ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Diseño de Sistema Domótico WIFI y Aplicación Androide Utilizando Hardware Idetec- Inventio"

INFORME DE PROYECTO DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN TELEMÁTICA

Presentada por:

John Alexander Arellano Riera

Nury Stefanía Cornejo Córdova

GUAYAQUIL-ECUADOR

2015

AGRADECIMIENTO

A Dios por darnos las bendiciones necesarias para el desarrollo del presente proyecto. A mis queridos amigos: Natalia Candell, Sasha Palacios, Lenin Montes, Freddy Quimis y Carlos Espinoza que me dieron su motivación y apoyo incondicional, preocupándose por el avance de este proyecto. A mi hermano menor, John Jairo, que siendo pequeño, se ha interesado en mis deberes como estudiante y me ayudó en lo que estaba en sus manos. A mi enamorada y compañera de proyecto, por su apoyo y arduo esfuerzo. Y un especial agradecimiento a nuestro director de proyecto, MsC. Víctor Asanza, que nos ayudó con sus conocimientos y colaboración para el desarrollo del mismo.

John Arellano Riera

AGRADECIMIENTO

A Dios por permitirme llegar al momento en el que culmina un ciclo más de mi vida. A mi bella familia padres y hermanos, quienes han estado a mi lado en el transcurso de mi carrera, por darme su amor, alentarme y apoyarme de todas las maneras posibles.

A nuestro director de proyecto, por su guía y ayuda en la elaboración del presente.

A mi enamorado por ser para mí un apoyo incondicional y creer en mí aun cuando yo no lo hacía. A mis amigos por hacer de mis días universitarios la mejor época de mi vida, en especial a mi querida Maricela Freire por ser para mí ejemplo de persona y profesional, por estar presente en los momentos felices de mi vida pero sobre todo en los momentos díficiles me ayudó a seguir luchando y no rendirme.

Nury Cornejo Córdova

DEDICATORIA

A mis queridos padres, que durante toda mi carrera me forjaron y me guiaron por el camino del bien, brindándome su apoyo incondicional, toda mi educación se la debo a ellos, los amo.

A mis queridos tios: Bella Nancy y Martino Guzzardella, por su apoyo y motivación para culminar mis estudios.

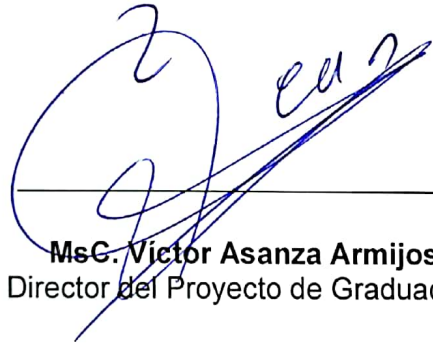
John Arellano Riera

A Dios porque me ha dado la fuerza y dedicación para culminar este proyecto. A mis padres: José Cornejo y Rosa Córdova, mis hermanos, cuñada y sobrinos porque han soportado mi ausencia en el transcurso de mi carrera y porque nunca perdieron la fe en mí.

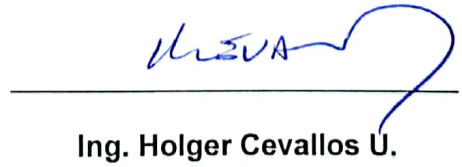
Nury Cornejo Córdova

TRIBUNAL DE SUSTENTACIÓN

PhD. Sixto García
Presidente



MsC. Víctor Asanza Armijos
Director del Proyecto de Graduación

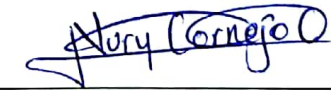


Ing. Holger Cevallos U.
Miembro Principal

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este informe, nos corresponden exclusivamente; y el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

Arellano Riera John Alexander



Cornejo Córdova Nury Stefanía

RESUMEN

El presente proyecto denominado “Diseño de Sistema Domótico WiFi y Aplicación Androide Utilizando Hardware IDETEC - Inventio”, se eligió con la finalidad de brindar seguridad, confort y eficiencia de consumo energético a los usuarios que deseen adquirir un sistema de automatización y control sobre sus hogares, tratando de generar la menor cantidad de cambios posibles en la infraestructura de los mismos. El presente sistema domótico planteado, es adecuado para implementarse incluso en viviendas ya construidas.

El capítulo 1, explica una breve descripción del problema, que existe en ciertos sistemas domóticos convencionales, dando una posible solución, justificando los principios para la solución del mismo. Se plantea los objetivos que se propuso, para el final del desarrollo del presente proyecto.

El capítulo 2, explica conceptos fundamentales de un sistema domótico, ya que en la actualidad no se tiene mucho conocimiento acerca de lo que significa y conlleva la domótica. Se detallan los estándares de sistemas domóticos más conocidos y usados en la actualidad. Finalmente se describe una clasificación de

los sistemas domóticos, donde se diferencian las ventajas y desventajas de cada uno.

El capítulo 3, describe conceptos con respecto a los sistemas domóticos centralizados, profundizando temas de interés para el desarrollo del presente proyecto, como por ejemplo el hardware utilizado con sus respectivos principios de elección.

El capítulo 4, explica teoría con respecto al sistema operativo Android, con su respectivo lenguaje de programación, dando las ventajas de usar esta plataforma, como parte del control en el sistema domótico.

El capítulo 5, detalla los principios necesarios y utilizados, para el desarrollo del sistema domótico propuesto, explicando conceptos asociados para el diseño del sistema domótico.

El capítulo 6, explica el desarrollo técnico y la implementación del presente proyecto, detallando las funcionalidades del sistema, de una manera técnica, con los principales métodos que se usaron para concluir el proyecto.

El capítulo 7, detalla las pruebas que se realizaron al sistema domótico, con los respectivos resultados obtenidos, tanto a nivel de aplicación androide, como a nivel de hardware.

ÍNDICE GENERAL

ABREVIATURAS Y SIMBOLOGÍA

WiFi: Es una marca comercial de WiFi-Alliance.

PDA: Asistente digital personal. (Personal Digital Assistant)

SFR: Registro de funciones especiales. (Special Function Registers)

PWM: Modulación por ancho de pulsos. (Pulse-width modulation)

PIC: Controlador programable de interrupciones. (Programmable Interrupt Controller)

IDE: Entorno de desarrollo integrado. (Integrated Development Environment)

UDP: Es protocolo de nivel de transporte. (User Datagram Protocol)

HTML: Lenguaje de marcas de hipertexto. (HyperText Markup Language)

TTL: Lógica de transistor a transistor. (Transistor-transistor logic)

CMOS: Semiconductor complementario de óxido metálico. (Complementary metal-oxide-semiconductor)

JDK: Es la plataforma de desarrollo de JAVA. (Java Development Kit)

WLAN: Es una red de área local inalámbrica. (Wireless local area network)

DHCP: Protocolo de configuración dinámica de host. (Dynamic Host Configuration Protocol)

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Para la presente propuesta de este proyecto, se pensó en la flexibilidad, adaptabilidad, movilidad y la fácil instalación de un sistema domótico, cuando se hace el uso de tecnologías inalámbricas, es por eso que se utiliza el protocolo ZigBee, por su poco consumo energético, entre otras características ventajosas. De igual manera la tecnología WiFi, que es muy común verla hoy en día en los hogares de las personas.

Al usar el medio inalámbrico, se puede gozar de la flexibilidad de instalación, evitando cambios drásticos en la infraestructura de hogares que ya se encuentran construidos. Este sistema aparte de contar con una instalación flexible gracias al uso de la tecnología inalámbrica, cuenta con la característica de movilidad, gracias al desarrollo de una aplicación para teléfonos inteligentes o Smartphones, con sistema operativo Android. Desde dicha aplicación se podrá controlar de manera manual, automática o programable, los elementos que se conecten al hogar, sin tener que movilizarse. Todo esto sin dejar a un lado las características de todo sistema Domótico, que son el confort, seguridad y ahorro energético para el usuario.

CAPÍTULO 1

1. ANTECENDENTES Y JUSTIFICACIÓN

En este capítulo se detallan los motivos por los que el presente proyecto es viable, en qué situaciones se verá aún más su utilidad y la forma en que se plantea su desarrollo.

1.1. Descripción del problema

En la actualidad las personas generalmente desean contar con comodidades que hagan más fácil su vida, estas pueden ser simplemente por lujo, o a veces por necesidad.

Uno de los principales problemas que contamos con los hogares de hoy en día es el uso ineficiente de la energía, o problemas de seguridad dentro y fuera del hogar. Los sistemas domóticos convencionales dirigidos a los hogares generan un poco de molestia a quienes quieren obtenerlos, ya que se deben hacer cambios en la infraestructura del hogar por el cableado que se tiene que colocar, llegando así a ser un servicio demasiado costoso. La seguridad es un aspecto importante en el hogar, por lo tanto tener un sistema domótico que brinde información a los usuarios sobre lo que sucede en sus hogares es tranquilizante hasta cierto punto, ya que les permitirá tomar decisiones de manera manual o automática dentro de sus hogares, según la información que reciban.

1.2. Justificación

La presente propuesta de proyecto fue elegida con la finalidad de ayudar a brindar un mejor estilo de vida, mediante la comodidad, seguridad y demás servicios que nos ofrece la Domótica, ayudando a las actividades que las personas realizan dentro de sus hogares. Así mismo ayudar a las personas, para que puedan ejecutar

acciones de control en sus hogares, de manera fácil y accesible, llegando a un ahorro energético. Gracias a las ventajas como ejercer control, mediante la automatización que nos ofrece la domótica, aprovechando la tecnología WIFI, evitaremos cambios drásticos en la estructura del hogar, como normalmente ocurren en los sistemas domóticos convencionales de hoy en día.

1.3. Solución propuesta

Lo que proponemos es un sistema domótico que permita realizar todo o la mayoría de acciones de control que nos ofrecen los sistemas domóticos actuales, pero con la ventaja de que será un sistema parcialmente inalámbrico, por lo cual generará menos cambios en la estructura del hogar.

Además el hecho de que los controles puedan ejercerse desde el celular, le da movilidad al usuario al momento de realizar una acción, y no tener que acudir al tablero de controles.

1.4. Objetivos

Diseñar un sistema domótico bajo aplicación Android y hardware CONSORCIO IDETEC-INVENTIO, capaz de automatizar y controlar las funciones que se realizan dentro de un hogar, que permita un control eficiente de ciertos servicios básicos, a fin de brindar confort, seguridad y eficiencia energética dentro de su hogar.

Creación de aplicación Android, con capacidad de configurar funciones del hogar de manera: manual, automática o programada.

Automatizar el encendido y apagado de la iluminación y artefactos, abrir-cerrar puertas, activar o desactivar sistemas de riego, etc.

Capacidad para registrar y respaldar usuarios en una base de datos local, con sus respectivos roles o privilegios en el hogar.

Implementación de los módulos concentradores y actuadores del CONSORCIO IDETEC-INVENTIO, para la ejecución de las diferentes tareas automatizadas.

Generar reportes de actividades de los servicios implementados, visibles para el usuario a nivel de la aplicación.

Sistema con capacidad para medir parámetros ambientales, como son la temperatura, humedad, humo, etc., que permitan tomar decisiones, para salvaguardar la integridad del hogar.

1.5. Alcance

El alcance de este proyecto contempla realizar un sistema domótico con el fin de controlar elementos de un hogar mediante una aplicación androide. Para lo cual implementará un módulo concentrador, módulos actuadores y módulos sensores, que se comuniquen inalámbricamente. El módulo concentrador está

conformado por un microcontrolador, módulo Ethernet y módulo XBee, que se comunica con la aplicación mediante tecnología WiFi. La comunicación entre el módulo concentrador, los módulos actuadores y módulos sensores es inalámbrica, mediante el protocolo Zigbee con el que trabajan los módulos XBee.

1.6. Metodología

En el presente proyecto como primera instancia se realizará un análisis sobre los servicios que pueden ser automatizados dentro de un hogar, para luego realizar un diseño adecuado que cubra todas o la mayoría de estos servicios. Según este diseño, realizar la programación de una aplicación ANDROID que pueda ser usada de manera más sencilla, ya sea visualmente o por comandos de voz. Realizar la programación de hardware del CONSORCIO IDETEC-INVENTIO, de tal manera que las acciones que realicen coincida con las tareas que está pidiendo el usuario. Implementar un puente de comunicación mediante la tecnología WIFI, con estos nos referimos a los Sockets que se enviarán desde la aplicación al hardware, para activar los actuadores de los dispositivos finales. Por

último realizar las pruebas y correcciones necesarias hasta que se verifique que el sistema domótico está actuando de manera adecuada y acorde con lo que se esperaba.

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS DE LOS SISTEMAS DOMÓTICOS

En el presente capítulo se explicará que es la domótica, los diferentes tipos de sistemas domóticos existentes, y los factores que se deben considerar para la elección de un tipo de sistema domótico u otro.

2.1. Introducción a la domótica

Los enormes y continuos avances tecnológicos, permiten que las funciones del hogar puedan realizarse de manera automatizada, lo que incluye seguridad, ahorro de energía, comunicación remota con ciertos elementos del hogar, etc. Este tipo de integración, “tecnología – hogar”, es lo que solemos denominar domótica. Sin embargo, “el término domótica nace del neologismo francés

‘domotique’, que procede de la palabra latina domus (casa) y del francés telematique (telecomunicación informática)” [1].

La domótica también es conocida como “hogar inteligente” u “hogar digital”, ya precisamente gracias a la digitalización es que podemos ejercer control y automatización en el hogar. Las tareas automatizadas no son muy lejanas, ya que todos hemos podido observar alguna vez, en centros comerciales o hoteles, como alguna puerta se abre de manera automática. El objetivo de la domótica es integrar todos los aparatos del hogar, de manera que funcionen en armonía, asegurando su máxima utilidad y a la vez la mínima intervención del usuario.

2.1.1. Concepto de hogar digital

El concepto de hogar digital trata de acogerse a un significado más amplio del que tiene la domótica en sí. En este concepto se desea abarcar tanto la automatización de las tareas del hogar, así como la comunicación remota con la mayoría de

los elementos que conforman el mismo. Con este concepto nace el hecho de ahorro de energía y costos para el usuario, seguridad del hogar, aumentar confort, uso de la tecnología para ofrecer nuevos servicios, todo con el fin de mejorar la calidad de vida.

La convergencia entre entretenimiento, comunicación y gestión del hogar de manera digital se logra cada vez de manera más completa gracias a las tecnologías de infraestructura y soporte para la automatización de servicios en el hogar. El papel del Internet es muy importante en el hogar digital, ya que ha mejorado la capacidad de crear, transmitir, almacenar y procesar información, todo posible gracias a la digitalización, he ahí el término hogar digital.

2.1.2. Componentes de un sistema domótico

Los principales componentes del sistema domótico son el centro de control, actuadores, sensores e interfaz de usuario.

Cada uno de ellos juega un papel importante para el adecuado funcionamiento del sistema domótico.

- El centro de control es el encargado de receptar información, procesarla y tomar decisiones de acción, según lo que tenga programado. Existen dos principales tipos de centro control, uno es integrado en un panel y el otro se refiere a módulos lógicos programables. El control integrado en un panel, para su correcto funcionamiento necesita que se le adecue un espacio y suministro de energía en el hogar, su principal característica es la rapidez con que procesa la información. Los módulos lógicos programables son los más usados en el control de procesos industriales debido a que tienen más de capacidad de programación y pueden ser adaptables a condiciones futuras. Su principal ventaja es que sus medios de comunicación son estandarizados. Su respuesta a tareas es más limitada con respecto al panel de control, sin embargo es una opción más económica y

con la potencia suficiente para ser personalizado a las necesidades del usuario.

- Los actuadores son dispositivos que controlan algún elemento específico del sistema, transforman la energía de un proceso en una acción controlada y pueden estar o no integrados al sistema de control principal. Los actuadores pueden ser preaccionadores o accionadores, entre los cuales encontramos relés o contactores, motores eléctricos, electroválvulas, cerraduras eléctricas, que se encargan de proporcionar la energía necesaria para llevar a cabo una acción como lo hacen los relés, o también llevar a cabo directamente una acción como abrir o cerrar una puerta como lo hace una cerradura eléctrica.

- Los sensores estén en contacto con el exterior e interior del hogar, receptan la información del medio y la envían al controlador central para que sea procesada y éste a la vez envié una señal al actuador

correspondiente. Los sensores juegan un papel importante en el sistema domótico, ya que la información que receptan es la que permitirá llevar a cabo una acción, como el encendido o apagado de luces, la activación de una alarma, etc.

- La interfaz de usuario permite la interacción entre el usuario y el sistema domótico, por los tanto debe ser entendible y de fácil utilización. La interfaz de usuario varía según la arquitectura del sistema, lo tradicional es un panel con botones o de pantalla táctil para accionar los elementos de hogar, sin embargo hace un tiempo se ha ido integrando como interfaz de usuario el uso de interfaces Web, PDA's, posicionamiento, comandos de voz, dispositivos móviles. El uso del internet le permite al usuario tener control del hogar tanto de manera presencial como remota, lo cual es una limitación superada en las interfaces de usuario tradicionales.

2.1.3. Arquitectura de los sistemas domóticos

La arquitectura de un sistema domótico nos indica la forma en que se ubicarán los elementos de control del sistema. Se dice que la clasificación de la arquitectura del sistema domótico depende de donde se encuentre la “inteligencia” del sistema, a continuación se detallan las cuatro arquitecturas de hoy en día.

- Arquitectura centralizada: En este tipo de arquitectura el controlador del sistema se encarga de la gestión, proceso y acciones de los elementos del sistema, en el que se incluyen interfaces, actuadores, sensores, etc. De entre las mayores ventajas de este tipo de arquitectura están su potencia, versatilidad e inteligencia, ya que suelen ser manejados por procesadores potentes, que permiten flexibilidad de programación y velocidad para el manejo de información de grandes sistemas. Obviamente al ser

una arquitectura centralizada su mayor desventaja es que, al faltar el controlador del sistema, todo dejará de funcionar.

- Arquitectura distribuida: En este tipo de arquitectura cada sensor y actuador tiene situado cerca de él un controlador. El sistema domótico puede ser de arquitectura distribuida en cuanto a la ubicación física de los elementos de control, pero no lo son en cuanto a la capacidad de proceso y viceversa. Una de sus grandes ventajas es la autonomía que brinda a los elementos del sistema, esto implica que si alguna parte del sistema deja de funcionar no significa que el resto lo haga. Debido a esta individualidad se tiene como dificultad la programación de los elementos de control, y el hecho que para adicionar funciones al sistema se deben adquirir nuevos módulos de control y actuación.

- Arquitectura descentralizada: En este tipo de arquitectura nos encontramos con varios controladores conectados con actuadores. Los controladores pueden comunicarse entre ellos, interpretar y procesar la información que reciben de los sensores o las interfaces de usuario.

- Arquitectura híbrida/mixta: Esta arquitectura combina la centralizada, descentralizada y distribuida, ya que puede tener ubicados varios controladores con actuadores que pueden recibir y procesar la información enviada por los sensores y comunicarse con los dispositivos distribuidos por la vivienda. Los controladores pueden comunicarse o no con un controlador central, dependerá de la programación. Este tipo de arquitectura es la usada en los sistemas domóticos inalámbricos que suelen usar ZigBee.

2.1.4. Características de un hogar inteligente

Cuando se menciona hogar inteligente, los primeros pensamientos son de confort, seguridad, ahorro energético, estas son las principales características que brinda un hogar inteligente, que a la vez son las generan los beneficios de adquirir un sistema domótico. Para el usuario es importante saber lo que sucede en su hogar a cada momento, por lo mismo el acceso remoto o el uso de alertas al usuario deben estar presentes cuando se piensa en la implementación de un sistema domótico

2.1.5. Gestión de la domótica

La gestión de la domótica indica las tareas de las que se hace cargo, para evitar que el usuario las realice, además de su intervención en la seguridad y el ahorro energético, factores importantes en un sistema domótico.

- Programación y ahorro energético: En este aspecto la domótica no exige un cambio de los electrodomésticos convencionales para poder lograr un ahorro energético, ya que se centra más en la gestión y uso eficiente de los mismos. Parte de la gestión se basa en las tarifas de consumo de ciertos electrodomésticos, dejándolos funcionar en ciertas horas, el uso de sensores para la activación de ciertos elementos, únicamente cuando sea necesario, como es el caso de la iluminación.

- Seguridad: Lo central de este punto es el uso de alarmas para la protección del hogar frente a intrusos. Sin embargo la seguridad no solo se refiere a la protección de los bienes patrimoniales, sino a la seguridad personal y la vida.

  Los sistemas de seguridad dentro del hogar dependiendo del contexto pueden ser: alarmas de intrusión para las que se emplean detectores de

movimiento, presencia, etc, alarmas técnicas que se encargan de la seguridad frente a incendios, fugas de gas, fallas del suministro energético, etc.; alarmas personales que se refiera a las alertas médicas que puedan tener los miembros del hogar, haciendo posible que reciban tele asistencia; vigilancia remota que se logra con el uso de cámaras IP.

- Confort: Se refiere a la comodidad y bienestar que el sistema domótico genera al usuario. En este aspecto se encuentran la automatizaciones de ciertas tareas, como el encendido y apagado de luces, la activación del sistema de riego mediante un sensor de humedad que detecte sólo cuando sea necesario, etc.

- Comunicación: Se refiere a los sistemas del hogar que permiten la comunicación ya sea entre usuario y equipos domésticos, equipos y equipos, de manera presencial o remota. También se incluyen las infraestructuras de telecomunicaciones necesarias

para la transferencia de voz, video y datos, el acceso al sistema domótico de varios usuarios de manera simultánea.

- Accesibilidad: Incluye las aplicaciones o sistemas de control remoto que dan autonomía al usuario. En el diseño del hogar inteligente se debe tener en cuenta todos los posibles usuarios y dar facilidades de uso para que brinden autonomía necesaria especialmente a usuarios con capacidades especiales, ya sea por discapacidad, enfermedad o envejecimiento.

