADOS

En la Figura XII.1 se muestra el esquema del circuito electrónico para la tarjeta de sensado de puertas/ventanas y control del botón de pánico. La placa incluye conectores para el sensor PIR y el sensor de temperatura LM35.

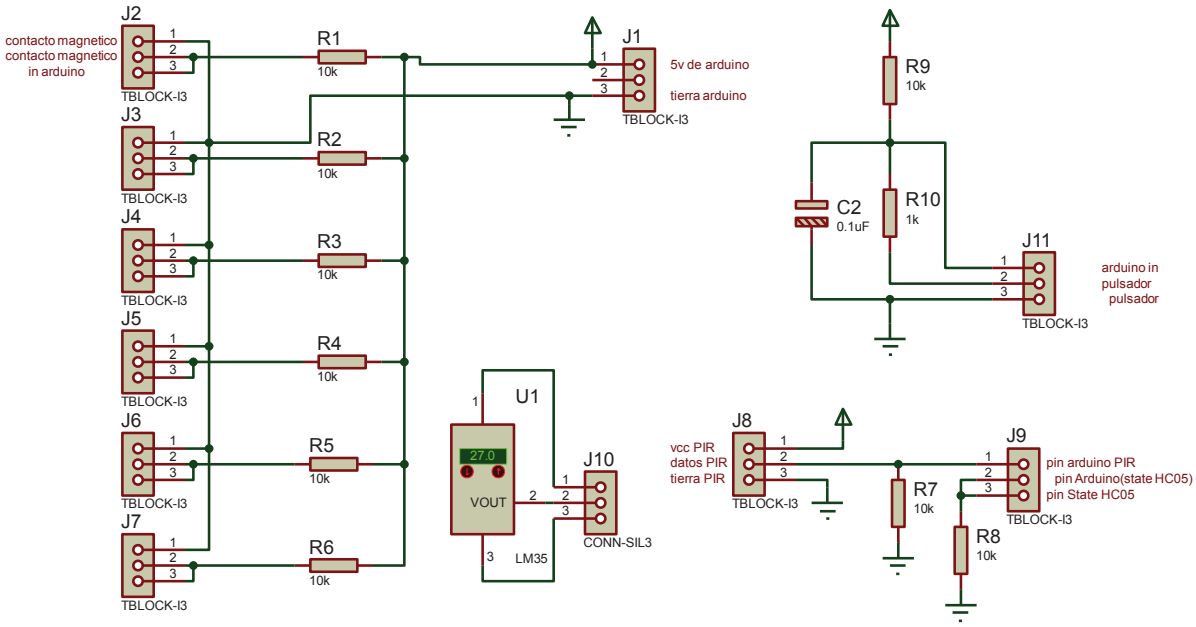

Figura XII.1 Tarjeta electrónica para sensado de puertas/ventanas y botón de pánico

El esquema del circuito electrónico de la tarjeta actuadora *On/Off* se indica en la Figura XII.2.

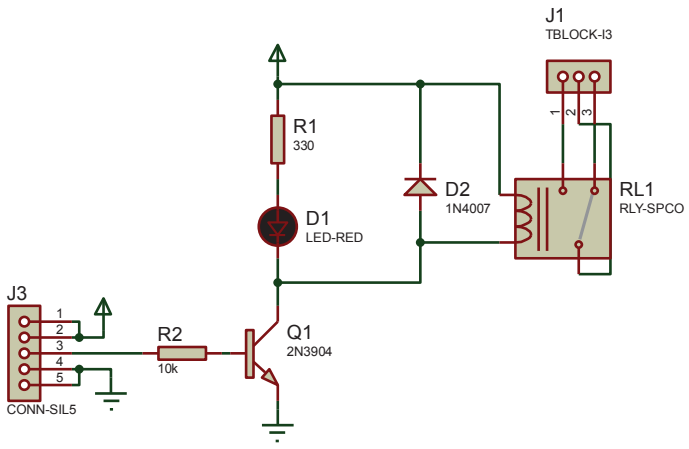

Figura XII.2 Tarjeta electrónica actuador *On/Off*

El esquema del circuito electrónico de la fuente de poder, tomado de [81], utilizado para alimentar todos los componentes del sistema se muestra en la Figura XII.3.