2.1.6. Estándares de sistemas domóticos

Los principales estándares de control de sistemas domóticos son:

- X-10: Fue el primer estándar en desarrollarse. Su principal característica es la facilidad de instalación y

uso, por ese motivo es conocido como un sistema domótico sin instalación. La transmisión de datos es de baja velocidad y se realiza mediante el uso de la instalación eléctrica, lo que evita la instalación de cableado, logrando que la implementación del sistema domótico sea de bajo costo. En el protocolo X-10 existen tres tipos de dispositivos X-10: Los que sólo pueden transmitir órdenes; Los que sólo pueden recibirlas; Los que pueden enviar/recibir.

- EIBus: Es un sistema domótico propuesto en Europa que se basa en un Bus de datos para lo cual utiliza su propio cableado. Es el más usado para instalaciones en lugares de grandes extensiones como hoteles, centros deportivos, viviendas. A nivel físico se basa en el modelo OSI por tanto maneja una arquitectura descentralizada y los tiempos de montaje del sistema son bajos. Maneja prioridades logrando que los procesos se ejecuten rápidamente.

- LonWorks – LonTalk: Ofrece una plataforma completa para la conexión de redes y su control. Sus sistemas de control pueden ser utilizados casi en cualquier ambiente, como edificios, fábricas, trenes, viviendas, etc. Una de sus más importantes características es la flexibilidad, ya que la plataforma LonWorks es independiente de los medios de comunicación, por lo tanto puede trasportar información por casi cualquier medio, incluyendo PL, TP, RF, Coaxial, IR, y FO.

- Konnex: Este estándar es muy dinámico y permite que los distintos fabricantes que lo componen trabajen sin interferir el uno con el funcionamiento del otro e incluso colaborándose entre sí, usa un cable bus dedicado para cada servicio que se desee controlar, garantizando la calidad de las señales emitidas, haciéndolo el modelo de transmisión más seguro y robusto. La desventaja de este estándar es que para su correcto funcionamiento el bus de comunicación debe estar correctamente preinstalado, por lo que

implicaría cambios en la estructura del hogar y por ende más gastos.

2.1.7. Tipos de sistemas domóticos

Según su topología, que se refiere a la forma en que la red está diseñada, ya sea físicamente, con respecto a cómo está ubicado el hardware, o lógicamente respecto a cómo funciona el software y la comunicación que se genera entre todos los elementos del sistema.

La topología de una red se representa por la interacción que tienen todos los elementos del sistema domótico, es decir la forma en que se relacionan y comunican los dispositivos finales, los actuadores y el controlador.

- Topología estrella

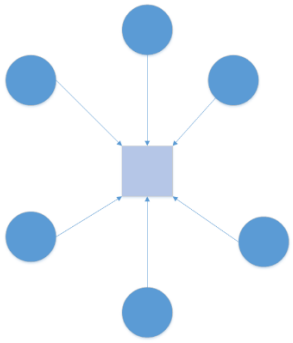


Figura 2.1 Diagrama de topología estrella

En esta topología se pueden encontrar enlaces punto a multipunto, donde todos los dispositivos del sistema se conectan al único controlador, también denominado concentrador, y no existen enlaces directos de los dispositivos entre sí, para que estos puedan comunicarse el concentrador actuará como intermediario. La principal ventaja de esta topología, es que si uno de los dispositivos del sistema deja de funcionar no afectará a los demás. La topología estrella es la que se usa en los sistemas domóticos centralizados.

- Topología anillo

Figura 2.2 Diagrama de topología anillo

Esta topología se basa en enlaces punto a punto, ya sean físicos o lógicos, entre un dispositivo y sus vecinos inmediatos, los que están a sus lados. Para comunicarse entre sí deben hacerlo de dispositivo en dispositivo hasta alcanzar su destino. Las desventajas de esta topología se encuentran relacionadas con el aspecto físico, refiriéndonos a la longitud del anillo, y el aspecto del tráfico, debido a que en el anillo siempre estaría circulando una señal.

- Topología bus

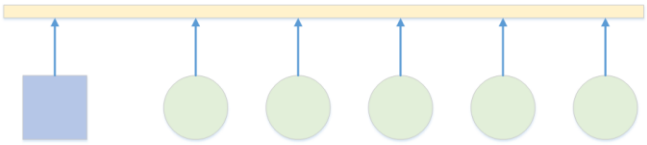


Figura 2.3 Diagrama de topología bus

Esta topología es multipunto, todos los dispositivos se conectan a un cable largo, denominado como red troncal. El cable de conexión es aquel que enlaza a los dispositivos de la red con el cable troncal. La principal ventaja de la topología bus es su sencillez de instalación, ya que si se pasa el tendido cable troncal de manera eficiente, se logrará conectar todos los dispositivos al troncal usando menos cableado que con una estrella o anillo.

2.2. Medios de transmisión

Son los medios físicos por el cual los elementos del sistema domótico intercambiaran información. De acuerdo al sistema que se desee implementar, debemos tener conocimiento de cuáles son los medios de transmisión disponibles y sus características, para así escoger el más adecuado para nuestro sistema domótico.

2.2.1. Conceptos del medio alámbrico

El medio de transmisión alámbrico, llamado también guiado, está constituido por un cable que conduce o guía las señales de un extremo al otro de la transmisión. En este medio, la distancia entre los terminales influye directamente en la velocidad de transmisión. Las características principales de los medios alámbricos son:

- Tipo de conductor usado.

- Velocidad máxima de transmisión.
- Inmunidad frente a interferencias electromagnéticas.
- Facilidad de instalación.
- Soporte de diferentes tecnologías a nivel de enlace.

Los medios alámbricos usados en los sistemas domóticos son:

- Cable dedicado

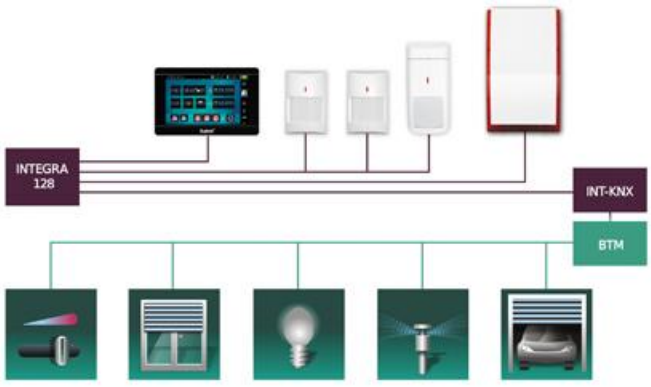


Figura 2.4 Sistema de cable dedicado [7]

Son aquellos sistemas que usan cables específicos o dedicados para la transmisión de órdenes. Son más caros con respecto a otros sistemas en cuanto a los

dispositivos de control, licencias de software, a cambio de ese costo brindan seguridad y robustez en la transmisión. Los cables más usados para este tipo de sistemas son, de pares trenzados, paralelo, coaxial, fibra óptica.

- Sistemas de corrientes portadoras

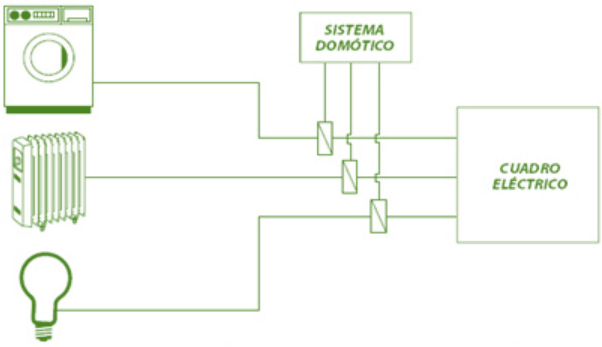


Figura 2.5 Sistema de corrientes portadoras [8]

Estos sistemas usan el cable de alimentación de los elementos para el envío de órdenes de control, en primera instancia se podría pensar que es el más adecuado ya que todos los elementos poseen una fuente de alimentación y por tanto se tendría acceso

total a los elementos del hogar, sin embargo debemos tener en cuenta que la alimentación suele ser inestable en algún momento, lo que generaría en nuestro sistema falsos positivos (órdenes al azar generadas automáticamente) que reducen la calidad de servicio del sistema domótico. Por la razón antes mencionada es que este tipo de sistemas es más usado para órdenes de control que no son críticas o cuando el uso de cable dedicado eleva demasiado el presupuesto como es el caso del control de la iluminación urbana.

2.2.2 Conceptos del medio inalámbrico

El medio de transmisión inalámbrico, llamado no guiado, debido a que no existe ningún medio tangible por el cual se transmita la señal de un extremo a otro de la comunicación. Los obstáculos o interferencia existentes en el medio generan ciertos problemas de transmisión, por lo que resulta importante el espectro de frecuencias en que se transmite la señal.

El medio inalámbrico, según el rango de frecuencias de trabajo, puede clasificarse en tres tipos:

- Radio.
- Microondas.
- Luz (láser o infrarrojos).

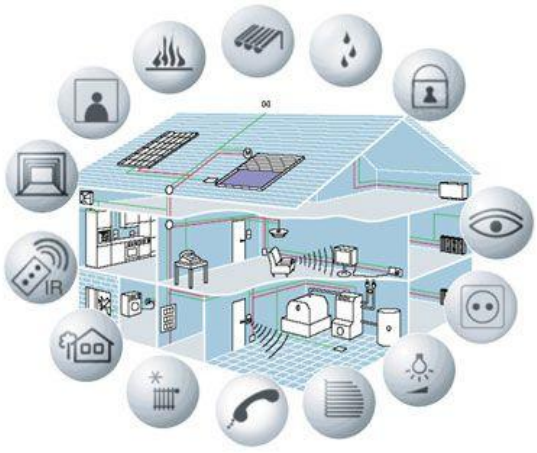


Figura 2.6 Sistema domótico inalámbrico [9]

Los sistemas domóticos con medios de transmisión inalámbrico usan ondas de radio frecuencia, con emisores y

receptores, que reciben, interpretan y de ser necesario reenvían las órdenes, lo que se conoce como red Mesh, por lo tanto no utilizan cables para transmitir las señales. Su mayor ventaja es que pueden ser instalados en proyectos de hogares en construcción u hogares ya construidos, debido a que no requieren de un proceso de preinstalación. La desventaja de este tipo de sistemas es la necesidad de mantenimiento de energía en los dispositivos, sin embargo existen varios métodos de auto alimentación energética para contra restar este efecto negativo, entre los cuales tenemos, el uso de células fotovoltaicas o inducción electromagnética.

Los sistemas domóticos mixtos combinan los medios de transmisión alámbrico e inalámbrico, con la finalidad de satisfacer la mayor cantidad de necesidades del usuario. En esta combinación de sistemas se realiza la programación de toda la instalación en un único sistema denominado MASTER, los otros sistemas con distintos protocolos son dependientes de éste y se denominan ESCLAVOS. El

intercambio de información y la realización de acciones programadas en el MASTER se llevan a cabo mediante pasarelas de comunicación.

2.3. Análisis de las diferentes áreas de mercado de la domótica

Este análisis está dirigido a las áreas de mercado de la domótica presentes en el Ecuador, analizar la oferta y la demanda de este mercado en la actualidad del país.

2.3.1. Análisis de diferentes marcas en el Ecuador

En el Ecuador existen varias empresas, que se dedican y están inmersa en el gran mundo de la automatización de hogares, o la domótica. Estas empresas son:

- SODEL: Es una empresa que se dedica a la comercialización de equipos de domótica, dado que

son importadores directo y autorizados de “HDL Buspro”, es decir ellos no son desarrolladores del sistema. También se dedican a dar cursos de capacitación.

- ISDE: Es una empresa fundada en el 2005 como filial para el Ecuador de ISDE España, dicha empresa se dedica a importar y distribuir equipos electrónicos de control. Esta empresa utiliza la tecnología LonWorks.

- Smart Controls: Empresa que no solo comercializa sistema domóticos, si no también inmóticos y urbóticos.

- Batel: Empresa reconocida en Ecuador, que no solo comercializa productos de domótica, también de seguridad industrial.

2.3.2. Oferta

Las empresas mencionadas en la sección 2.3.1, ofrecen a sus clientes ofertas de sistemas, a manera de kits, ya sean estos básicos o completos, también comercializan sistemas más robustos, para empresas industriales. Dado que el mercado se está haciendo más competitivo, las empresas tienden a mejorar sus servicios, sin dejar a un lado el aumento de la oferta para el cliente final.

2.3.3. Demanda

Según las investigaciones realizadas, las personas visualizan más la domótica como una fuente de ahorro y bienestar, que como un lujo, es por eso está creciendo la compra de sistemas domóticos, de todas las escalas. Dado esto existe una mayor demanda de compra de sistemas domóticos en el mercado, que nos impulsó a desarrollar este proyecto.

2.3.4. Estadísticas de sistemas domóticos convencionales, usados en el Ecuador

Por medio de un cuadro estadístico, se representa cuáles de los distribuidores de sistemas domóticos vigentes en Ecuador son los que tienen más demanda. Se realizó esto, en base a los productos que ofertan estas empresas nombradas en la sección 2.3.1 y la demanda que sobre los mismos que en la actualidad existe en el país, recalcando que este mercado en el país es poco conocido, o poco accesible a nivel económico.

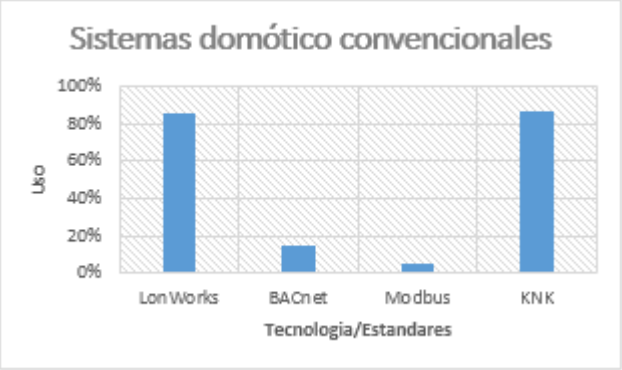


Figura 2.7 Estadísticas de sistemas domóticos convencionales

CAPÍTULO 3

3. TEORÍA DE LOS SISTEMAS CENTRALIZADOS

El presente capítulo describe las características más importantes de este tipo de sistema, además se indica para qué tipo de viviendas es adecuada su instalación.

3.1. Conceptos asociados a este sistema

En este tipo de sistema el controlador se encarga de la gestión, proceso y acciones de los elementos del sistema, en el que se incluyen interfaces, actuadores, sensores, etc. De entre las mayores ventajas de este tipo de sistema están su costo reducido, debido a

que los elementos y actuadores son de tipo universal, versatilidad e inteligencia; suelen ser manejados por procesadores potentes, que permiten flexibilidad de programación y velocidad para el manejo de información de grandes sistemas; su instalación suele ser relativamente sencilla y su interfaz de usuario suele ser de fácil manejo.

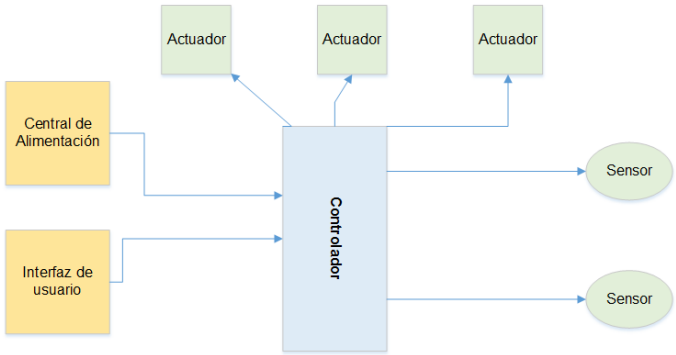


Figura 3.1 Diagrama de sistema centralizado

Por ser un sistema centralizado su mayor desventaja es que, al faltar el controlador del sistema, todo dejará de funcionar. Al requerir de significativas cantidades de cableado para su instalación, este tipo de sistema no tiene una alta escalabilidad. Agregar más elementos de control depende de la capacidad de canales o puntos con los que se implemente la instalación.

3.2. El concentrador y su rol principal

Se define como concentrador al micro controlador principal, o cerebro del sistema quien es el encargado de recibir peticiones de control o acción por parte de la aplicación Android y retransmitirla al o los dispositivos finales que hicieron la petición de acción. Este módulo también será el encargado de entregar información por parte de los sensores a la aplicación Android, para luego realizar una petición de acción con respecto a la información recibida y procesada. El microcontrolador que se usó para desarrollar este módulo concentrador, es el PIC 18F4520, de la familia de microchip. La programación del microcontrolador mencionado se encuentra en el Anexo A.

Figura 3.2 PIC18F4520 [10]

3.2.1. Familias de microcontroladores

Microchip tiene entre su familia de microcontroladores las series: 10F, 12F, 16F, 18F. En el presente proyecto se usa un microcontrolador de la familia 16F que es el 16F886, para módulos actuadores y módulos sensores, en el caso del concentrador se usa un microcontrolador de la familia 18F que es el 18F4520 descrito en el punto 3.1.

3.2.1.1. Arquitectura de los microcontroladores

Se describe las principales arquitecturas de los microcontroladores.

Arquitectura Von Neumann

La arquitectura tradicional de computadoras y microprocesadores está basada en la arquitectura Von Neumann, en la cual la unidad central de proceso

(CPU), está conectada a una única memoria donde se guardan las instrucciones del programa y los datos.

Figura 3.3 Arquitectura de Von Neumann

Arquitectura Harvard

La arquitectura Harvard tiene la unidad central de proceso (CPU) conectada a dos memorias, una con las instrucciones y otra con los datos, por medio de dos buses diferentes.

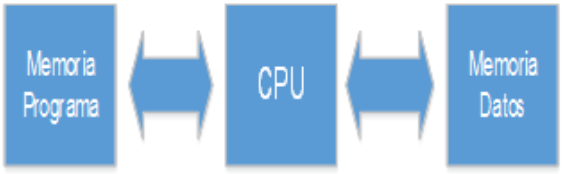


Figura 3.4 Arquitectura Harvard

3.2.1.2 Características y beneficios de la familia 18F

El PIC18F4520 tiene dentro de su memoria de datos, los SFR (Special Function Registers) a traves de los cuales controlamos la mayor parte de las funciones de este microcontrolador y sus perifericos. Cada periférico tiene asociado dos o tres SFR a través de los cuales se controla. Por ejemplo los puertos B de entrada/salida: TRISB, PORTB, LATB, donde:

- TRISB determina si es entrada (1) o salida (0)
- En PORTB leemos valores del puerto.
- LATB asigna valores a un puerto.

Comunicaciones USART:

- TXREG, TXSTA: datos a transmitir, status/configuración de TX.
- RCREG, RCSTA, datos recibidos, status/configuración de RX.

- BRGH establece la velocidad del puerto (baudios).

Frecuencia de trabajo	Hasta 40 Mhz, 10 Mhz en ciclos de instrucción.
Comparadores	2 comparadores, 2 canales PWM
Conversores analogicos	13 canales ADC, de 10 bits de resolución
Comunicaciones serie	UART (RS232,R485), y síncrona SPI, I2C
WatchDog Timer	Hasta 130 segundos
Multiplicador de hardware	8x8 en un solo ciclo.
Fuentes de interrupción	20 fuentes, con dos niveles de prioridad.
Memoria (Ram)	1536 bytes
Memoria EEPROM	256 bytes EEPROM no volatil (como un periférico)

Memoria Flash	32K memoria flash de programa reprogramable (16K instrucciones)

Tabla 3.1 Características de PIC18F4520

En la tabla 3.1 se muestra las caracteristicas principales, del PIC 18F4520, como son frecuencia de trabajo, conversores, comunicaciones en serie, memoria, etc.

Temporizadores del PIC18F4520

Un temporizador es un contador que se incrementa en cada ciclo de maquina, aunque en ciertos casos se puede configurar para contar una entrada externa. En los microcontroladores un ciclo de maquina son cuatro oscilaciones del reloj, por ejemplo en un PIC que trabaje a una frecuencia de 4 Mhz, los temporizadores se incrementaran cada microsegundo.

Este PIC cuenta con cuatro temporizadores, de los cuales algunos de ellos pueden ser usados por otras funcionas, de manera transparente para el usuario, como por ejemplo el PWM del pic, que usa el temporizador 2. Los temporizadores pueden trabajar a modo de 8 y 16 bits, pudiendo asi definir en que momento activar la interrupción correspondiente.

Interrupciones del 18F4520

Como se dijo anteriormente el PIC18F4520 consta con veinte fuentes de interrupción y dos niveles de prioridad, es decir interrupciones altas pueden interrumpir a las bajas.

4 interrupciones por desbordamiento de temporizadores
4 interrupciones causadas por cambios de un pin (RB0, RB1, RB2, RB4-7)

Interrupciones asociadas a periféricos, por ejemplo:
• TX el registro de transmisión está vacío y listo para enviar un byte. • RX acabamos de recibir un byte. • AD ha terminado una conversión AD. Podemos acceder al resultado.

Tabla 3.2 Tipos de interrupciones del 18F4520

3.2.2. Entorno de desarrollo de software MikroC PRO for PIC

Para programar los Microcontroladores PIC18F4520 y PIC16f886, correspondientes al módulo concentrador y módulo actuador de dispositivos finales, se usó del IDE de desarrollo propio del fabricante Microchip, “mikro C for PIC”, el cual tiene una versión demo de prueba, limitada, y una completa pero con un costo.

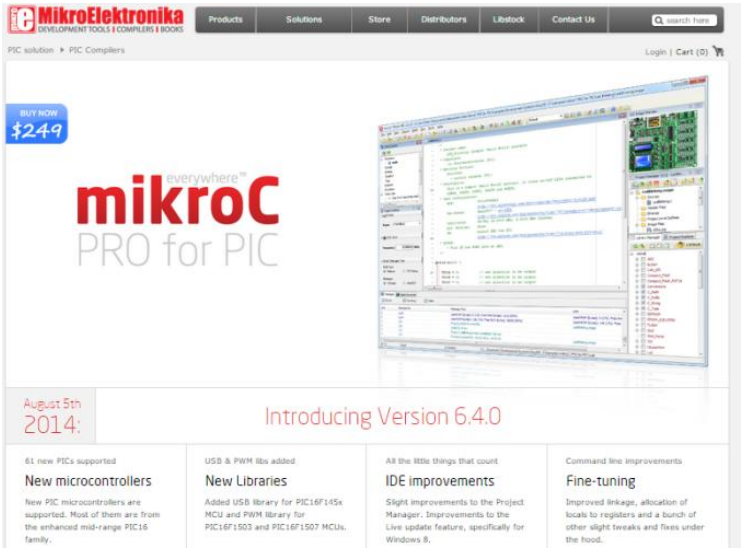


Figura 3.5 Pagina web de MikroC, para descarga de IDE [11]

En la figura 3.5 se muestra una imagen de la página web oficial de mikro C, desde esta página se puede proceder a descargar el IDE de desarrollo, ver actualizaciones de librerías, nuevos productos de microchip, tutoriales, etc. Esta página dado que es la oficial de MikroC cuenta con la última versión funcional del IDE.

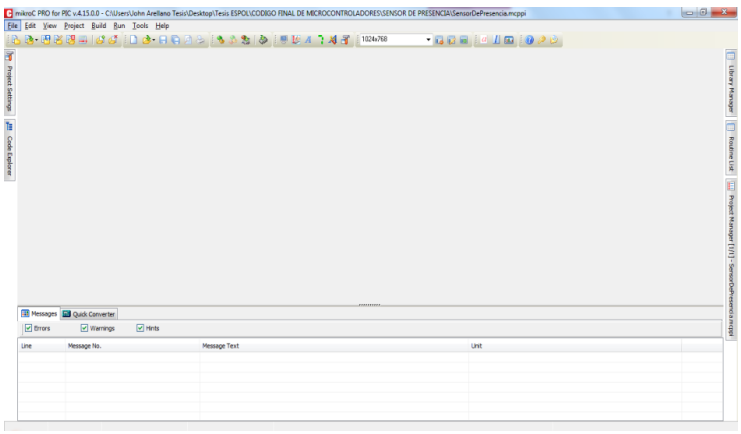


Figura 3.6 Interfaz principal del IDE MikroC

En la figura 3.6 se muestra la interfaz principal del IDE de programación de microcontroladores, MikroC, Desde aquí se puede crear nuevos proyectos, compilar, depurar, grabar el código a su familiar, configuraciones de registros del PIC, sin necesidad de código de programación.

MikroC tiene opciones extras y muy útiles, como lo son:

- Comunicación Puerto Serial.

- Test de comunicación UDP.
- Codificación de caracteres ASCII
- Exportar código a HTML.

3.3. Sensores y su papel en el sistema

En un sistema domótico el papel de los sensores es fundamental para la automatización de ciertas tareas e incluso para la seguridad, mediante la información que éstos proveen. Por lo tanto se describirá los conceptos y características de los sensores de luz, humedad y piroeléctrico, que son los usados en el presente proyecto.

3.3.1. Conceptos y características

Un sensor es un dispositivo electrónico, el cual convierte un parámetro físico, en señales eléctricas. Todos los sensores cumplen con tres características fundamentales y son:

- Diseño: Esta característica abarca la magnitud medida, que es lo detectado por el sensor (temperatura, humedad), con su respectivo rango y escala de medición. También se refiere a las características eléctricas y mecánicas del sensor.

- Prestaciones: Describen las prestaciones del sensor en condiciones ambientales normales, cuando la entrada cambia muy lentamente. También se refiere al comportamiento del sensor frente a condiciones ambientales externas, que no sean lo que mide el sensor, como humedad o presión ambiental.

- Fiabilidad: Es la capacidad para realizar su función bajo ciertas condiciones durante un tiempo establecido, esto se conoce como una probabilidad de fallo en un tiempo de uso.

3.3.2. Tipos de sensores y sus beneficios

Para los sensores que conforman los dispositivos finales, se hizo uso de módulos IDETEC. Estos módulos son usados con la finalidad de saber parámetros ambientales, con la utilidad de poder realizar alguna acción por el usuario, a algún dispositivo de la casa que tenga relación con el o los sensores usados que son:

Sensor piroeléctrico

Figura 3.7 Módulo Idetec Sensor Piroeléctrico [3]

Está hecho de un material cristalino que genera una pequeña carga eléctrica cuando es expuesto al calor en forma de radiación, cuando la cantidad de radiación es notable el cristal

cambia, la cantidad de carga también cambia y entonces puede ser medida con un sensible dispositivo FET construido dentro del sensor. En un amplio rango los elementos del sensor son sensibles a la radiación, por lo que se agrega una ventana que actúa como filtro para limitar la radiación de llegada a un rango de 8 a 14 micras donde es más sensible a la radiación del cuerpo humano. Entonces, si alguna parte del cuerpo humano cruza o pasa frente del sensor, éste se activa, pero si cruza otro tipo de elemento, no va activar al sensor.

En el presente proyecto el papel del sensor piroeléctrico es muy importante, ya que interviene en la seguridad del hogar, funcionando como alarma al detectar la presencia de intrusos, que es programada por el usuario desde la aplicación. También sirve para dar confort al usuario, ya que mientras no esté programada la alarma, automatizará el encendido o apagado de las luces del hogar.

Sensor de humedad FC-28

Figura 3.8 Módulo Idetec Sensor de Humedad del suelo [3]

Este tipo de sensor es usado en sistemas de invernadero, monitoreo de plantas, lectura de este parámetro en aplicaciones, esto debido a que mediante la salida de la tarjeta se puede conectar a un microcontrolador.

Las especificaciones de este sensor son:

- Voltaje de operación: 3.3-5V
- Comparador: LM393
- La salida de la tarjeta puede ser conectada directamente a cualquier microcontrolador

El papel del sensor de humedad en nuestro sistema domótico es informar al usuario el estado del suelo de su jardín, para que pueda tomar medidas, respecto a la información que recepte.

Sensor LDR

Figura 3.9 Módulo Idetec Sensor de Luz [3]

Un sensor LDR es una resistencia cuyo valor varía de acuerdo a la luz que recibe, cuanta más luz recibe, menor es

su resistencia. Este fenómeno se da debido a que el LDR está compuesto por un semiconductor de alta resistencia, como el sulfuro de cadmio, cuyas células se basan en la capacidad del cadmio de variar su resistencia según la cantidad de luz que incide la célula. Cuanta más luz incide, más baja es la resistencia.

El papel de este sensor en nuestro sistema domótico inalámbrico, es automatizar el encendido u apagado de luces en el exterior del hogar, con esto nos referimos al portal de la entrada principal de la casa. También permitirá la apertura o cierre de las persianas según la intensidad de luz que recepte.

3.4. Actuadores

Los módulos actuadores para los tipos de dispositivos finales tienen un papel importante en el sistema domótico planteado.

Los dispositivos finales actuadores tienen conexión directa con el sistema de alimentación de la vivienda, los dispositivos finales sensores tienen alimentación por medio de baterías.

3.4.1. Conceptos y características

Los actuadores son dispositivos mecánicos los cuales brindan la capacidad de transformar energía para generar el funcionamiento dentro de un sistema automatizado determinado.

Los actuadores según su fin de uso, consta con las siguientes características: Potencia, peso y volumen, precisión, velocidad.

3.4.2. Tipos de actuadores y beneficios

Los actuadores son aquellos dispositivos que permiten la ejecución de una acción en un dispositivo final, esto lo logran aportando la energía necesaria al sistema que modifique el valor de la magnitud física controlar. Según el tipo de señal de entrada existen tres tipos de actuadores, que se nombran a continuación: [2]

- Actuador todo/nada
- Actuador digital
- Actuador analógico

Para los dispositivos finales actuadores del sistema domótico planteado, se hizo uso de un módulo IDETEC, que cuenta las características óptimas para el funcionamiento.

Módulo Disparador de Relé, sirve para el manejo de cargas de gran Potencia. [3]

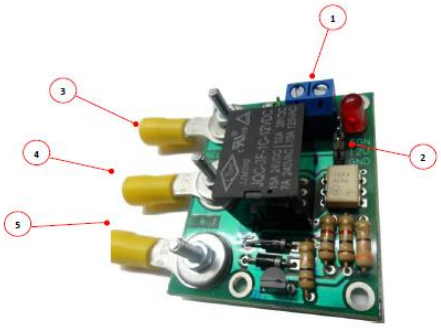


Figura 3.10 Módulo Idetec Disparador de relé [3]

El módulo mostrado en la Figura 3.10 permite el encendido o apagado de motores AC/DC y el control ON/OFF de luces, su fuente de alimentación puede ser mediante 12V de alimentación directa o 12V de alimentación por las señales de control. La carga en la salida del módulo puede ser con los siguientes parámetros: 110V/10A; 28V/10A; 24V/15A; 240V/7A. Las salidas pueden ser normalmente abierto (pin 3), normalmente cerrado (pin 4), común (pin 5).

Especificaciones

- Led indicador de activación de relé.
- Alimentación independiente para relé.
- Salidas mediante terminales de potencia.

Señales de control

- SGN: Señal TTL/CMOS para controlar la conmutación de los relés.

- EN/DS: AL seleccionar EN escogemos que la alimentación de 12V sea por el control o DS por el control.

CAPÍTULO 4

4. EL SISTEMA OPERATIVO ANDROIDE Y SU PAPEL IMPORTANTE EN LOS SISTEMAS DOMÓTICOS

Este capítulo detalla a breves rasgos la historia del sistema operativo androide, sus principales características, beneficios y las diferentes versiones que han surgido.

4.1. Fundamentos teóricos del sistema operativo Androide

El sistema operativo androide es una plataforma de software con un conjunto de herramientas y aplicaciones vinculadas a una distribución de Linux. De entre sus principales ventajas se tiene que su código es abierto, esto le permite ser adaptado a diferentes dispositivos como tablets, teléfonos inteligentes o Smartphone, relojes inteligentes, televisores y automóviles; también en caso de

error se puede agilizar correcciones, ya que los usuarios pueden indagar en el código sin limitación alguna, esto a la vez fomenta la retroalimentación en una comunidad de desarrolladores reconocida por su activa participación en concursos, competiciones, eventos y reuniones. Las desventajas de este sistema operativo tienen que ver con el hecho de que permite tener varias aplicaciones abiertas a la vez, lo que genera que se dispare el consumo de la batería, además por su “filosofía aperturista” está más propenso a ataques y se sabe que la mayoría de software malicioso está destinado al sistema operativo Androide. [4]

4.1.1. Historia

El sistema operativo Androide, Inicialmente fue creado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. Sin embargo la historia de este sistema operativo ha ido evolucionando desde hace seis años. [5]

Año de lanzamiento	Versión de S.O.	Características
2008	Android 1.0 Apple Pie	Tenía aplicaciones muy básicas como navegador web, Gmail, Google Maps, Talk.
2009	Android 2.0 Eclair	Compatibilidad con fondos de pantalla animados, compatibilidad con Bluetooth 2.1, mejoras en la aplicación de cámara.
2010	Android 2.2 Froyo	Compatibilidad con Adobe Flash 10.1, una Radio FM, la opción de crear una zona WiFi.
2011	Android 3.0 Honeycomb	Diseño para tabletas, cambio del tamaño de los widgets, mejoras en la barra de notificaciones,

		videollamadas a través de Google Talk.
2012	Android 4.1, Android 4.2, Android 4.3 Jelly Bean	Dieron vida a teléfonos inteligentes como el Samsung Galaxy S, S2, el Nexus 4, el Huawei Ascend Y, primeras tabletas Samsung Galaxy Tab 10.1.
2013	Android 4.4 KitKat	Importantes mejoras en la interfaz, nuevo teclado de emoticones, nuevas opciones de Hangouts y significativas mejoras de rendimiento.
2014	Android 5.0 Lollipop	Promete diseños y prestaciones espectaculares, la época en la que incluso los accesorios de pulsera

		comienzan a introducirse en el mercado de la telefonía móvil.

Tabla 4.1 Historia de versiones de Android [5]

4.1.2. Características

Estas características describen mejor el sistema operativo Androide y se listan a continuación:

- Código abierto.
- Núcleo basado en el Kernel de Linux.
- Soporte de Java y muchos formatos multimedia.
- Soporte de HTML, HTML5, Adobe Flash Player, etc.
- Adaptable a muchas pantallas y resoluciones.
- Utiliza SQLite para el almacenamiento de datos.
- Ofrece diferentes formas de mensajería.
- Navegador web basado en Web Kit incluido.
- Bluetooth.

- Google Talk desde su versión HoneyComb, para realizar videollamadas.
- Multitarea real de aplicaciones.

4.1.3. Beneficios

Existen varios beneficios que Android ofrece, por lo que es uno de los sistemas operativos más usado dentro del mercado de la telefonía móvil, esto gracias a su compatibilidad con la mayoría de hardware disponible para los Smartphone.

Los beneficios más destacados de este sistema operativo son:

- Es que es un lenguaje de código abierto, esto permite la creación de aplicaciones por cualquier persona, lo que permite retroalimentación de conocimientos entres los desarrolladores.

- Es un sistema operativo para Smartphone muy usado, ya que es compatible con marcas como Samsung y LG.

- En la actualidad existen más de 100.000 aplicaciones disponibles para Android y gran parte de ellas son gratuitas.

- Permite el funcionamiento simultáneo de varias aplicaciones, lo que se conoce como multitarea.

4.1.4. Versiones

El sistema operativo Androide consta actualmente con 18 versiones, que va desde la 1.0, que corresponde al API 1, creada en septiembre del 2008, hasta la última versión lanzada noviembre del 2014, que corresponde a la versión 5.0. Sin embargo las características de las versiones más importantes han sido detalladas en la Tabla 4.1.

4.2. Lenguaje de programación JAVA

Es un lenguaje de programación orientada a objetos que fue desarrollado por la compañía Sun Microsystems, cuyo objetivo original era la creación de páginas WEB. La principal característica de Java es que es un lenguaje independiente de la plataforma, también se pueden hacer rutinas individuales que sean usadas por más de una aplicación, lo que se conoce como modularidad. Permite tanto el desarrollo de aplicaciones bajo el esquema cliente-servidor, como de aplicaciones distribuidas.

4.2.1. Historia

Nace en 1991 con el nombre "OAK", luego por problemas legales, fue cambiado a JAVA, dicho cambio se dio por Green. JAVA nació con el objetivo de crear un lenguaje de programación parecido a C++ en estructura y sintaxis, fuertemente orientado a objetos, pero con una máquina virtual propia. Esto se hizo bajo el principio, de poder ser usado bajo

cualquier arquitectura "Write Once, Run Anywhere (escríbelo una vez, ejecútalo en cualquier sitio)". [6]

En 1992 se presenta el proyecto verde, con los prototipos a bajo nivel. Entre 1993 y 1994 se trabaja para poder presentar un prototipo funcional (hotJava) donde se ve todo el potencial que JAVA puede ofrecer. Finalmente en 1995, es presentada la versión alpha, y un 1 año después en 1996 es lanzado el primer JDK (JDK 1.0). El desarrollo de java a partir de entonces es imparable, se van presentando nuevos paquetes y librerías hasta la actualidad.

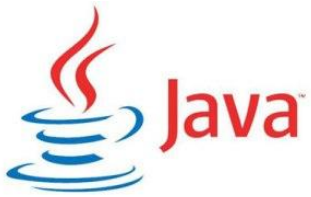


Figura 4.1 Logotipo de JAVA [12]

4.2.2. Beneficios de programar en lenguaje Java

Uno de los beneficios más importantes de JAVA es que es se puede crear un programa multiplataforma, esto quiere decir que un programa desarrollado en java puede ser ejecutado en cualquier sistema operativo de escritorio. Al ser un lenguaje libre, no se debe pagar ningún tipo de licencia por su uso. Java no necesita la instalación frecuente de plugins para poder programar correctamente, por otra parte es u lenguaje que se adapta a la perfección a todo tipo de dispositivos móviles, lo que permite tener acceso desde cualquier lugar y en cualquier momento a un desarrollo en Java.

4.2.3. Conceptos de la programación orientada a objetos

La programación orientada a objetos es una serie de reglas y normas para realizar las cosas, tal que otras personas puedan re utilizar el código. Se diferencia de otros tipos de programación por expresar las cosas de manera más cercana

a la vida real. Las clases son las que definen un objeto, cuando se definen las características y funciones, en realidad se está programando una clase con sus propiedades o atributos, y sus métodos o funcionalidades de los objetos.

Entre las principales características que tiene la programación orientada a objetos son abstracción, encapsulado, modularidad, jerarquía, herencia, paso de mensajes y polimorfismo.

4.2.4. Clases y métodos importantes para la comunicación a través de la red

Esta sección se refiere a las clases que se programaron en el presente proyecto para lograr la comunicación WiFi a través de la red del sistema, en este caso desde la aplicación “Control Home” hacia el concentrador, las clases con sus respectivos atributos y funciones se detallan en el Anexo B.

4.2.5. ADT para desarrollo de aplicaciones Androide

El ADT es el ambiente en que se desarrollan las aplicaciones Android, el cual al ser descargado desde la página oficial de google, incluye:

- Eclipse como entorno de desarrollo.
- Herramientas SDK de Android.
- La última plataforma de Android.
- L última imagen de Android para el emulador.

Figura 4.2 Página oficial del IDE de desarrollo eclipse [13]

En la figura 4.2 se muestra la página oficial del IDE de desarrollo eclipse, desde esta página se puede acceder a

opciones de descarga de las distintas versiones que trae eclipse, así como foros de preguntas, plugins y noticias.

4.3. Características de las aplicaciones Androide

Existen características que diferencian y hacen de Android un mejor sistema operativo para dispositivos móviles. Entre estas características se encuentras sus teclados móviles, automatización, launchers personalizados, widgets, batería intercambiable, memoria de almacenamiento externo, instalación de aplicaciones desde el PC, ROMs personalizadas, control del móvil desde e PC, adobe flash, integración real de servicios y aplicaciones.

4.3.1. Sistema de capas de Android

El sistema de capas nos indica cómo está estructurado el sistema operativo Android, que en este caso las capas permiten facilitar al desarrollador la creación de aplicaciones.

La arquitectura de Android es conocida como pila, debido a que cada capa utiliza elementos de la capa inferior, y se puede acceder a ellos mediante librerías, evitando así que el desarrollador programe a bajo nivel.

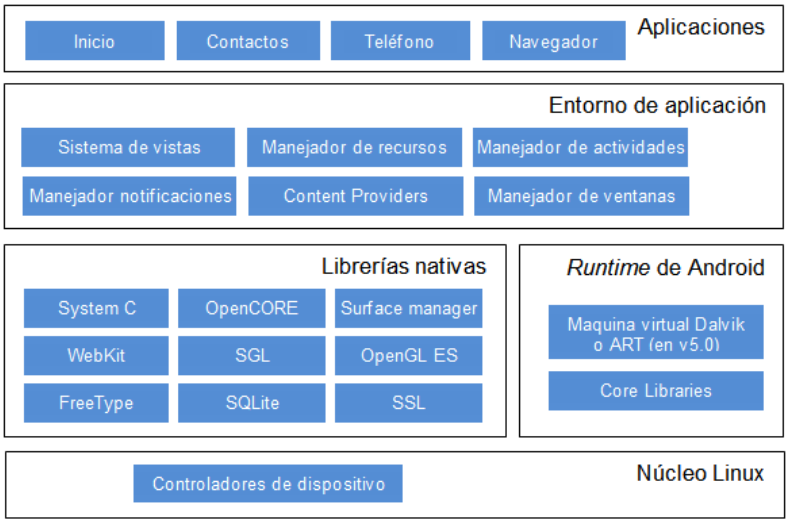


Figura 4.3 Sistemas de capas de Android [14]

En la figura 4.3, se describe las capas o arquitectura en la que está formado el sistema operativo Android. Un punto importante acerca de cada una de las capas, es que todas están basadas en software libre.

4.3.2. Componentes básicos de una aplicación Androide

Los componentes básicos, por lo que está formada una aplicación Android, en particular la que se desarrolló para el proyecto planteado, se refiere a lo mínimo que necesita la interfaz para ser amigable y entendible al usuario, estos componentes son:

- Vistas (Views): Son los elementos que conforman la interfaz de usuario de una aplicación, como por ejemplo un botón, un texto, etc.

- Layout: Conjunto de vistas agrupadas, de una determinada forma.

- Activity: Es la clase que representa la pantalla o interfaz de usuario.

- Servicio (Service): Proceso que se ejecuta en background, sin necesidad de iteración con el usuario.

- Intent: Voluntad de realizar alguna acción.

- Fragment: Unión de varias vistas, para crear un bloque funcional.

- Broadcast receiver: Receptor de anuncios, por parte de la o las aplicaciones, que tengan relación con este.

4.3.3. Explicación detallada de carpetas y archivos creaos en algún proyecto, en el entorno de software eclipse

Eclipse genera una serie de carpetas cuando se crea un proyecto de aplicación Android, es importante para que sirve y que contiene cada una de ellas, al momento de colocar o quitar elementos a la aplicación. Estas carpetas se detallan a continuación.

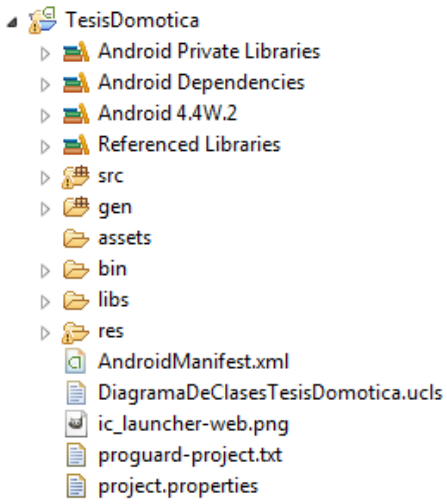


Figura 4.4 Carpetas de proyecto Android.

- Carpeta src: Contiene todas las clases necesarias para poder compilar la aplicación, y por ende el correcto funcionamiento de la misma.