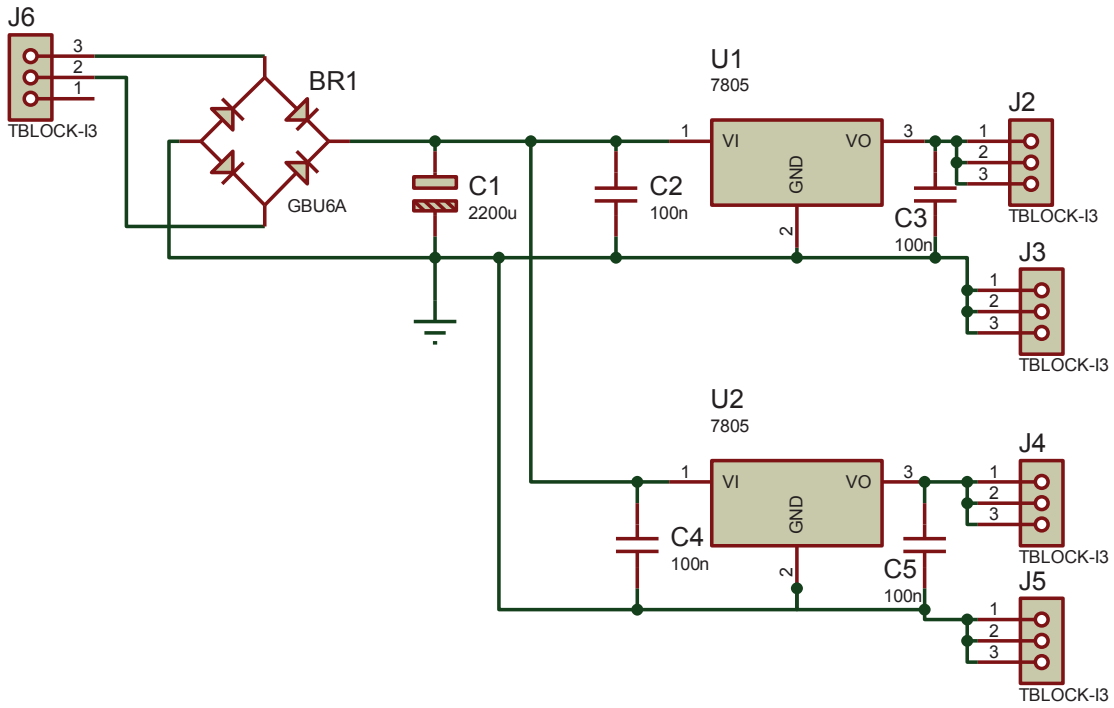

Figura XII.3 Circuito electrónico para la fuente de poder

ANEXO XIII

PISTAS DE LAS TARJETAS ELECTRÓNICAS

El circuito impreso para la tarjeta de sensado de puertas/ventanas y control del botón de pánico se muestra en la Figura XIII.1.

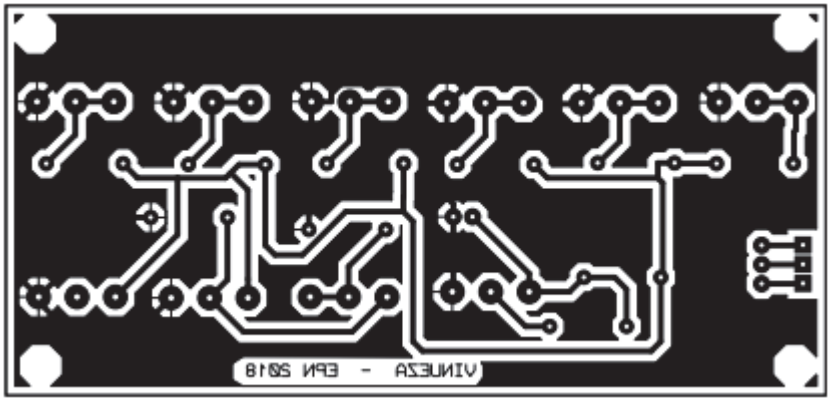

Figura XIII.1 Pistas para tarjeta electrónica de sensado y botón de pánico

El circuito impreso del actuador *On/Off* se muestra en la Figura XIII.2.

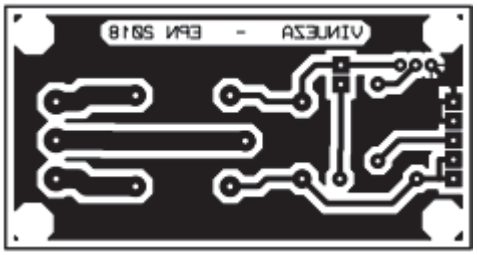

Figura XIII.2 Pistas para tarjeta electrónica actuador *On/Off*

El circuito impreso de la fuente de poder del prototipo se muestra en la Figura XIII.3.