- Carpeta gen: Aquí se encuentra código generado automáticamente, por la aplicación, estos códigos no deben ser manipulados por desarrollador.

- Carpeta bin: Contiene el instalador de la aplicación, o también conocido como apk.

- Carpeta libs: Contiene las librerías necesarias para la correcta compilación de la aplicación.

- Carpeta res: Se ubican los archivos necesarios para la interfaz y experiencia del usuario, estos son archivos XML, imágenes png, archivos de audio, etc.

CAPÍTULO 5

5. PRINCIPIOS PARA EL DISEÑO DE LOS SISTEMAS DOMÓTICOS TIPO CENTRALIZADO

En el presente capítulo se detalla las principales características por las que un sistema domótico centralizado conlleva ventajas para el usuario, al momento de adquirir este tipo de sistema y en qué circunstancias debería ser implementado.

5.1. Características generales

En un sistema domótico centralizado, la primera característica que resalta es que todos los elementos del sistema concentrador, sensores y actuadores residen en un mismo espacio físico, donde

están la mayoría de interconexiones entre los módulos. De manera lógica se tiene que ningún módulo de dispositivo final, actuador o sensor, podrá establecer comunicación entre sí a menos que tenga como intermediario al concentrador del sistema domótico centralizado.

5.1.1. Integración

Esta característica se refiere a tener integrados en un mismo equipo físico, el controlador, los sensores y actuadores necesarios para el control del hogar. Este equipo integrado evita grandes extensiones de cableado en el hogar, por lo que resulta ideal para instalaciones pequeñas, con funciones muy limitadas de control.

5.1.2. Interrelación

Esta característica se refiere a la capacidad de interrelacionar los distintos componentes del sistema, es decir que ayude a tomar decisiones acerca del control del hogar. Por ejemplo que la información receptada por un sensor temperatura ayude a decidir el encendido o apagado de sistemas de ventilación.

5.1.3. Facilidad de uso

Esta característica es importante para el usuario y se debe a que es fácil acceder a cualquier dispositivo final para controlar desde un mismo panel y con la menor cantidad de pasos para accionamiento posible, es decir no más que la pulsación de unas pocas teclas.

5.1.4 Actualización

Debe ser fácil de actualizar, a medida que transcurre el tiempo se implementan mejoras, ya sea para agregar nuevos elementos, o mejorar procesos de acciones.

5.1.5. Fiabilidad

Esta característica es importante, porque hace referencia a la disponibilidad del sistema, y las medidas de contingencia que tiene para mantenerse con la menor cantidad de fallas posibles. Es decir que cuente con una fuente de alimentación alterna en caso de fallas, para que los principales subsistemas se mantengan activos.

5.1.6. Control remoto

Es muy útil que el usuario, vía internet tenga accesos al sistema domótico de su hogar, y pueda ver los estados de los

elementos de su hogar o aún mejor inclusive pueda ejercer control sobre ellos.

5.2. Los sistemas de bases de datos en los sistemas domóticos

El un sistema domótico es importante el uso de base de datos, para poder almacenar de manera ordenada los eventos que se llevan a cabo en el sistema domótico propuesto. De esta manera se puede obtener estadísticas de consumo energético, frecuencia de uso por parte de los miembros del hogar. El sistema de base de datos usado en el presente proyecto es MySQL.

5.2.1. Historia

Una base de datos es un conjunto de información relacionada que se encuentra estructurada o agrupada, este término fue escuchado por primera vez en un simposio celebrado en California en 1963.

Los orígenes de las bases de datos se remontan a la antigüedad donde se llevaban toda clase de registros, como información sobre cosechas y censos. Sin embargo en esa época la búsqueda era lenta y poco eficaz, ya que no se contaba con la ayuda de máquinas que reemplazaran el trabajo manual. Desde la aparición de las primeras computadoras, se desarrolló el concepto de las bases de datos como la solución a las necesidades de almacenamientos de grandes cantidades de información, y desde entonces se ha ligado a las bases de datos con la informática. En la década de los 60 se dio inicio a las primeras generaciones de bases de datos de red y las bases de datos jerárquicas, ya que fue posible guardar estructuras de datos y árboles. En esta década se obtuvieron logros como SABRE e IDS, que eran sistemas para manejo de reservas de vuelos y un nuevo tipo de sistemas de bases de datos, respectivamente. [7]

5.2.2. Conceptos y tipos

Una base de datos es una herramienta para recopilar y organizar información, que es una colección de archivos relacionados, donde se dice que estos archivos son una colección de registros, y que estos registros son una colección de campos. Existen sistemas manejadores de bases de datos que se usan ampliamente para: organizar y manipular grandes volúmenes de datos de las empresas. Según su estructura interna, que es la manera en que organizan la información, existen cinco tipos de bases de datos que se describen a continuación.

1) Bases de datos jerárquicas: Este tipo de base de datos utiliza estructuras arborescentes (árbol), en las que existen elementos denominados nodos, y entre ellos tienen dependencias. Las dependencias se manejan con una relación de 1:M, en la que un hijo no puede tener más de un padre, pero un padre puede tener varios hijos.

2) Bases de datos en red: Este tipo utiliza la estructura de grafo/red, como en el caso de las bases de datos jerárquicas, se manejan relaciones de 1:M, pero en este caso un objeto puede relacionarse como hijo con varios elementos padres.

3) Bases de datos relacionadas: Para aplicaciones de gestión, esta es la estructura que se ha impuesto. Esta estructura organiza los datos a maneras de tablas, donde las relaciones entre tablas hijos y padres se consigue incluyendo en la tabla hijo la clave del objeto padre.

4) Bases de datos orientadas a objetos: Esta base de datos trata de almacenar objetos completos, lo que incluye estado y comportamiento. La información se organiza en atributos y los comportamientos en operaciones.

5) Bases de datos multidimensionales: En este tipo de estructura los datos son almacenados en tablas de múltiple dimensiones. Son usadas para almacenamiento de grandes volúmenes de información.

5.2.3. Características y ventajas

Las principales características de los sistemas de bases de datos tienen son, su velocidad respecto a la consulta de datos; los datos pueden ser usados por cualquier aplicación, debido a que son independientes; se reduce la redundancia, identificada como el hecho de duplicar datos; genera menos inconsistencias, que son datos con contradicciones.

Las principales ventajas de los sistemas de bases de datos son la seguridad debido a que el acceso de usuarios es

controlado; la información se recolecta y almacenas una sola vez, por lo tanto los datos son coherentes; reduce el espacio de almacenamiento, debido a la mejor estructuración de los datos; el procedimiento de búsqueda es más rápido y flexible, debido a que sólo necesita escribir breves oraciones.

5.3. Tecnología WiFi como medio principal de comunicación entre software y hardware del sistema domótico

En el sistema domótico planteado la tecnología WiFi es usada como puente de comunicación entre la interfaz de usuario, aplicación androide “Control Home”, y el concentrador. Su papel es importante debido a que permite que las órdenes que el usuario envía desde la aplicación sean receptadas, procesadas y reenviadas hacia los módulos actuadores, por el concentrador.

5.3.1. Historia de la tecnología WiFi

El origen y desarrollo de esta tecnología se remonta al año 1880, cuando se inventó el fotófono, un aparato de transmisión de sonido mediante un haz de luz. Luego de ocho años de este evento, el físico alemán Rudolf Hertz utilizó ondas de radio para establecer la primera comunicación inalámbrica. La primera WLAN, bautizada con el nombre de ALOHAnet fue diseñada en 1971 por un grupo de investigadores americanos, quienes establecieron comunicación entre ordenadores ubicados en las distintas islas de Hawai, mediante ondas de radio. En 1991 se desarrollaron las bases del estándar 802.11, que establece la normativa en la comunicación inalámbrica, hasta 1993 las velocidades de transmisión eran realmente bajas del orden de los 5Mbps. En 1997, por parte del IEEE se lanza el estándar 802.11. Posteriormente en el año 1999 compañías como Lucent, Nokia o SymbolTechnologies, se reunieron para crear una asociación conocida como WECA (Wireless Etlityhernet Compatibility), cuya finalidad era establecer estándares para que los equipos dotados de esta tecnología

inalámbrica sean compatibles entre sí. En 2003 esta asociación se rebautizó como WiFi Alliance. [8]

5.3.2. Conceptos y características

WiFi es una tecnología de comunicación inalámbrica mediante ondas, conocida también como WLAN o estándar IEEE 802.11. WiFi, por su nombre, es confundido con Wireless Fidelity, sin embargo WiFi sólo es un nombre comercial, utilizado con la finalidad de ser fácil de recordar. La estructura básica de una red WiFi, posee los siguientes elementos, con sus principales características:

- Punto de acceso: Este elemento permite la comunicación entre todos los elementos de la red con el router. En lugares abiertos cada punto de acceso tiene un alcance de hasta 210 metros, sin embargo en entornos cerrados este alcance se reduce a 90 metros.

- Tarjeta de Red Wireless: Es el que permite al usuario conectarse a su punto de acceso más próximo.

- Router: Permite la conexión de un punto de acceso a internet.

WiFi utiliza los estándares 802.11a, 802.11b, 802.11g y 802.11n.

Tecnología	Velocidad de transmisión	Características
802.11b	11 Mbps	Trabaja en la banda de frecuencia de 2.4GHz y es compatible velocidades menores
802.11g	11/24/54 Mbps	Trabaja en la banda de frecuencia de 2.4GHz

802.11n	300 Mbps	Frecuencia 2.4GHz y 5GHz simultáneamente, ya que utiliza tecnología MIMO

Tabla 5.1 Estándares WiFi

5.3.3. Ventajas ante el medio alámbrico

Frente al medio alámbrico WiFi tiene varias indiscutibles ventas, que se detallan a continuación. La ausencia de cables es su ventaja principal, puede comunicar dispositivos con diversas características sin tener que buscar un cable adecuado para cada enlace. Es ideal para implementarse en lugares considerados monumentos históricos, ya que sería inaceptable realizar el cableado necesario para acceder a internet.

Brinda comodidad al usuario, ya que sólo necesita estar lo suficientemente cerca de un punto de acceso y tener un dispositivo con tecnología WiFi, para proceder con el acceso a la red. Genera menor gasto en infraestructura, esto respecto a agregar equipos una vez que se estableció una red inicial, esto no es así en la tecnología por cable.

5.3.4. Tipos de equipos usados

En esta sección se mencionan los equipos usados en el presente proyecto, para lograr la completa comunicación inalámbrica entre los elementos que conforman el sistema domótico planteado. Para lo cual se usan dos estándares de comunicación inalámbrica, detallados a continuación.

- Comunicación entre Concentrador y Aplicación: En este tramo de comunicación se usó la tecnología WiFi, donde el concentrador está conectado vía Ethernet con el router, el cual envía los datagramas UDP a la

aplicación mediante tecnología WiFi. Esta conexión se puede ver en el Anexo N.

- Comunicación entre concentrador y módulos de dispositivos finales: En este tramo de comunicación se usó módulos inalámbricos XBee, de los que se puede informar en el Anexo C, con protocolo de comunicación 802.15.4, que al igual que el protocolo 802.11 trabaja en la banda de 2.4 GHz. Esta conexión se puede ver en el Anexo N.

CAPÍTULO 6

6. DISEÑO E IMPLEMENTACIÓN GENERAL DEL SISTEMA DOMÓTICO

En el presente capítulo se detalla el diseño, programación e implementación de los diferentes elementos que conforman el sistema domótico planteado.

6.1. Descripciones del sistema domótico, basado en un modelo centralizado

El sistema domótico planteado es centralizado a nivel lógico, en ciertos escenarios de comunicación, donde todos los elementos del sistema que desean establecer comunicación pasan por la intervención del concentrador, por tanto se explicará más a detalle

este tipo de sistema domótico. Además se revisarán los esquemas de comunicación que existen entre el concentrador con el dispositivo final y la aplicación androide.

6.1.1. Generalidades

En este tipo de sistema el controlador es el principal, quien se encarga de la gestión, proceso y acciones de los elementos del sistema, en el que se incluyen interfaces, actuadores, sensores, etc. El sistema puede ser centralizado a nivel lógico, la manera en cómo se comunican los elementos, o a nivel físico, la manera en cómo están ubicados los elementos del sistema.

6.1.2. Conceptos asociados al software a usar para programación del hardware

Dado que no se ocupa completamente la memoria en los PIC 18F4520 y 16F886, los cuales son usados para el concentrador y dispositivos finales, respectivamente, el IDE

de desarrollo nos permite compilar los códigos sin ningún problema, esto es una ventaja, dado que no se necesita adquirir la versión pagada y completa de MikroC, para poder compilar los respectivos programas, para los microcontroladores.

6.1.3. Conceptos asociados al software a usar para programación de la aplicación.

Dado que la aplicación se programa en el lenguaje nativo de Android, que es JAVA, y como se describe en capítulos anteriores, JAVA es un lenguaje libre y podemos usarlo, sin tener que pagar ninguna licencia, esto es ventajoso dado que eclipse también es un IDE gratuito, por ende para el desarrollo de la aplicación no se tiene ningún tipo de gasto.

6.1.4. Diagrama funcional del sistema

En el Anexo D se muestra un diagrama completo del sistema domótico, donde se nota el tipo de comunicación que existe entre cada parte del sistema.

6.1.5. Establecimiento de las tramas de comunicación

En esta sección se explica el establecimiento de las tramas para cada uno de los escenarios de comunicación posibles en el sistema. Se detallan los pasos que conlleva agregar y usar un nuevo dispositivo actuador o sensor.

Petición de nuevo elemento a ser agregado

Este escenario ocurre únicamente cuando un elemento del hogar es energizado, conectado a su respectivo módulo “end device” y con un valor de “FF” en el registro 0x00 de la memoria EEPROM, si estas condiciones se cumple se procede a enviar una sola vez una trama única por elemento o dispositivo, al concentrador, por puerto serial. Esta

comunicación se lleva a cabo de manera inalámbrica gracias a los módulos XBee.

Figura 6.1 Trama enviada por dispositivos finales

La Figura 6.2 muestra los campos de la trama que se envía por parte de los dispositivos finales, para poder ser controlados en el hogar, desde la App.

El concentrador será el encargado de verificar que la trama que le llega por puerto serial, sea de los dispositivos finales, siendo esto correcto, procede a crear un datagrama UDP, que tiene los siguientes parámetros:

- PUERTO DESTINO (App): 11111
- PUERTO LOCAL (Concentrador): 5505
- DIRECCIÓN IP DE DESTINO: Dirección de red broadcast de la red local.
- Dato a enviar: Trama enviada por los dispositivos finales, que se recibió por el puerto serial.

Una vez creado el datagrama UDP, el concentrador procede enviar dicho datagrama por la red Ethernet, gracias al módulo Ethernet ENC28J60, a manera de broadcast, esto con el fin de que en todos los terminales móviles donde se encuentre instalada la aplicación, puedan escuchar y detectar el nuevo elemento que se ha conectado al hogar, procediendo a elegir el ambiente donde se desea agregar el dispositivo del hogar conectado. Una vez finalizado el proceso de agregación del elemento a la aplicación, esta misma procede a crear un datagrama UDP, con los siguientes campos:

- PUERTO LOCAL (App): 5506
- PUERTO DESTINO (Concentrador): 5505
- IP DESTINO (Concentrador): 192.168.0.60
- IP LOCAL: IP propia del terminal móvil.
- DATO: Trama de elemento agregado correctamente.

Finalmente la aplicación envía este datagrama al concentrador y él lo remite por puerto serial al dispositivo final que solicito ser agregado. El dispositivo final al recibir la trama de confirmación, procede a cambiar el registro 0x00 de la memoria EEPROM a un valor de 1, caso contrario el enviara la trama de ser agregado a la App, cada vez que se energice el mismo dispositivo final.

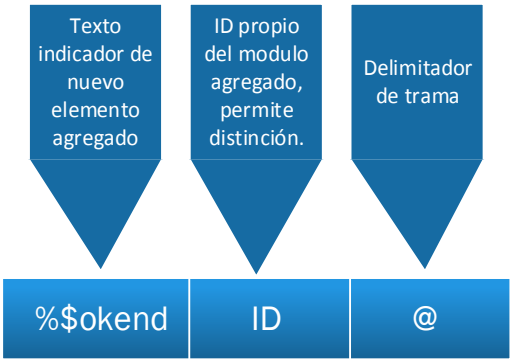


Figura 6.2 Trama de confirmación enviada por la aplicación

En la Figura 6.2 se muestra la trama de confirmación que recibe el dispositivo final por parte de la aplicación. En esta trama va el ID destinatario. En el Anexo E se muestra el ejemplo de agregar un dispositivo final como una lámpara, este funcionamiento se muestra a manera de un diagrama de secuencia funcional.

Acción en elemento del hogar

Este escenario es iniciado por la aplicación, en la que, estando en un ambiente, se escoge el dispositivo final, al cual

se desee realizar alguna acción, pudiendo ser: encender, apagar o dimerizar. La aplicación Androide forma una trama que contiene la acción a realizar, el ID correspondiente al dispositivo sobre el cuál se deberá actuar un delimitador de trama, que se envía en un datagrama al concentrador.

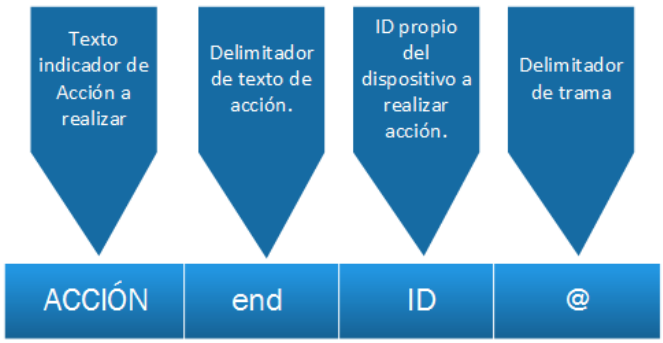


Figura 6.3 Trama de acción

Los cuatro primeros bytes de la trama indican el tipo de acción que se va a realizar, donde se pueden tener las siguientes combinaciones:

- “1000” → Indica una acción de activación al dispositivo final que lo reciba.

- “0000” → Indica una acción de desactivación al dispositivo final que lo reciba.

- “2000” → Indica una acción de dimerización al dispositivo final que lo reciba, donde los últimos tres bytes, indicarán un valor que se encuentra entre 000 y 100.

La aplicación, procede a crear un datagrama UDP, con los siguientes campos:

- PUERTO LOCAL (App): 5506
- PUERTO DESTINO (Concentrador): 5505
- IP DESTINO (Concentrador): 192.168.0.60
- IP LOCAL: IP propia del terminal móvil.
- DATO: Trama de acción de elemento.

Una vez que el datagrama es receptado por el concentrador, este lo retransmite por puerto serial a todos los XBee que conforman la red. La acción es ejecutada por el dispositivo que tenga coincidencia entre el ID enviado en la trama y el ID propio del dispositivo. En el Anexo E se puede ver un ejemplo del proceso de ejecución de acción de un foco.

Solicitud de lectura de sensor

Esta solicitud es realizada por la aplicación, que de acuerdo al ambiente en que se encuentre ubicado el usuario dentro de la aplicación, puede escoger diferentes lecturas de los sensores, para decidir adecuadamente que acciones desea realizar. Para lo cual envía una trama de petición de acción, con información de la acción a realizar, en este caso de lectura, que es procesada por el módulo actuador de sensores. La trama usada para esta acción tiene la misma estructura mostrada en la Figura 6.3.

La aplicación, procede a crear un datagrama UDP, con los siguientes campos:

- PUERTO LOCAL (App): 5506
- PUERTO DESTINO (Concentrador): 5505
- IP DESTINO (Concentrador): 192.168.0.60
- IP LOCAL: IP propia del terminal móvil.
- DATO: Trama de consulta lectura de sensor.

Una vez que el datagrama es receptado por el concentrador, este lo retransmite por serial a todos los XBee que conforman la red. El módulo del sensor que tenga coincidencia entre su ID propio y el ID recibido en la trama, procesa la petición de lectura y envía mediante comunicación serial una trama que contiene el tipo de sensor que responde la petición, la lectura del mismo, el ID del sensor, y el delimitador de trama.

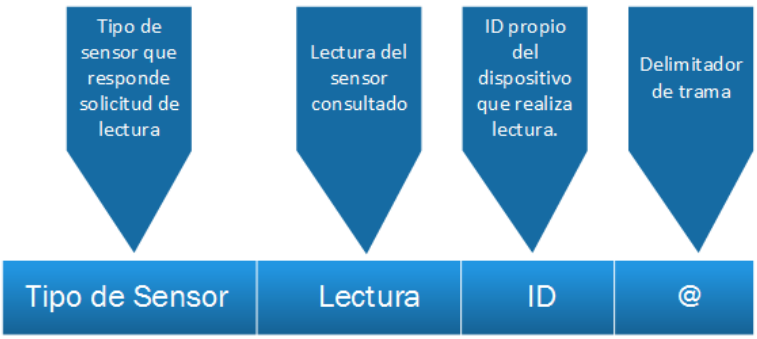


Figura 6.4 Trama de respuesta de sensor.

En la figura 6.4 se muestra los parámetros con que se estructura la trama de respuesta de lectura, que envía el sensor consultado al concentrador. Los dos primeros bytes de la trama corresponden al tipo de sensor, y pueden ser:

- "SP" → Sensor de presencia.
- "SH" → Sensor de humedad.
- "ST" → Sensor de temperatura.

Mediante comunicación serial el concentrador recibe la trama enviada por el módulo del sensor, luego de verificar que la

trama que recibe por puerto serial, sea de los dispositivos finales, procede a crear un datagrama UDP, que tiene los siguientes parámetros:

- PUERTO DESTINO (App): 11111
- PUERTO LOCAL (Concentrador): 5505
- DIRECCIÓN IP DE DESTINO: Dirección de red broadcast de la red local.
- Dato a enviar: Trama enviada por los dispositivos finales, que se recibió por el puerto serial.

En el Anexo E se muestra el ejemplo de petición de lectura a un sensor de temperatura, este funcionamiento se muestra a manera de un diagrama de secuencia funcional.

6.2. Diseño y programación de hardware Idetec

Esta sección detalla el hardware Idetec usado para conformar el módulo concentrador, los módulos actuadores y los módulos sensores del proyecto planteado.

Módulo concentrador

Para el módulo concentrador se procedió con el diseño de un circuito con todos los elementos que lo conforman, esto incluye un PIC16F4520, módulo Ethernet ENC28J60, módulo XBee, un cristal de 10 MHz y un LED indicando su funcionamiento, esto se lo realizó en Proteus.

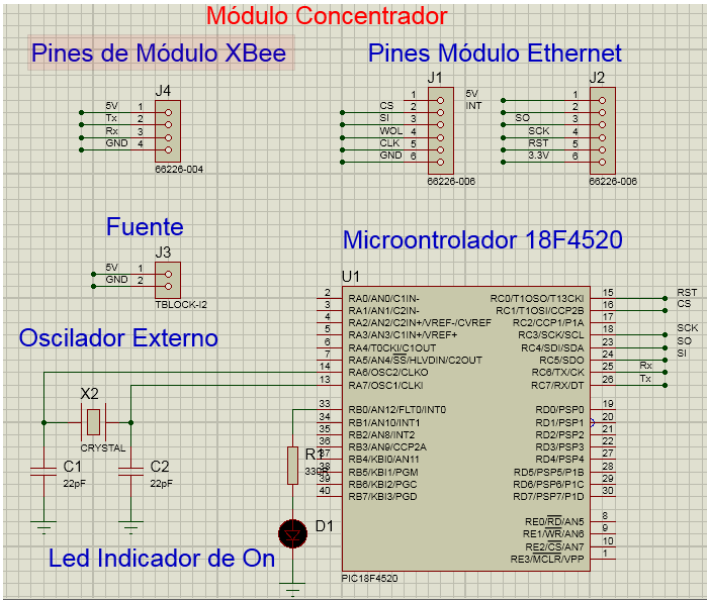


Figura 6.5 Diagrama esquemático del módulo concentrador

Para el diseño del circuito impreso se usó la herramienta ARES, la cual viene integrada en Proteus. Para esto se trató de que todos los elementos ocupen el menos espacio posible, con la finalidad de que el tamaño del concentrador no sea exageradamente grande. En la Figura 6.6 se muestra una imagen del diseño PCB del módulo concentrador, dicho diseño fue creado en ARES.

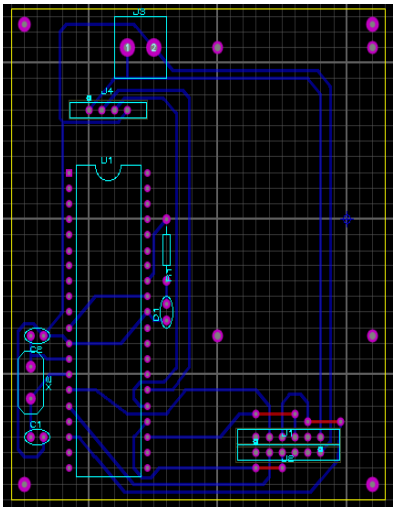


Figura 6.6 Diseño de circuito impreso del módulo concentrador

Proteus permite también tener una noción de cómo se verá el circuito una vez impreso y soldados los elementos que lo conforman, esto ayuda a saber si el diseño elaborado es el adecuado, y no

impedirá a algún elemento ser soldado adecuadamente, tal como se puede visualizar en la Figura 6.7.

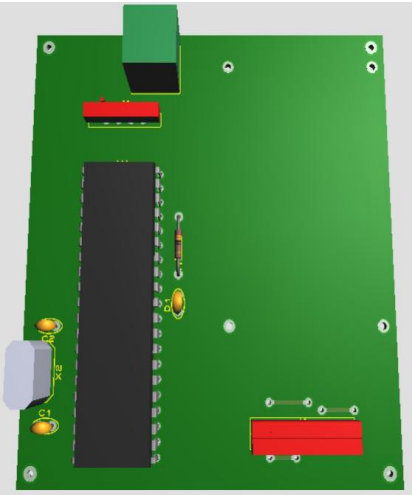


Figura 6.7 Vista 3D de diseño del módulo concentrador

El módulo concentrador es el módulo principal encargado de redirigir los datagramas UDP que envía la App Android, con las acciones que se desee realizar dentro del hogar hacía los dispositivos finales, por el puerto serial y de manera inalámbrica gracias a los módulos XBee. Así mismo el módulo concentrador está escuchando constantemente el puerto serial, donde recibe información a través de los módulos inalámbricos XBee, por parte de los dispositivos finales, pudiendo ser información de los actuadores o sensores; esta

información es convertida en un datagrama UDP y es enviado por la red WiFi a la aplicación Android para su procesamiento e información para el usuario. El código de funcionamiento del concentrador se describe en el Anexo A. También en el Anexo F se muestra en un diagrama de flujo, los procedimientos llevados a cabo por el concentrador.

Módulos de dispositivos finales

Los módulos dispositivos finales son los módulos que van a interactuar con el usuario, es decir módulos actuadores que servirán para encender, apagar o dimerizar la iluminaria del hogar, o ventilación, etc, para los cuales se muestra un diagrama de flujo de los procesos que realizan en el Anexo F. De igual manera estos módulos pueden ser sensores, que le servirán al usuario para saber el estado de la temperatura, la humedad, detectar presencia y más funciones, dependiendo el que se conecte y del lugar de interés donde el usuario conecte el módulo.

Algunos actuadores tienen la funcionalidad de poder dimerizar la carga, como en el caso de los actuadores para los focos o lámparas. Para poder dimerizar la carga, se hace uso de la señal de PWM del PIC16F886 que está integrado en los módulos de dispositivos finales. Para ajustar el brillo, se usa la siguiente ecuación:

$$\boldsymbol{duty\ rate} = (\boldsymbol{porcentaje} * \mathbf{255})/\mathbf{100}$$

6.1 Ecuación para ajuste de brillo

En la Ecuación 6.1 “duty rate” es el valor que se le asignara al PWM del PIC, y “porcentaje” es el valor de nivel de brillo que se desea tener en una lámpara o foco. Dado esto la aplicación enviara un valor de 0 a 100, según el brillo que desea establecer en su lámpara o foco, el usuario tiene tres opciones de intensidad que son baja, media o alta. En el Anexo E mediante un diagrama se muestra el proceso para cambiar la intensidad de luz de un foco.

El sistema domótico diseñado cuenta con un sensor de humedad, presencia, temperatura. Cuando la aplicación solicita leer el valor de uno de estos sensores, ella transmite el datagrama UDP, con el ID destinatario, al concentrador, y él se encarga de solicitar la lectura del parámetro físico al sensor, para luego dicho valor retransmitirlo a la aplicación. Este proceso se muestra mediante un diagrama de flujo en el Anexo F.

6.2.1. Sección de ajustes manuales en casa

El usuario podrá ajustar manualmente la activación o desactivación de dispositivos en su hogar, desde la aplicación, usando switch, radio button o botones, para la respectiva acción que desee el usuario.

6.2.2. Sección de ajustes automáticos mediante sensores

El usuario podrá realizar acciones automáticas en su hogar, involucrando a los dispositivos finales, con módulos sensores, desde la aplicación Android. Por ejemplo se podrá

activar automáticamente una regadera en el jardín, mediante el sensor de humedad, dicha acción se puede realizar independiente del usuario, o cuando el mismo disponga realizar la acción desde la aplicación.

6.2.3. Sección de programación de ajustes Ethernet, para el concentrador.

El módulo concentrador está conectado a la red LAN del usuario, gracias a un módulo de red, llamado: “Modulo Ethernet ENC28J60”. Este módulo permite al microcontrolador 18F4520 conectarse a internet, se basa en el chip de microchip ENC28J60 y cuenta con toda la electrónica necesaria para poder conectar nuestro controlador a la red a través de un router, switch o módem.

El control del módulo Ethernet se realiza a través del puerto SPI de microcontrolador 18F4520. El módulo Ethernet consta de un LED indicador de encendido, un LED indicador que se

ha recibido datos y un LED indicador de que se ha enviado datos, como se muestra en la Figura 6.8. Las principales aplicaciones que tiene este módulo son la comunicación, envío de parámetros por email, y visualización de parámetros en web sencilla.

Figura 6.8 Modulo Ethernet ENC28J60 [15]

En la tabla 6.1 se muestran las especificaciones técnicas del módulo Ethernet.

Tensión de alimentación	**5V, 3.3V**
Consumo	138mA a 3.3V / 145mA a 5V
Peso	16.6g

Conector de red	RJ45
Frecuencia de trabajo	25 MHz

Tabla 6.1 Especificaciones técnicas del módulo Ethernet

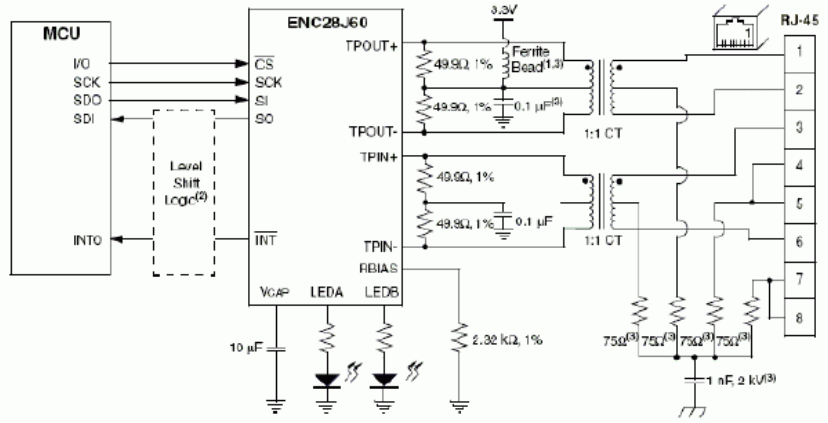


Figura 6.9 Esquema de conexión entre modulo Ethernet y Microcontroladores. [16]

6.2.3.1. Asignación de IP estática para el concentrador

Todo equipo el cual este recibiendo paquetes de datos, de distintos dispositivos en la red, en cualquier

momento, ellos deben responder ante la solicitud en cualquier momento, es decir debe tener un trabajo de 24/7, sin interrupciones de retardos de configuración, es por eso que se pensó en configurar una IP estática al concentrador, para que cuando la aplicación Android desee realizar una acción, él se encuentre siempre disponible, y evitar problemas de conexión entre la aplicación y el concentrador. Para la configuración de dicha IP estática se usó el siguiente segmento de código:

```
#define SPI_Ethernet_HALFDUPLEX     0
#define SPI_Ethernet_FULLDUPLEX     1

sfr sbit SPI_Ethernet_Rst at RC0_bit;
sfr sbit SPI_Ethernet_CS  at RC1_bit;
sfr sbit SPI_Ethernet_Rst_Direction at TRISC0_bit;
sfr sbit SPI_Ethernet_CS_Direction  at TRISC1_bit;

unsigned char myMacAddr[6] = {0x00, 0x14, 0xA5, 0x76, 0x19,
0x3f}; // mi MAC address
// mi IP addr
unsigned char myIpAddr     = {192, 168,   0, 60 };
SPI1_Init();
SPI_Ethernet_Init(myMacAddr, myIpAddr, SPI_Ethernet_FULLDUPLEX);
```

Figura 6.10 Código de configuración de IP del concentrador.

6.2.3.2. Asignación de Gateway y máscara de subred de manera estática, deshabilitando DHCP

Para una mejor integridad dentro de la red local, y para una completa configuración del concentrador, dentro de nuestra red LAN, se procedió a configurar de manera estática, los parámetros Gateway y máscara de subred, importantes dentro de una LAN. Para la configuración de dichos parámetros, se usó el siguiente segmento de código:

```
// network mask (for example : 255.255.255.0)
char ipMask[4]    = {255, 255, 255,  0 };
// gateway (router) IP address
char gwIpAddr[4]  = {192, 168,   1,  1 };
// DNS server IP address
char dnsIpAddr[4] = {192, 168,   1,  1 };
// set network configuration parameters

      SPI_Ethernet_confNetwork(ipMask, gwIpAddr,
      dnsIpAddr);
```

Figura 6.11 Código de configuración de Gateway y máscara de subred

6.3. Diseño y programación de base de datos

En esta sección se detalla los principales métodos de MySQL para obtener una conexión a la base de datos. Además se mencionan los reportes que generará el sistema.

6.3.1. Métodos principales de MySQL

MySQL es un gestor de base de datos, de código abierto, muy utilizado para dar soporte a la gestión de datos de una aplicación web. Es un software de código abierto escrito en C o en C++, que goza con privilegios de seguridad y contraseñas. Entre los métodos más usados dentro del gestor de base de datos MySQL, se tiene:

- La creación de la tabla de la base de datos.
- Asignación de columnas con su tipo de variable.
- Insertar datos en la base de datos.
- Leer registros dentro de la base de datos.
- Eliminar registros dentro de la base de datos.

6.3.2. Sección de reportes en el sistema domóticos

La aplicación Android “Control Home” podrá visualizar reportes, de datos que se almacenaran en un servidor de base de datos MySQL, de esta manera el usuario podrá tener un reporte mensual de las acciones realizadas en su hogar.

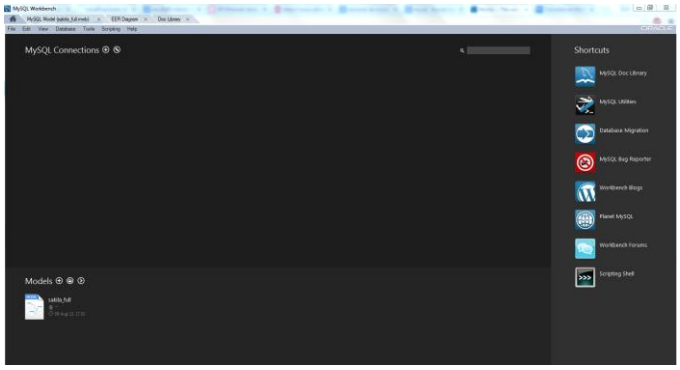


Figura 6.12 Interfaz de inicio de MySQL WorkBench

6.4. Diseño y programación de aplicación androide

Esta sección detalla el diseño de la aplicación a nivel de interfaz gráfica, como accederá el usuario a cada opción y se indicará la programación para obtener las funciones y vistas de la aplicación. Para este diseño se tomó en cuenta los principios de diseño para

una aplicación androide, así el uso de elementos con los que el usuario está familiarizado.

Se diseñó el ícono de la aplicación, de una manera en que pueda representar la finalidad de la misma, sin caer en el plano de ser aburrido ni sobrecargado.

Figura 6.13 Ícono de la aplicación “Control Home”

En la Figura 6.13 se muestra el ícono de la aplicación “Control Home”, para el que se trató represente la funcionalidad del sistema domótico planteado, cuya principal característica es ejercer control sobre los elementos del hogar, de manera inalámbrica.

6.4.1. Entorno de desarrollo Eclipse

Como se describe en otros capítulos, eclipse es un entorno de desarrollo integrado, capaz de aceptar diferentes lenguajes de programación, dándole una ventaja al usuario, en el cual con un solo IDE, pueda programar distintos lenguajes. Programar aplicaciones Android dentro de eclipse no es un problema, dado que el propio google sacó un plugin para eclipse, con las librerías necesarias para poder desarrollar aplicaciones en este ambiente.

6.4.2. Descarga e instalación del ADT bundle para Windows

Para instalar el plugin de Android en eclipse es muy sencilla, sólo se debe seguir los siguientes pasos:

- Abrir eclipse
- Dirigirse a la opción de: Help
- Escoger la opción de: install new software
- Pegar el siguiente link:
- https://dl-ssl.google.com/android/eclipse/

- Descargar los paquetes
- Aceptar la instalación
- Reiniciar eclipse

6.4.3. Actualización de distintas plataformas en el SDK tolos

Android en cada versión nueva de sistema operativo, saca nueva documentación de código, nuevos métodos de programación y nuevas librerías, estas actualizaciones funcionan acorde a la versión del sistema operativo, es por eso que se sugiere tener la mayor cantidad de Apis de desarrollo dentro de eclipse. Para proceder a la respectiva actualización de Apis de Android, procedemos a ejecutar el SDK manager de Android, incluido en el plugin descargado para eclipse.

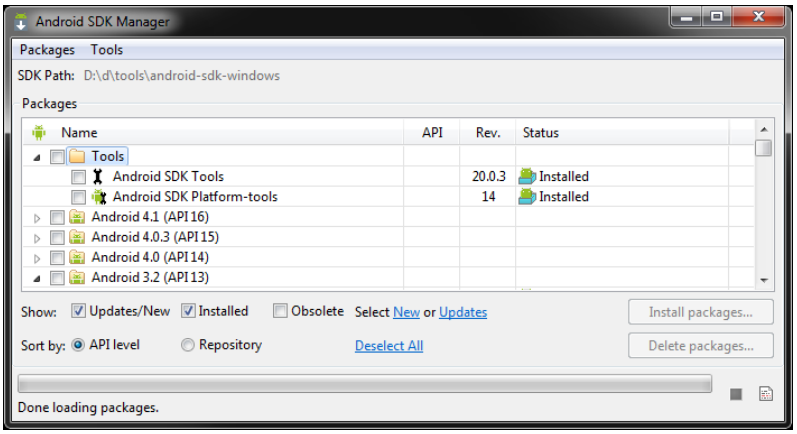


Figura 6.14 Interfaz de SDK manager.

6.4.4. Interfaz de autenticación de usuarios pertenecientes al hogar

La aplicación control home, cuenta con una pantalla inicial de autenticación para los usuarios del hogar, esto con el fin de poder brindar más seguridad y no cualquier persona pueda hacer uso de la aplicación y por ende de elementos en el hogar. En el Anexo G podrá observar la interfaz de autenticación.

6.4.4.1. Criterios y diseños

Esta sección hace referencia a las razones por las que se escogió una interfaz de autenticación básica y como se implementaron las funciones, para que la interacción con el usuario sea eficaz y sea amigable su uso, esta información se encuentra en el Anexo G.

6.4.4.2. Programación java y xml

La programación java es la que se hace cargo de manejar las acciones del usuario por medio de la interfaz, es decir si presiona cierto botón, entonces que sucede, también se da inicio a servicios de la aplicación. La programación xml se refiere a los elementos que verá el usuario en la pantalla, tiene que ver más con el diseño que con la funcionalidad. El código java y xml respectivos se encuentra en el Anexo H.

6.4.5. Interfaz de privilegios de usuarios pertenecientes al hogar

La aplicación “Control Home”, cuenta con una pantalla de privilegios de configuración para su sistema, así como personalización de la aplicación, esto se realizó con el fin de brindar una mejor experiencia de usuario, con el sistema diseñado, dado que no solo se podrá comprobar estados de funcionamiento, también mejor confort dentro de la aplicación. Sin embargo en esta pantalla hay restricciones de configuración, según el usuario que acceda a la misma.

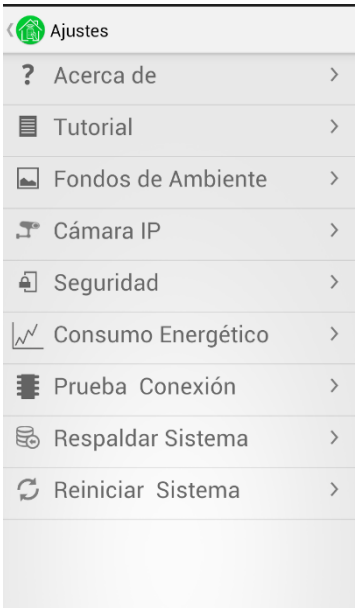


Figura 6.15 Privilegios de la aplicación.

6.4.5.1. Criterios y diseños

Esta sección hace referencia a las razones por las que se escogió una interfaz y como se implementaron las funciones, para que la interacción con el usuario sea eficaz, esta información se encuentra en el Anexo G.

6.4.5.2. Programación java y xml

La programación java es la que se hace cargo de manejar las acciones del usuario por medio de la interfaz, es decir si presiona cierto botón, entonces que sucede, también se da inicio a servicios de la aplicación. La programación xml se refiere a los elementos que verá el usuario en la pantalla, tiene que ver más con el diseño que con la funcionalidad. El código java y xml respectivos se encuentra en el Anexo H.

6.4.6. Interfaz principal de acciones en el sistema domótico

La aplicación control home, tiene una interfaz de acciones, amigable para el usuario, dado que divide las tareas en el hogar, por ambientes, esto le resulta al usuario una manera más ordenada y de fácil acceso, cuando desee realizar alguna acción sobre un elemento en su hogar.

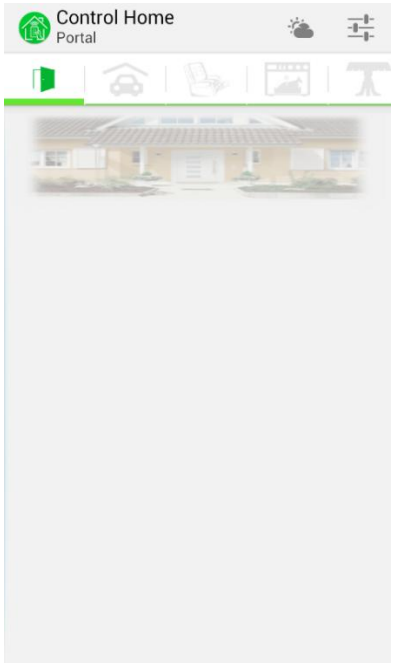


Figura 6.16 Ambientes del hogar

6.4.6.1. Criterios y diseños

Se escogió un diseño de la interfaz de tal manera que, el usuario pueda ejercer acciones sobre los elementos de su hogar sin experimentar cambios de pantalla, únicamente deslizando las pestañas horizontalmente para ubicarse en el ambiente que desee controlar. Esta información se encuentra en el Anexo G.