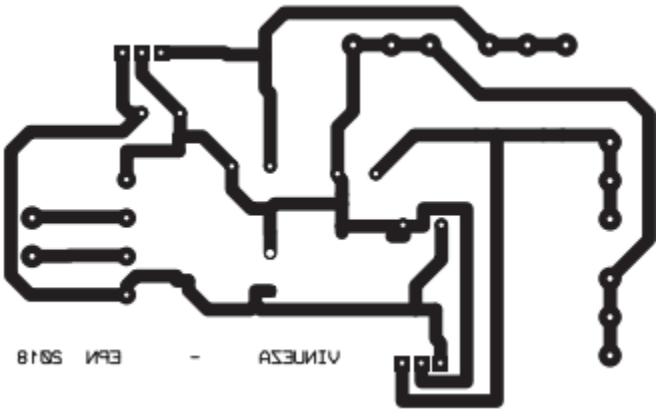

Figura XIII.3 Pistas para la fuente de poder

ANEXO XIV

CARACTERÍSTICAS CIRCUITO INTEGRADO LM78S05

El archivo se encuentra en el CD adjunto.

ANEXO XV

CARACTERÍSTICAS CIRCUITO INTEGRADO KBL400

El archivo se encuentra en el CD adjunto.

ANEXO XVI

MANUAL DE USUARIO

El archivo se encuentra en el CD adjunto.

ANEXO XVII

COSTO REFERENCIAL DEL PROTOTIPO

Para el costo referencial se considera el costo de los elementos de hardware empleados. Las siguientes tablas muestran el detalle del costo de los componentes utilizados en el desarrollo del prototipo:

Tabla XVII.1. Costo del mecanismo de sensado y botón de pánico

Mecanismo de sensado y botón de pánico			
Elementos	**Cantidad**	**Precio unitario ($USD)**	**Precio total ($USD)**
Bornera de 3	10	0,45	4,5
Regleta de conectores	2	1,5	3
Resistencia 1 $K\Omega$ ½ W	1	0,3	0,3
Resistencia 10 $K\Omega$ ½ W	9	0,3	2,7
Capacitor 0,1 uF	1	0,15	0,15
LM35	1	1,8	1,8
PIR HC-SR501	1	8,5	8,5
Contacto magnético	6	3,5	21
Pulsador	1	1	1
Cloruro Férrico	2	1	2
Impresión de pistas	2	1	2
Papel termotransferible	2	0,95	1,9
Baquelita	1	3,5	3,5

Tabla XVII.2. Costo del mecanismo actuador *on/off*

Mecanismo actuador *on/off*			
Elementos	**Cantidad**	**Precio unitario ($USD)**	**Precio total ($USD)**
Bornera de 3	8	0,45	3,6
Regleta de conectores	3	1,5	4,5
Resistencia 330 Ω ½ W	8	0,3	2,4
Resistencia 10 K Ω ½ W	8	0,3	2,4
Diodo 1N4007	8	0,1	0,8
Transistor 2N3904	8	0,1	0,8
Diodo Led	8	0,12	0,96
Relé 5 V	8	0,95	7,6
Cloruro Férrico	3	1	3
Impresión de pistas	2	1	2
Papel termotransferible	2	0,95	1,9
Baquelita	2	3,5	7

Tabla XVII.3. Costo de la fuente de poder

Fuente de poder			
Elementos	**Cantidad**	**Precio unitario ($USD)**	**Precio total ($USD)**
Rectificador KBL400	1	0,7	0,7
LM78S05	2	1,35	2,7
Bornera de 3	5	0,45	2,25
Disipador	2	0,9	1,8
Juego de disipador	2	0,5	1
Capacitor 2200 uF	1	1	1
Capacitor 100 nF	4	0,1	0,4
Cloruro Férrico	1	0,45	0,45
Impresión de pistas	1	1	1
Papel termotransferible	1	0,95	0,95
Baquelita	1	1,5	1,5

Tabla XVII.4. Costo de los cables empleados

Cables			
Elementos	**Cantidad**	**Precio unitario ($USD)**	**Precio total ($USD)**
Cables de Arduino	3	3,5	10,5
Cable de timbre	10	0,15	1,5
Cable utp cat6	20	0,6	12
Cable gemelo #20	10	0,75	7,5
Canaleta	2	2,5	5
RJ-45	4	0,35	1,4
Protector RJ-45	4	0,15	0,6
Patch cord	2	1	2
Tubo termo-contráctil	4	1,5	6
Cable de poder	1	2,5	2,5
Estaño	1	4	4
Caja tablero principal	1	15	15

Tabla XVII.5. Costo del hardware del prototipo

Elementos	**Costo ($USD)**
Sirena	20,5
Arduino Mega	32
Ethernet *shield*	30
Módulo HC05	12,5
Cámara D-LINK	69
Transformador 110 V	19,5
Mecanismo actuador *on/off*	36,96
Mecanismo de sensado y botón de pánico	52,35
Fuente de poder	13,75
Cables	68
Total	354,56

ORDEN DE EMPASTADO