6.4.6.2. Programación java y xml

La pantalla mostrada en la Figura 6.15 le da accesibilidad al usuario para cualquier ambiente que desee controlar, sin embargo a nivel de programación se usa más de una clase para obtener la funcionalidad mostrada, también se da inicio al servicio de escuchar paquetes enviados por el controlador. La programación xml, tiene que ver más con el diseño que con la funcionalidad. El código java y xml respectivos se encuentra en el Anexo H.

CAPÍTULO 7

7. PRUEBAS DEL SISTEMA DOMÓTICO FINAL

En este capítulo se describen las pruebas que se realizaron al sistema, con los resultados obtenidos, pensando siempre en la buena experiencia del usuario, esto con lleva a un análisis no solo por parte del hardware, sino también por parte de la aplicación “Control Home” que maneja el usuario. Se realizaron cuatro pruebas importantes e indispensables para un sistema domótico.

7.1. Pruebas de ahorro de energía

Esta sección detalla a nivel de aplicación y sistema domótico los consumos generados por la aplicación y el uso eficiente de la energía por parte de los módulos del sistema.

Dado que la aplicación “Control Home” tiene 2 servicios ejecutándose todo el tiempo en background en el Smartphone, se procedió a realizar una prueba de consumo de batería, por parte de la aplicación, esto dado que al usuario siempre busca las aplicaciones que le consuman menos recursos en su dispositivo móvil. En la figura 7.1 se muestran los servicios de la aplicación, corriendo en background, estos son: ServiceUDPListener, quien es el encargado de receptar los datagramas UDP por parte del concentrador, y ServiceConsumoEnergético, quien es el encargado de calcular constantemente el consumo energético por parte de los dispositivos finales, conectados en el hogar.

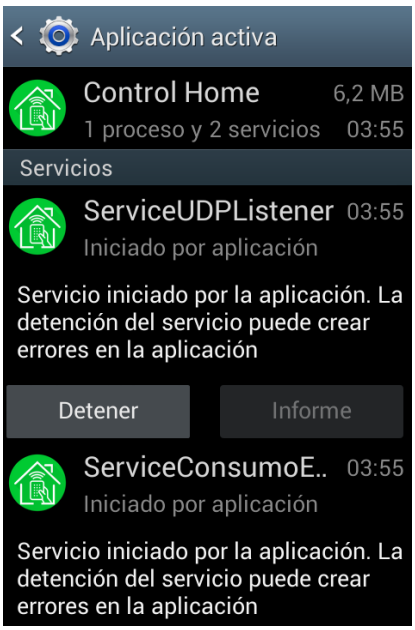


Figura 7.1 Servicios de Control Home

Algo muy importante para el usuario a la hora de usar una aplicación, es saber si esta aplicación le consume o no el paquetes de datos, en caso de que si le consuma, el usuario buscara siempre la aplicación que menos paquetes use, esto con motivo de ahorrar dinero para el mismo. Es por eso que se diseñó la aplicación, de tal manera que consuma la menor cantidad de bytes o paquetes de datos, con una trama de información eficiente. Cabe recalcar que la aplicación solo hará uso del paquete, cuando el usuario desee manejar o verificar dispositivos en su hogar, estando fuera del mismo.

Toda aplicación debe tener un tiempo de respuesta ante las acciones que el usuario elija o realice, es por eso que “Control Home” se programó de una manera en que sus acciones sean de respuesta rápida de no más de 1 segundo, para el caso de abrir la app, y de no más de 2 segundos para realizar acciones en el hogar. Para realizar esta prueba se procedió a usar una herramienta propia de android, que obtiene el porcentaje de recursos usado por las aplicaciones instaladas, en el móvil.

Figura 7.2 Detalles de la aplicación

En la figura 7.2 se muestra una imagen con los datos de consumo de recursos por parte de la aplicación, estos resultados son obtenidos directamente por parte del sistema operativo Android. Se puede apreciar que el consumo de la batería llega a un 2%, que la cantidad de bytes usado no llega ni a un 1KB.

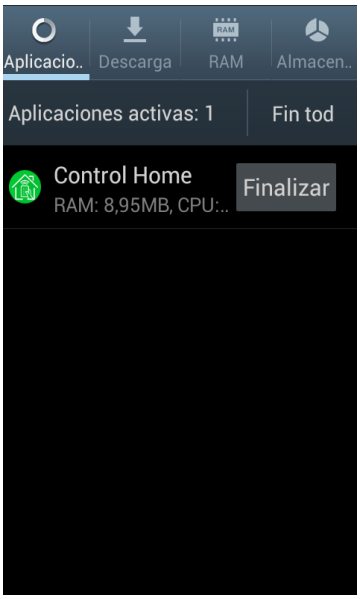


Figura 7.3 Uso de memoria RAM por la aplicación.

En la figura 7.3 se muestra una imagen donde se aprecia el consumo de memoria RAM en el dispositivo móvil, por parte de la aplicación Control Home. Según esta figura se aprecia que Control Home usa a penas 8 MB de la memoria RAM del móvil, mucho menos que otras aplicaciones que el usuario usa más a menudo, como Facebook, WhatsApp o Line.

Las pruebas de consumo de batería se las realizó durante una semana, obteniendo los siguientes resultados.

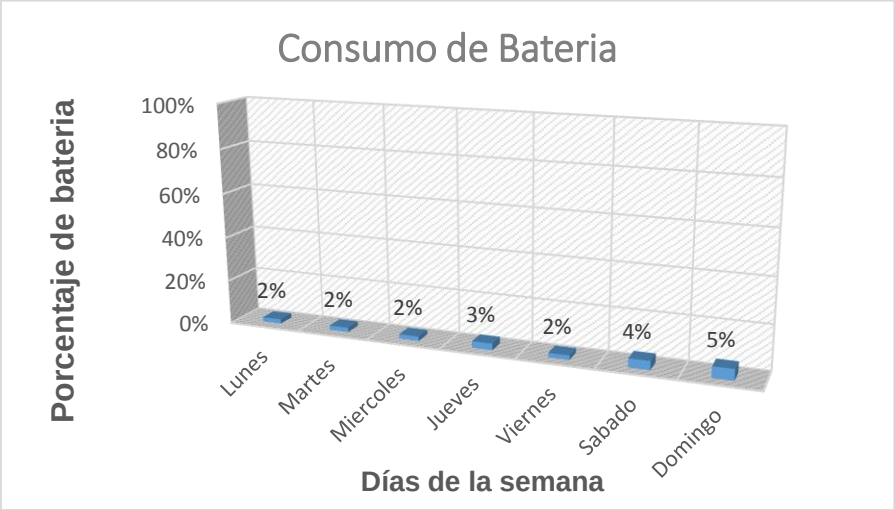


Figura 7.4 Resultados consumo de batería móvil

En la figura 7.4 se visualiza un gráfico, donde se puede apreciar un mayor uso de la aplicación los fines de semana, esto se debe a que en estos días, las familias hace un mayor control de los dispositivos o artefactos en su hogar, y por ende mayor uso en la aplicación.

Figura 7.5 Resultados consumo de Paquete de datos (bytes)

En la figura 7.5 se muestra el grafico estadístico del consumo de datos por parte de la aplicación. En este gráfico se puede apreciar que el consumo de datos no llega ni a un 1 MB por semana, esto siendo un gran beneficio y ventaja para el usuario.

A nivel de dispositivos finales se realizaron medidas de consumo energético desde la aplicación “Control Home”, dándole una opción muy útil al usuario, que significa poder elegir la cantidad en dólares que desea pagar por el consumo energético del mes, respecto a los dispositivos conectados al sistema. Esto se logra con un servicio que

se ejecutando en background, todo el tiempo, contabilizando la cantidad de KWh que se está consumiendo en el hogar, notificándole al usuario a manera de alarma, cuando ya se acerque límite del pago que programó. Esto se calculó en base a la tasa actual de pago de consumo energético, que es $ 0.089 aproximadamente por 1KWh.

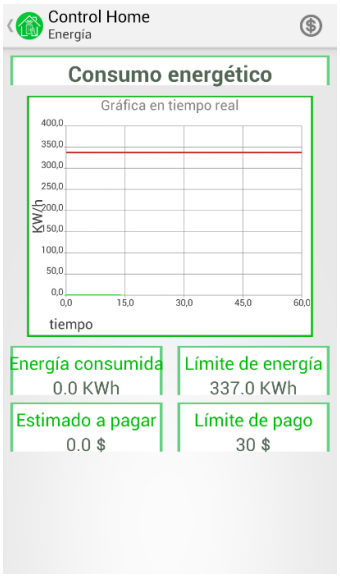


Figura 7.6 Consumo energético

En la figura 7.6 se muestra una imagen de la aplicación que corresponde a la actividad donde se informa al usuario sobre el consumo energético en tiempo real, así como el estimado a pagar.

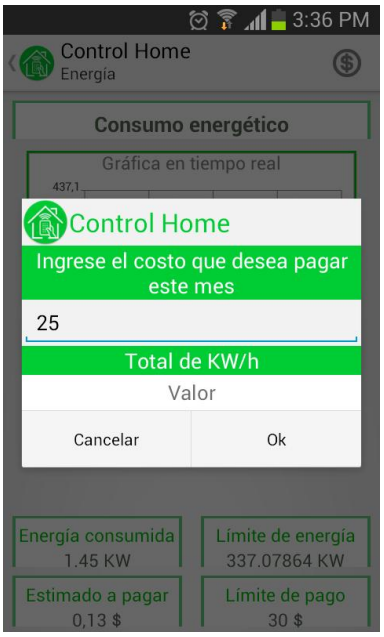


Figura 7.7 Configuración de límite a pagar

En la figura 7.7 se muestra el mensaje que el usuario visualizará cuando desee configurar el límite de pago que desee realizar, por el consumo energético. Una vez configurado este valor, el usuario podrá saber cuándo se esté aproximando a su límite de pago, gracias a la alarma informativa, que la aplicación lanzará automáticamente.

Figura 7.8 Alarma de proximidad a límite de pago

En la figura 7.8 se muestra una imagen de la aplicación, donde se puede visualizar la notificación de la aplicación, avisando al usuario que ya está cerca del límite de pago que previamente programó.

7.2. Pruebas de confort del sistema

Uno de los aspectos importantes de todo sistema domótico es generar confort al usuario, esto como se mencionó en el capítulo 1, referente a las automatizaciones de ciertas tareas, por lo que se realizó pruebas sobre los elementos agregados al hogar y las tareas automatizadas que estos podrían realizar. También el hecho de poder ejercer control sobre un elemento del hogar sin tener que estar cerca del mismo, por lo que una aplicación móvil es de las mejores opciones para lograr este objetivo.

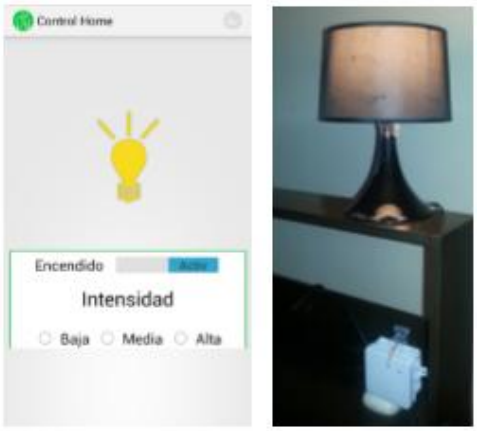


Figura 7.9 Encendido correcto de iluminaria

En la figura 7.9, se puede observar la acción de activación que se ejerce sobre la lámpara desde la aplicación móvil Control Home.

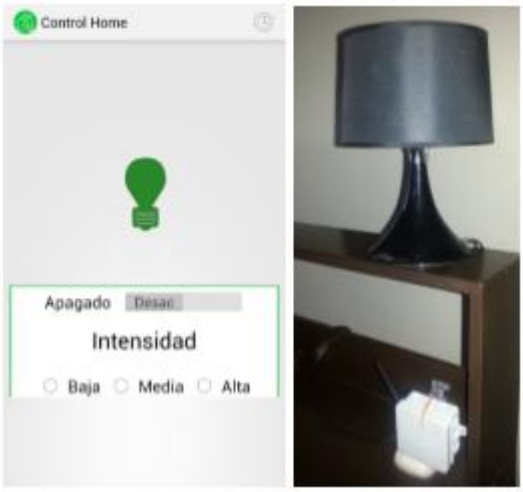


Figura 7.10 Apagado correcto de iluminaria

En la figura 7.10 podemos observar la acción de desactivación que el usuario ejerce sobre la lámpara, desde la aplicación y su exitosa ejecución.

Algo que ayuda a sentir confort es que todo lo que debe funcionar, funcione bien, para lo cual realizamos una prueba de conexión entre la aplicación y el concentrador, así el usuario confirmará que la comunicación con el cerebro de su sistema es correcta, caso

contrario al menos tendrá esa información disponible y podrá consultar con los desarrolladores.

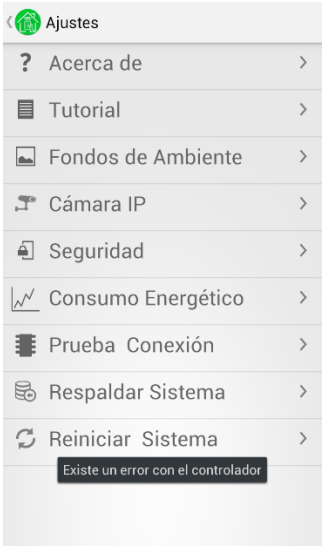


Figura 7.11 Prueba de conexión a concentrador

En la Figura 7.11 se puede observar el mensaje después de la prueba de conexión le indica al usuario, la comunicación y el concentrador es correcta.

Al usuario le suele suceder que olvida apagar algún foco, regadera, ventilador, etc., lo que genera consumo de energía innecesario y por ende mayor gasto económico. Sin embargo si se programa esta

tarea para ejecutarse en un tiempo y lapso de tiempo determinado por el usuario, ya no tendría mayor importancia el recordar realizar alguna tarea, porque automáticamente se ejecutará en la hora establecida.

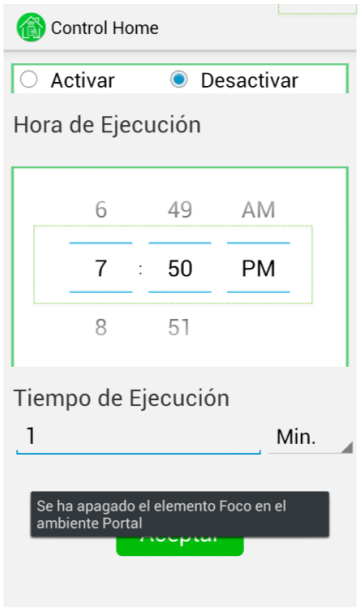


Figura 7.12 Programado de acción

En la figura 7.12 podemos observar, como se notifica al usuario que la acción que programó fue ejecutada, además podemos notar que la hora en la que se programó coincide con la hora del sistema operativo del Smartphone.

Es muy importante asegurar al usuario, que si su vivienda es amplia y no todos los elementos que desee controlar se encuentren cerca del concentrador, aun así podrá agregarlos y ejercer control sobre ellos. Se realizó una prueba con la finalidad de asegurar al usuario el alcance del sistema domótico inalámbrico, esto con respecto a que tan lejos del concentrador podrá ejercer control sobre los elementos del hogar, para lo cual se colocó un módulo actuador de un foco a una distancia de aproximadamente 35 metros del concentrador.

Figura 7.13 Prueba de alcance de dispositivos finales

En la figura 7.13 podemos observar un plano del hogar donde se realizó la prueba de alcance, nótese que el dispositivo final actuador se encuentra en la planta alta a una distancia de 35 metros del

concentrador, que se encuentra en la planta baja. Aun así el elemento fue agregado exitosamente al sistema.

7.3. Pruebas de seguridad

Sabemos que un sistema domótico deberá brindar seguridad al usuario, en esta sección detallaremos cómo funcionan las medidas de seguridad que el usuario tiene disponibles en el sistema domótico planteado.

En primera instancia se encuentran las opciones de seguridad como la configuración de contraseña y tipo de seguridad para acceder a la aplicación y ejercer control sobre los elementos del hogar, esto una vez que se haya configurado tener un tipo de seguridad alta. Para acceder a estas configuraciones de seguridad se debe dirigir a la lista de Ajustes mostrada en la Figura 6.14.

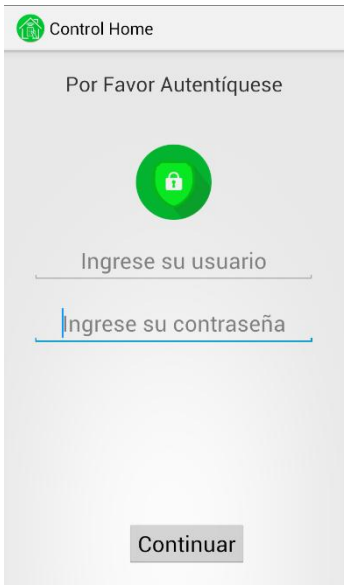


Figura 7.14 Contraseña para ingresar a aplicación

La Figura 7.14 describe el paso previo de autenticación que el usuario debe realizar para acceder o ingresar a la aplicación, impidiéndole control sobre los dispositivos del hogar, en caso de una autenticación no exitosa.

El usuario tiene también la opción de ver quien está en la entrada principal de su hogar y entonces tomar una decisión respecto a abrir

la puerta o no, esto brinda seguridad, ya que en cualquier momento podrá acceder al video vigilancia de la cámara.

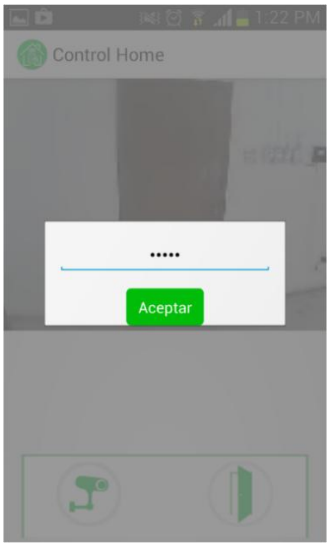


Figura 7.15 Contraseña para abrir puerta.

En la figura 7.15 se visualiza la autenticación necesaria por parte del usuario, para poder abrir la chapa eléctrica y por ende la puerta de su hogar, esto evitando que algún menor de edad abra la puerta cuando no deba.

7.4. Pruebas de reporte de base de datos

Se realizaron pruebas de importación y exportación de la base de datos, esto referente a que no se permita realizar importación de la base da datos si en la base de datos interna del teléfono exiten registros, antes de realizar una importación se deberá reiniciar el sistema. Así mismo si se desea realizar una exportación hacia la base de datos, se valida que existan registros para importar en la base interna del teléfono.

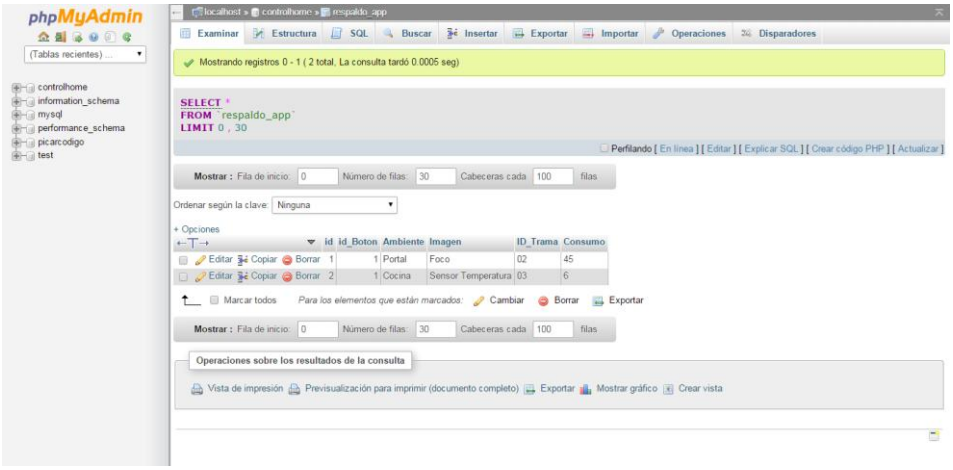


Figura 7.16 Registros exportados a la base de datos.

En la Figura 7.16 se muestra el resultado en la base de datos, una vez realizada una exportación de registros desde la aplicación Control Home.

CONCLUSIONES Y RECOMENDACIONES

1. El sistema domótico inalámbrico diseñado, dará las principales facilidades de ejecución de tareas automáticas, generando confort, así como también generará el consumo óptimo de energía como base mínima de todo sistema domótico.

2. Las medidas de seguridad que brinda el sistema, tales como acceso al sistema, detección de intrusos en el hogar, han sido controlados a través de accesos autenticados a la aplicación y alarmas de notificación al usuario.

3. Se pudo demostrar que, para implementar el sistema domótico planteado, no es necesario realizar grandes cambios en la infraestructura del hogar, esto referente a usuarios que deseen adquirir el sistema y ya tengan su hogar establecido.

4. Debido a la versión 2.3.3 (Gingerbread) mínima requerida para la instalación de la aplicación Control Home y según las estadísticas brindadas por Google, se concluye que el 99.6% de los usuarios de Android podrían adquirir la aplicación, lo que genera un mayor alcance del sistema domótico planteado.

5. Para generar mayores alcances a nivel de usuarios por sistema y mayor ejecución de tareas por parte del microcontrolador, se recomienda usar tecnología un tanto más avanzada, tales como sistemas embebidos o un procesador de mayor potencia y mejores características, como Arduino en conjunto con su Ethernet Shield.

6. Se recomienda implementar la seguridad que brindan los módulos XBee, esto referente a la configuración de un PAN ID y/o una contraseña de 32 dígitos para la red de módulos que forman parte del sistema, ya que con esto evitamos confusiones de tramas o ataques de intrusos, debido a que otros módulos externos al sistema puedan escuchar las tramas enviadas, si no se implementan estas medida.

7. Por motivos de seguridad, se recomienda el uso de Web Services, para las consultas, importación y exportación de registros de la base de datos, debido a que un usuario no debe tener acceso directo a los parámetros de conexión de la misma.

8. Se recomienda elegir adecuadamente los canales de frecuencia de trabajo, para los módulos inalámbricos XBee, esto con el fin de evitar interferencias con el router o módem del hogar.

ANEXOS

ANEXO A

PROGRAMACIÓN DE MICROCONTROLADORES

Programación de microcontrolador PIC18f4520, para el Concentrador.

```
// duplex config flags
#include  "__EthEnc28j60.h"

#define Spi_Ethernet_HALFDUPLEX     0x00  // half duplex
#define Spi_Ethernet_FULLDUPLEX     0x01  // full duplex

// mE ehternet NIC pinout
sfr sbit SPI_Ethernet_Rst at RC0_bit;
sfr sbit SPI_Ethernet_CS  at RC1_bit;
sfr sbit SPI_Ethernet_Rst_Direction at TRISC0_bit;
sfr sbit SPI_Ethernet_CS_Direction  at TRISC1_bit;
// end ethernet NIC definitions

unsigned char recibido[10];
int posByte=0;

/***********************************
 * RAM variables
 */
unsigned char   myMacAddr[6] = {0x00, 0x14, 0xA5, 0x76, 0x19, 0x3f} ;   // my MAC
address
unsigned char   myIpAddr[4]  = {192, 168,  0, 60 } ;                    // my IP address
unsigned char   gwIpAddr[4]  = {192, 168,  0,  1 } ;                    // gateway (router) IP
address
unsigned char   ipMask[4]    = {255, 255, 255,  0 } ;                   // network mask (for
example : 255.255.255.0)
unsigned char   dnsIpAddr[4] = {192, 168,  0,  1 } ;                    // DNS server IP address
unsigned char   destIpAddr[4] = {192, 168,  0, 255 } ;
unsigned char   dataApp[10];
unsigned char   dyna[29] ;
unsigned long   httpCounter = 0 ;
char uart_rd;
char trama[10];
int len=0;
int flag=0;
#define putConstString  SPI_Ethernet_putConstString
```

```
#define putString  SPI_Ethernet_putString
unsigned int  SPI_Ethernet_UserTCP(unsigned char *remoteHost, unsigned int
remotePort, unsigned int localPort, unsigned int reqLength, TEthPktFlags *flags)
{
      return(0);
 }
unsigned int  SPI_Ethernet_UserUDP(unsigned char *remoteHost, unsigned int
remotePort, unsigned int destPort, unsigned int reqLength, TEthPktFlags *flags)
 {

      if(remotePort==5506) //ESTE PUERTO ES EL DE LA APP
       {

       SPI_Ethernet_getBytes(dataApp,0xFFFF,10);
       dataApp[10] =0;

         if(dataApp[0]==37&&dataApp[1]==38&&dataApp[2]==105&&dataApp[3]==115
         &&dataApp[4]==111&&dataApp[5]==107&&dataApp[6]==63&&dataApp[7]==33
         &&dataApp[8]==64)
         {
              SPI_Ethernet_putBytes("PIC_OK",6);
              return(6);
         }
         else{
              UART1_Write_Text(dataApp);
         }
      }
}

/*
 * main entry
 */
void    main()
{
      TRISB = 0 ;
      PORTB=1;
      /*
       * starts ENC28J60 with :
       * reset bit on RC0
       * CS bit on RC1
       * my MAC & IP address
       * full duplex
       */
      SPI1_Init();
```

```
    SPI_Rd_Ptr = SPI1_Read;
    SPI_Ethernet_Enable(_SPI_Ethernet_BROADCAST| _SPI_Ethernet_UNICAST|
_SPI_Ethernet_MULTICAST);
    SPI_Ethernet_Init(myMacAddr, myIpAddr, Spi_Ethernet_FULLDUPLEX) ;

    // CONFIGURACIÓN ESTÁTICA DE PARÁMETROS
    SPI_Ethernet_confNetwork(ipMask, gwIpAddr, dnsIpAddr) ;
    UART1_Init(9600);
    delay_ms(100);
    while (1)
    {
    SPI_Ethernet_doPacket() ;       // procesamiento de paquetes entrantes ethernet
    if (UART1_Data_Ready())
    {
      uart_rd = UART1_Read();     // Leer un byte recibido
          switch(len)
          {
              case 0:
              trama[0]=UART1_Read();
              break;
              case 1:
              trama[1]=UART1_Read();
              break;
              case 2:
              trama[2]=UART1_Read();
              break;
              case 3:
              trama[3]=UART1_Read();
              break;
              case 4:
              trama[4]=UART1_Read();
              break;
              case 5:
              trama[5]=UART1_Read();
              break;
              case 6:
              trama[6]=UART1_Read();
              case 7:
              trama[7]=UART1_Read();
              break;
              case 8:
              trama[8]=UART1_Read();
              break;
              case 9:
```

```
                trama[9]='\0';
                break;
                default:
                break;
               }
               len++;
               if(uart_rd==64) //Delimitador @
               {
                  flag=1;
               }
    }
    if(flag==1)
    {
        // ESTA FUNCIÓN ENVÍA EL DATAGRAMA UDP, CON LA INFORMACIÓN
       RECIBIDA POR EL PUERTO SERIAL DEL CONCETRADOR
     SPI_Ethernet_sendUDP(destIpAddr,5505,11111,trama,9);
     len=0;
     flag=0;
    }
}
}
```

Programación del módulo actuador para un foco, lámpara, persiana o regadera.

```
char uart_rd;
char trama[10];
unsigned short asignado;
int porcentaje;
int totalPWM;
int total=1;
int centenas=0;
int decenas=0;
int unidades=0;
unsigned int light;
int len=0;
int flag=0;
void main()
{
  ANSEL  = 0x08;                    // Configure AN pins as digital
```

```
 ANSELH = 0;
 TRISA=0b00001100;
 PORTA=0;
 C1ON_bit = 0;                    // Disable comparators
 C2ON_bit = 0;
 RC1_bit = 0;                     // Disable comparators
 RC2_bit=0;
 TRISC1_bit = 0;                  // designate PORTC pins as output
 TRISC2_bit =0;
 PWM2_Init(5000);                 // Initialize PWM1 module at 5KHz  */
 UART1_Init(9600);                // Initialize UART module at 9600 bps
 Delay_ms(100);                   // Wait for UART module to stabilize
 PWM2_Start();
 PWM2_Set_Duty(10);
 while (1)
{                          // Endless loop
        if(total!=2)
        {
                if((asignado=EEPROM_Read(0x00))==255)
                {
                UART1_Write_text("NUEVOAI01@");
                total++;
                }
 }
  if (UART1_Data_Ready())
  {
    uart_rd = UART1_Read();     // read the received data,
    switch(len)
    {
     case 0:
     trama[0]=UART1_Read();
     break;
     case 1:
     trama[1]=UART1_Read();
     break;
     case 2:
     trama[2]=UART1_Read();
     break;
     case 3:
     trama[3]=UART1_Read();
     break;
     case 4:
     trama[4]=UART1_Read();
     break;
```

```
        case 5:
       trama[5]=UART1_Read();
       break;
       case 6:
       trama[6]=UART1_Read();
       case 7:
       trama[7]=UART1_Read();
       case 8:
       trama[8]=UART1_Read();
       case 9:
       trama[9]='\0';
       break;
       default:
       break;
      }
      len++;
      if(uart_rd==64)
      {
       flag=1;
      }
 }


if(flag==1){
              if(trama[4]==101
              &&trama[5]==110&&trama[6]==100&&trama[7]==48&&trama[8]==49)
              //PARA ESTE CASO SOY end1
              {
                  if(trama[0]==37&&trama[1]==36 && trama[2]==111&&trama[3]==107)
                  { //PARA ESTE CASO SOY %$OK
                    EEPROM_Write(0x00, 1);
                  }
                   else if(trama[0]==49)
                  {
                  EEPROM_Write(0x01,255);
                 PWM2_Set_Duty(255);
                  }
                  else if(trama[0]==48)
                  {
                            EEPROM_Write(0x01,255);
                            PWM2_Set_Duty(0);

                  }
                  else if(trama[0]==50)
```

```
                {
                        EEPROM_Write(0x01,255);
                        centenas=trama[1]-'0';
                        centenas=centenas*100;
                        decenas=trama[2]-'0';
                        decenas=decenas*10;
                        unidades=trama[3]-'0';
                         unidades=unidades*1;
                        porcentaje=centenas+decenas+unidades;
                        totalPWM=((porcentaje*255)/100);
                        PWM2_Set_Duty(totalPWM);
                 }
               }
               len=0;
               flag=0;
    }
    if(RA2_bit==1)
        {
        EEPROM_Write(0x00,255);
        //total=1;
        }
 }
}
```

Programación del módulo actuador para sensores.

```
char uart_rd;
char trama[10];
unsigned short asignado;
int total=1;
unsigned int light;
int len=0;
int flag=0;
int ind=0;
void main() {
  ANSEL  = 0x08;                  // Configure AN pins as digital
  ANSELH = 0;
  TRISA=0b00001101;
  PORTA=0;
  C1ON_bit = 0;                   // Disable comparators
  C2ON_bit = 0;
  UART1_Init(9600);               // Initialize UART module at 9600 bps
  Delay_ms(100);                 // Wait for UART module to stabilize

  while (1) {                   // Endless loop

  if(total!=2)
   {
    if((asignado=EEPROM_Read(0x00))==255)
    {
      UART1_Write_text("NUEVOSE02@");
      total++;
   }
  }

   if (UART1_Data_Ready())
   {
     uart_rd = UART1_Read();     // read the received data,

     switch(len)
     {
      case 0:
      trama[0]=UART1_Read();
      break;
      case 1:
      trama[1]=UART1_Read();
      break;
```

```
 case 2:
 trama[2]=UART1_Read();
 break;
 case 3:
 trama[3]=UART1_Read();
 break;
 case 4:
 trama[4]=UART1_Read();
 break;
  case 5:
 trama[5]=UART1_Read();
 break;
 case 6:
 trama[6]=UART1_Read();
 case 7:
 trama[7]=UART1_Read();
 case 8:
 trama[8]= UART1_Read();
 case 9:
 trama[9]='\0';
 break;
 default:
 break;
}
len++;
if(uart_rd==64)
{
 flag=1;
}
}
if(flag==1)
{
            if(trama[4]==101&&trama[5]==110&&trama[6]==100&&trama[7]==48&&
            trama[8]==50)
            //PARA ESTE CASO SOY end02
            {
          if(trama[0]==37&&trama[1]==36 && trama[2]==111&&trama[3]==107)
          {
           //PARA ESTE CASO SOY %$OK
              EEPROM_Write(0x00, 1);
          }
          else if(trama[0]==49)//LA APP ACTIVO EL MODO ALERTA
          {
                if(RA2_bit==1)
```

```
                {
                 UART1_Write_Text("SP0012@");
                }
                else if(RA2_bit==0)
                {
                 UART1_Write_Text("SP0002@");
                }
             }
            }
            len=0;
            flag=0;
    }
    if(RA2_bit==1)     //DETECTO PRESENCIA Y ENVÍA ENCENDER
    {
        if(ind==0)
       {
                UART1_Write_Text("1000end01@");
                ind=1;
       }

    }

    else if(RA2_bit==0)     //DETECTO PRESENCIA Y ENVÍA ENCENDER
    {
        if(ind==1)
       {
                UART1_Write_Text("0000end01@");
                ind=0;
       }
    }
     if(RA0_bit==1)
    {
      EEPROM_Write(0x00, 255);
    }
  }
}
```

ANEXO B

CLASES Y MÉTODOS PARA LA COMUNICACIÓN EN RED

Aquí mencionaremos las clases que intervienen en la comunicación en red del concentrador con la aplicación, esto incluye los servicios que escuchan los datagramas UDP provenientes del concentrador.

Clase EnviarDatoUDP.java

Esta clase es usada cada vez que la aplicación envía un datagrama UDP al concentrador, ya sea para confirmar la agregación de un dispositivo final, ordenar acción sobre un dispositivo final, o solicitar la lectura de un sensor.

```
public class EnviarDatoUDP {

    public static Handler Handler;
    public String dato="";
    public static final String SERVERIP = "190.131.90.84";
    public static final int SERVERPORT = 5505;
    public static final int PuertoLocalenvio=5506;


    public void IniciarClienteUDP(String comando)
    {
        new Thread(new Client(comando)).start();
        Handler = new Handler()
        {
            @Override
            public void handleMessage(Message msg)
            {

            }
        };
```

```
    }

class Client implements Runnable {

        public Client(String comando) {
         // TODO Auto-generated constructor stub
         dato = comando;
}



          @Override
public void run() {

    try {
                Thread.sleep(500);
          } catch (InterruptedException e1) {
                    // TODO Auto-generated catch block
             e1.printStackTrace();
          }
    try {
          InetAddress serverAddr = InetAddress.getByName(SERVERIP);
          DatagramSocket socket = new DatagramSocket(PuertoLocalenvio);
          byte[] buf;
          buf=dato.getBytes();
          DatagramPacket packet = new DatagramPacket(buf, buf.length, serverAddr,
SERVERPORT);

          socket.send(packet);

          Log.i("UDP","Se envio el paquete: "+dato);
          socket.disconnect();
          socket.close();
         } catch (Exception e) {

         }
         }

    }

}
```

Clase ServiceUDPListener.java

El servicio que permite a la aplicación escuchar los datagramas UDP enviados por el concentrador está desarrollado, para los cuales realiza las validaciones necesarias para saber de qué evento proviene el datagrama recibido, y así enviar la notificación correspondiente al usuario. A continuación el diagrama de flujo de la clase ServiceUDPListener y su código de programación.

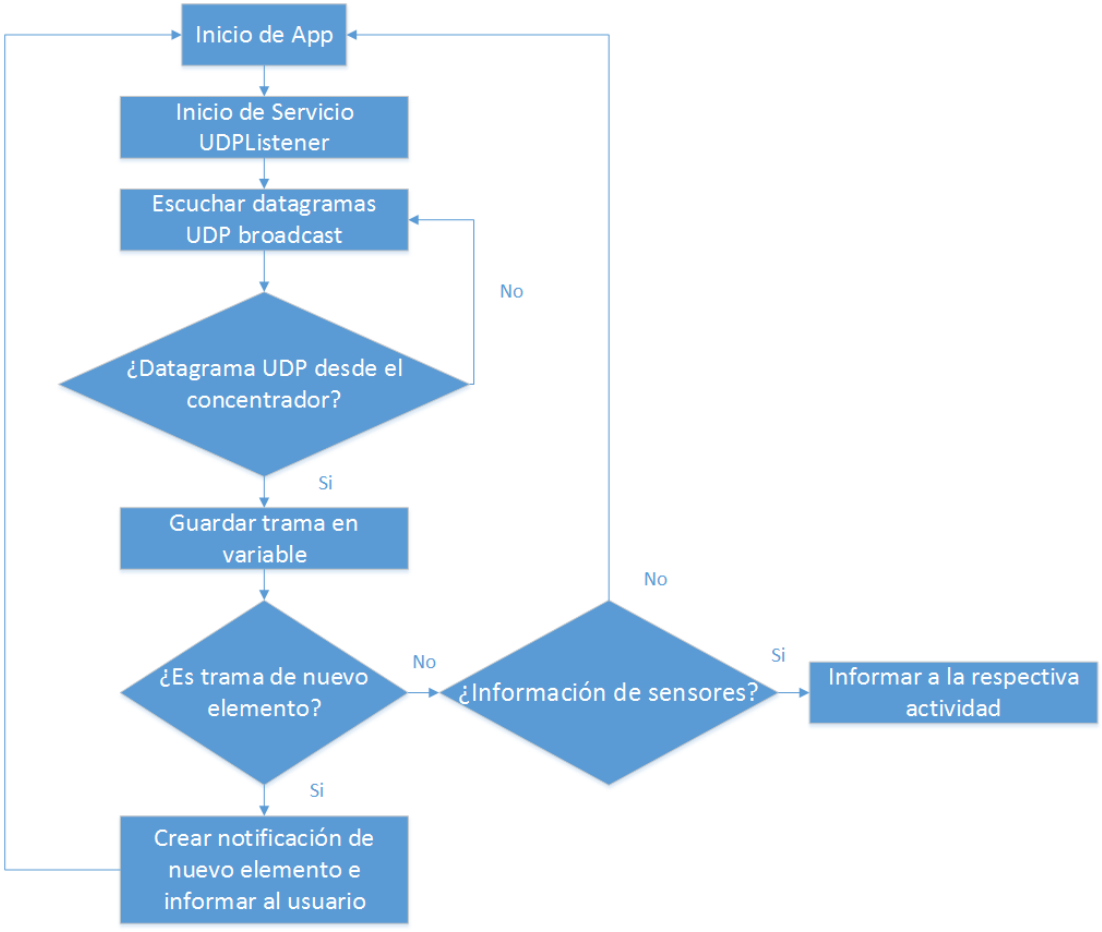


Figura B1 Diagrama de flujo de la clase ServiceUDPListener

```
public class ServiceUDPListener extends Service {
    private static ServiceUDPListener instance  = null;
    static String UDP_BROADCAST = "UDPBroadcast";
    String senderIP="";



    //Boolean shouldListenForUDPBroadcast = false;
    DatagramSocket socket;
    private String message;

    private void listenAndWaitAndThrowIntent(InetAddress
broadcastIP, Integer port) throws Exception {
        byte[] recvBuf = new byte[9];
        if (socket == null || socket.isClosed()) {
            socket = new DatagramSocket(port,broadcastIP);
            socket.setBroadcast(true);
        }
        //socket.setSoTimeout(1000);
        DatagramPacket packet = new DatagramPacket(recvBuf,
recvBuf.length);
        Log.i("UDP", "Waiting for UDP broadcast");
        try
        {
            socket.receive(packet);
            senderIP = packet.getAddress().getHostAddress();
            message = new String(packet.getData());
            Log.i("UDP", "Got UDB broadcast from " + senderIP +
", message: " + message);
        }
        catch (IOException e)
        {
            e.printStackTrace();
            Log.i("UDP", "Error: "+ e);
        }

        if(!senderIP.equals("")&&!message.equals(""))
        {
            String comp=message.substring(0, 5);
            if(comp.equals("NUEVO")){
                lanzarNotificacion();
            }
            else{
                //INFO SENSORES
                broadcastIntent(message);
```

```
            }
        }
        socket.close();
    }

    private void broadcastIntent(String message) {

        Intent intent = new
Intent(ServiceUDPListener.UDP_BROADCAST);
        intent.setAction("ACTUALIZACION_SENSORES");
        intent.putExtra("Trama", message);
        intent.putExtra("Update", getDatos(message));
        broadcaster.sendBroadcast(intent);
    }


    Thread UDPBroadcastThread;

    void startListenForUDPBroadcast() {
        UDPBroadcastThread = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    InetAddress broadcastIP =
InetAddress.getByName("192.168.1.255");
                    Integer port = 11111;
                    while (shouldRestartSocketListen) {
                        listenAndWaitAndThrowIntent(broadcastIP,
port);
                    }
                    //if (!shouldListenForUDPBroadcast) throw new
ThreadDeath();
                } catch (Exception e) {
                    Log.i("UDP", "no longer listening for UDP
broadcasts cause of error " + e.getMessage());
                    //Toast.makeText(UDPListenerService.this, "no
longer listening for UDP broadcasts cause of error " +
e.getMessage(), Toast.LENGTH_LONG).show();
                }
            }
        });
        UDPBroadcastThread.start();
    }

    private Boolean shouldRestartSocketListen=true;
```

```
    private LocalBroadcastManager broadcaster;

    void stopListen() {
        shouldRestartSocketListen = false;
        socket.close();
    }

    @Override
    public void onCreate() {
        broadcaster = LocalBroadcastManager.getInstance(this);
    };

    @Override
    public void onDestroy() {
        //stopListen();
    }


    @Override
    public int onStartCommand(Intent intent, int flags, int
startId) {
        shouldRestartSocketListen = true;
        startListenForUDPBroadcast();
        Log.i("UDP", "Service started");
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @SuppressWarnings("deprecation")
    public void lanzarNotificacion(){
         String ns = Context.NOTIFICATION_SERVICE;
         NotificationManager notManager = (NotificationManager)
getSystemService(ns);

         //Configuramos la notificación
         Notification notif = new
Notification(R.drawable.ic_noti_new_beta, "Nuevo control en tu
hogar", System.currentTimeMillis());
         //Configuramos el Intent
         Context contexto = ServiceUDPListener.this;
         CharSequence titulo = "Notificación de ControlHome";
```

```
        CharSequence descripcion = "Presione para configurar su
nuevo equipo";
        Intent resultIntent =new
Intent(ServiceUDPListener.this,NuevoElementoActivity.class);
        resultIntent.putExtra("IpConcentrador", senderIP);
        resultIntent.putExtra("Mensaje",
message.substring(7,9));
        resultIntent.putExtra("elemento",
message.substring(5,7));
        PendingIntent contIntent =
PendingIntent.getActivity(contexto, 0, resultIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

        notif.setLatestEventInfo(contexto, titulo, descripcion,
contIntent);
        long[] vibrate = {100,100,200,300};
        notif.vibrate = vibrate;
        notif.flags |=
Notification.FLAG_ONGOING_EVENT|Notification.FLAG_AUTO_CANCEL;
        notif.defaults |=
Notification.DEFAULT_VIBRATE|Notification.DEFAULT_SOUND;
        notManager.notify(1, notif);
     }

    public static boolean isRunning() {
          return instance != null;
    }

    private InformacionSensores getDatos(String mensaje)
    {
        InformacionSensores s = new InformacionSensores();

        if(mensaje.substring(0, 2).equals("SS"))
        {
            s.setHumo(message.substring(2,5));

        }
        else if(mensaje.substring(0, 2).equals("SP"))
        {
            s.setPresencia(message.substring(2,5));
        }

        else if(mensaje.substring(0, 2).equals("SH"))
        {
            s.setHumedad(message.substring(2,5));
```

```
        }

        else if(mensaje.substring(0, 2).equals("ST"))
        {
            s.setTemperatura(message.substring(2,5));
        }
        s.setID(message.substring(5));
        return s;
    }
}
```

ANEXO C

MÓDULOS INALÁMBRICOS XBEE

Los módulos inalámbricos XBee son módulos de radio frecuencias, que pueden trabajar en la banda libre de 2.4 GHz con protocolo de comunicación 802.15.4, fabricados por Digi. Se los usa en sistemas de automatización, sistemas de seguridad, monitoreo de sistemas remoto, alarmas contra incendios, plantas tratadoras de agua, etc.

Existen dos modelos principales de módulos XBee, que son:

- Módulos RF XBee: Estos módulos tienen un alcance en interiores de hasta 30 m, y en exteriores de 100 m con antena dipolo.
- Módulos RF XBee PRO: Estos módulos tienen un alcance en interiores de hasta 100 m, y en exteriores de 1500 m con antena dipolo.

Los módulos tienen 6 convertidores análogo-digital y 8 entradas digitales además de pines Rx y Tx, para comunicación serial. Trabajan a 2.4 GHz y generan una red propia a la que se puede conectar o desconectar. Son módulos microprocesados con lo cual tiene solucionados los problemas de fallo de trama, ruidos, etc. Los módulos, se comunican con un dispositivo RS232 a niveles TTL con lo cual la comunicación necesita un adaptador intermedio en el caso de un PC, pero pueden conectarse directamente a una placa de desarrollo.

Los módulos ofrecen una velocidad de comunicación hasta 256 Kbps pasando por todos los valores convencionales, también disponen de varias I/O q ue pueden ser configuradas para diferentes funciones.

Figura C1 Módulo Inalámbrico XBee Pro S1

Entre las necesidades que satisfacen los módulos se encuentran:

- Bajo costo.
- Muy bajo consumo de potencia.
- Uso de bandas de radio libres y sin necesidad de licencias.
- Instalación barata y simple.
- Redes flexibles y extensibles.

Una red ZigBee está básicamente formada por 3 tipos de elementos, estos son:

- Dispositivo Coordinador: Nodo encargado de formar la red, dado que es el responsable de formar el canal de comunicaciones y del PAN ID de toda la

red. Una vez formada la red este dispositivo también es capaz de enrutar paquetes, y ser fuente de origen y/o destinatario.

- Dispositivo Router: Nodo que crea y mantiene información sobre la red para determinar la mejor ruta para transmitir un paquete de información. Inicialmente este dispositivo debe estar asociado a la red ZigBee para poder enrutar paquetes de dispositivos XBee finales o de otros routers.
- Dispositivos de puntos finales: Estos dispositivos son incapaces de enrutar paquetes, deben interactuar siempre a través de un nodo padre. Dado que no enrutan paquetes, estos consumen menos energía.

Los módulos XBee tienen el siguiente circuito básico:

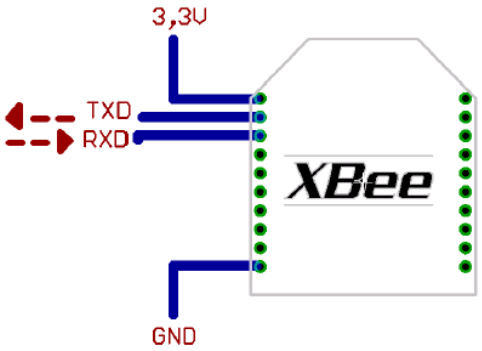


Figura C2 Conexión básica de Módulos XBee [19]

Para que el módulo XBee funcione correctamente y pueda ser utilizado, necesita estar energizado de 2.8 V a 3.4 V, la conexión a tierra y las líneas de comunicación por medio del UART (Tx y Rx).

Cabe recalcar que con esta configuración no se puede dar el uso del control de flujo (RTS & CTS). En caso que el tráfico de datos sea extremadamente grande, el buffer del módulo se suele saturar. Para resolver esto tenemos dos opciones:

- Bajar la tasa de transmisión.
- Activar el control de flujo.

Los módulos XBee tienen los siguientes modos de operación:

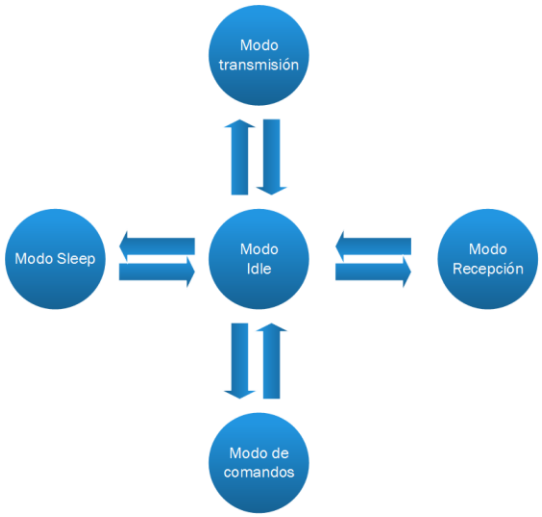


Figura C3 Modos de operación de módulos XBee

- Modo recepción/transmisión: Cuando al módulo le llega algún paquete RF a través de la antena(modo Receive) o cuando se manda información serial al buffer del pin 3 (UART Data in) que luego será transmitida (modo Transmit).
- Modo de bajo consumo (Sleep): Este modo permite al módulo XBee entrar en un modo de bajo consumo de energía.
- La configuración de los ciclos de sueño se realiza principalmente con el comando SM. Por defecto, los modos de sueños están deshabilitados (SM=0), permaneciendo el módulo en estado de reposo/recepción. En este estado el módulo está siempre preparado para responder a un comando, ya sea, por el puerto serial o la interfaz RF.
- Modo comando: Este modo sirve para configurar el módulo XBee, mediante comandos AT. Permite configurar parámetros como la dirección propia o la de destino, así como su modo de operación entre otras cosas.
- Modo IDLE: El módulo XBee entra en este modo, cuando no se encuentra en ninguno de los otros modos, es decir, ni transmitiendo, ni recibiendo, ni en modo de comandos, ni en modo Sleep.

A continuación se muestra una imagen donde se detalla los canales disponibles que usan los módulos XBee, para poder transmitir o recibir información, usando el protocolo 802.15.4

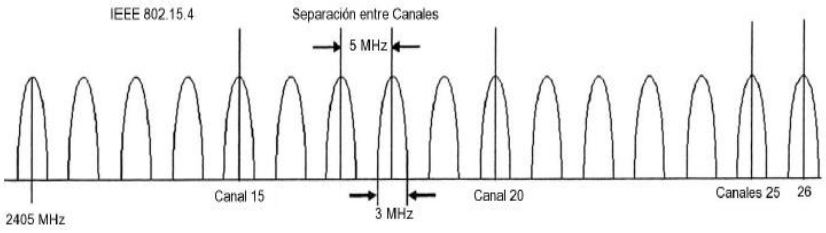


Figura C4 Canales disponibles de transmisión de módulos XBee [20]

En esta imagen se logra apreciar que se tienen 16 canales disponibles. Estos canales se asignan desde el 11 hasta el 26.

Para calcular la frecuencia central, se usa la siguiente formula:

$$canal = 2.405 + (CH - 11)\ x\ 0.005\ [GHz\ ]$$

Ecuación C1 Ecuación para frecuencia central

Donde CH equivale al canal del 11 al 26, en el cual se desea trabajar.

ANEXO D

Diagrama funcional del sistema completo

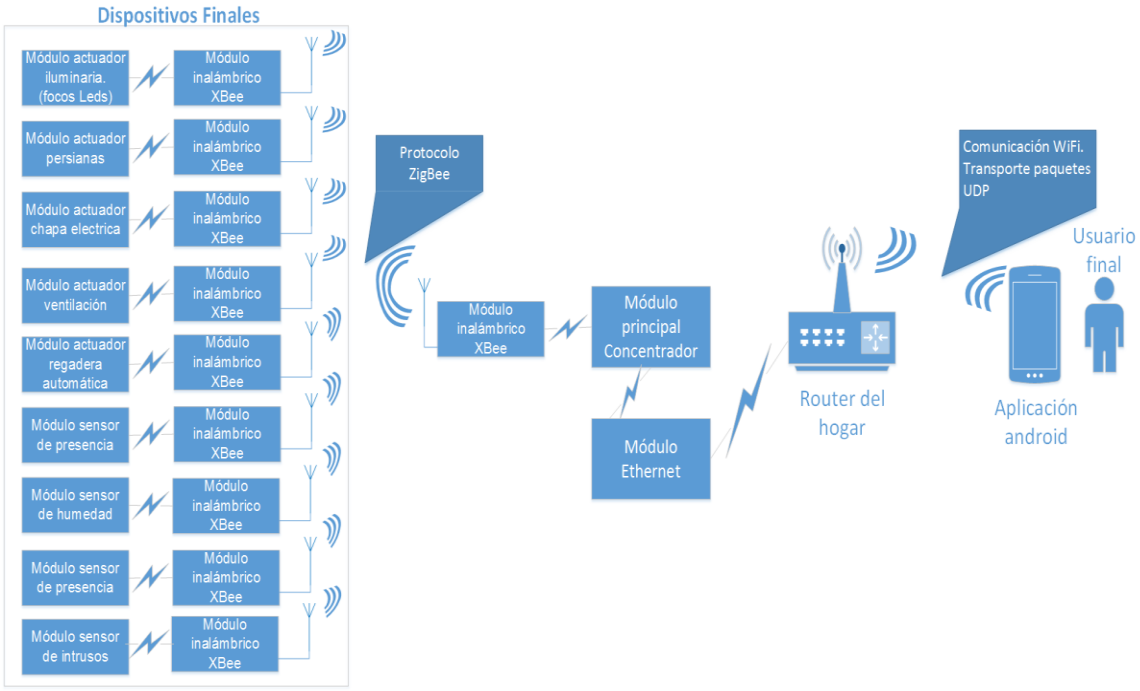


Figura D1 Diagrama de bloques del sistema domótico

En la Figura D1 se nota el tipo de comunicación que existe entre cada parte del sistema. En el diagrama se aprecia que existe un desarrollo por bloques, esto se realiza con el fin de dar mayor integridad al sistema, convirtiendo problemas complejos en partes más simples.

ANEXO E

DIAGRAMAS DE EJEMPLOS DE ESCENARIOS DEL SISTEMA DOMÓTICO PLANTEADO

En este anexo se mostrarán los diagramas de los procesos que se llevan a cabo en cada uno de los escenarios de comunicación que se pueden presentar en el sistema domótico.

Agregar un dispositivo final foco

Figura E1 Diagrama secuencial de dispositivo final agregado (Parte 1)

En la Figura E1 se puede visualizar los procesos que se dan al momento de energizar un nuevo elemento al sistema. El diagrama mostrado sólo pertenece a

la parte del proceso en la que el dispositivo final envía la petición de agregación al sistema, esto mediante comunicación inalámbrica entre los módulos XBee del dispositivo final y el concentrador, respectivamente. Como paso final en esta parte del proceso, el concentrador envía la petición del dispositivo a la aplicación mediante WiFi.

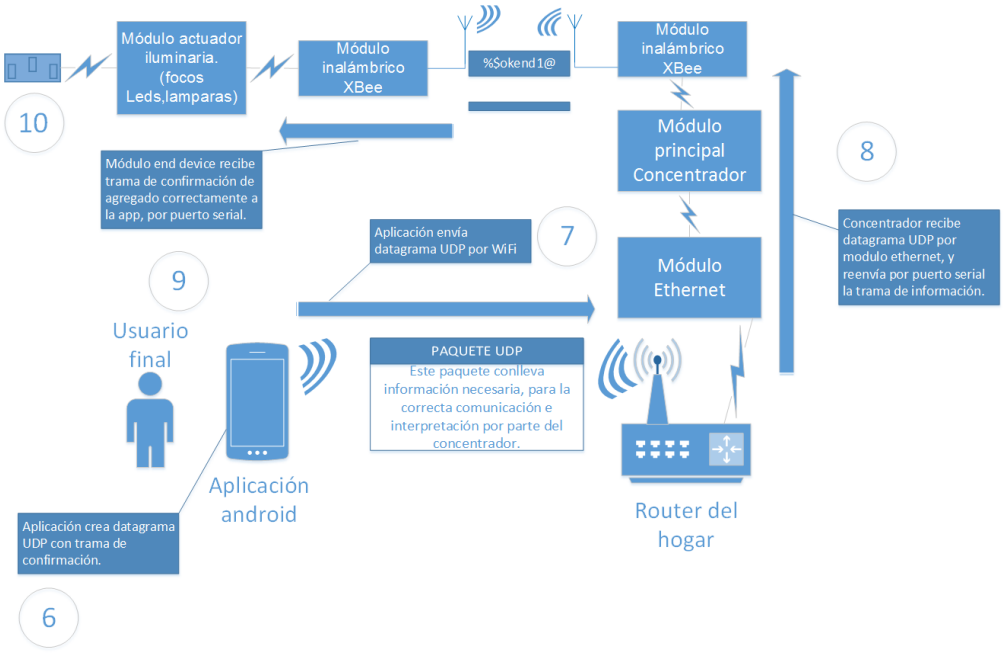


Figura E2 Diagrama secuencial de dispositivo final agregado (Parte 2)

En la Figura E2 una vez que el usuario recepta la petición, mediante una notificación de la aplicación, y procede a agregar el dispositivo, se envía una trama de confirmación al concentrador, el cual la reenvía al dispositivo final

mediante su módulo XBee. Cuando el controlador recibe la trama de confirmación cambia un registro de su memoria EPROM, esto para evitar que un dispositivo que ha sido agregado no vuelva a hacer peticiones de agregación.

Acción en un dispositivo final foco

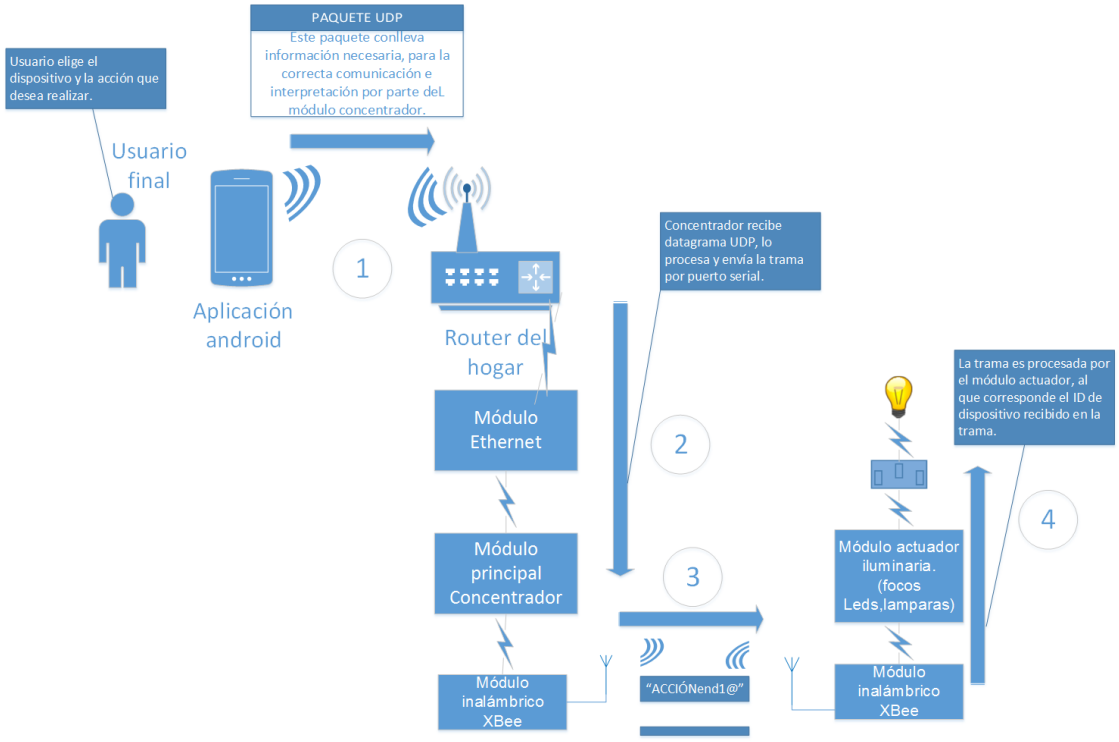


Figura E3 Diagrama de proceso de acción en un dispositivo final (foco)

En la Figura E3 se muestra a manera de un diagrama de secuencia funcional, el proceso para ejercer una acción sobre un dispositivo final, en este ejemplo, un foco. Se podrá notar que este proceso es de una sola vía, desde la aplicación

hacia el concentrador, mediante comunicación WIFI y desde el concentrador hacia el dispositivo final, mediante sus módulos XBee.

Cambio de intensidad en un dispositivo final foco

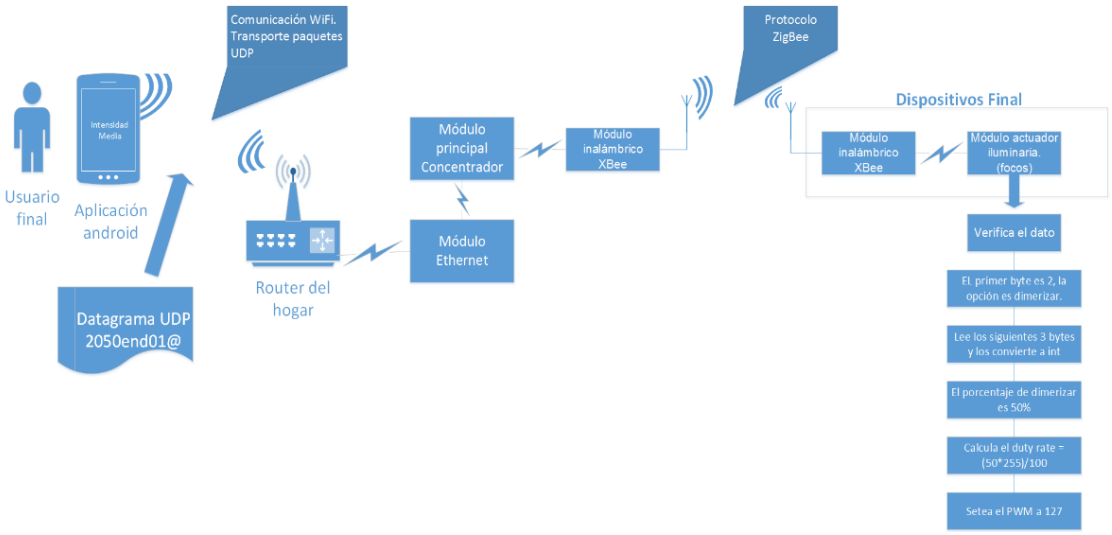


Figura E4 Diagrama de proceso para cambiar la intensidad de luz de un foco.

En la figura E4 se tiene de forma secuencial el procedimiento para cambiar la intensidad de un foco del sistema, la mayor diferencia con el proceso mostrado en la Figura E3 es el datagrama enviado.

Acción de lectura de sensor de temperatura

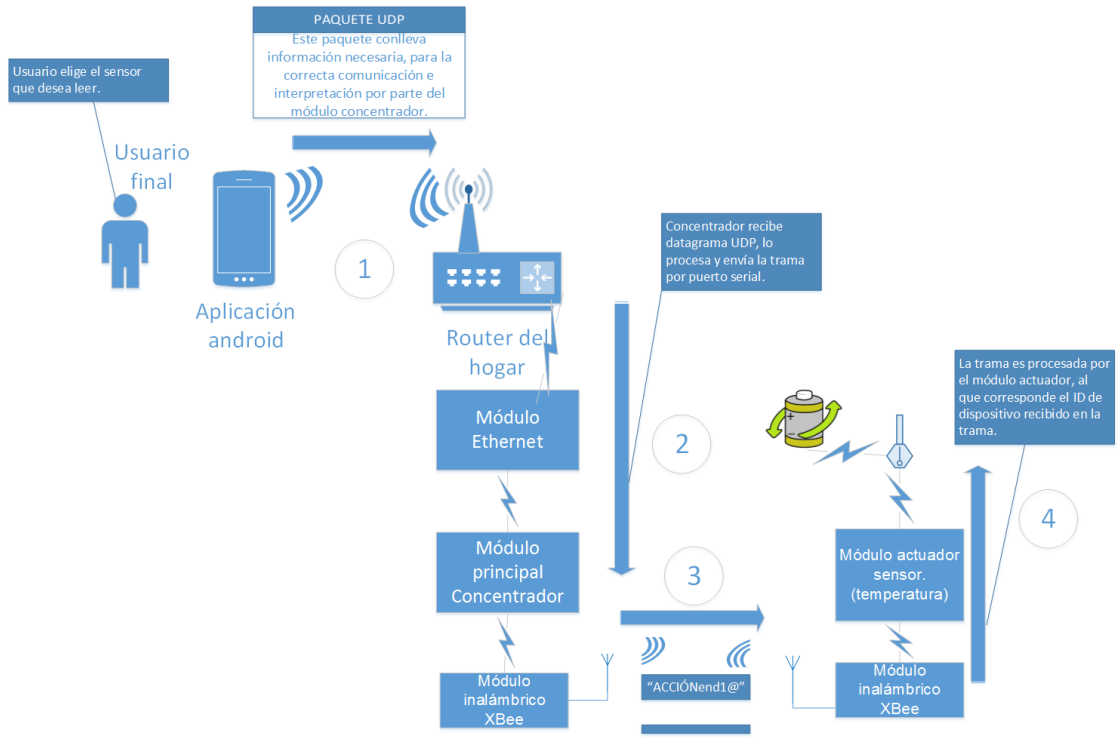


Figura E5 Diagrama de proceso de lectura de sensor (Parte 1)

En la figura E5 se muestra un diagrama de una parte de los procesos presentes, cuando el usuario desde la aplicación solicita la lectura de un sensor y esta solicitud llega hasta el concentrador mediante WIFI, que retransmite al dispositivo final, en este caso sensor de temperatura.

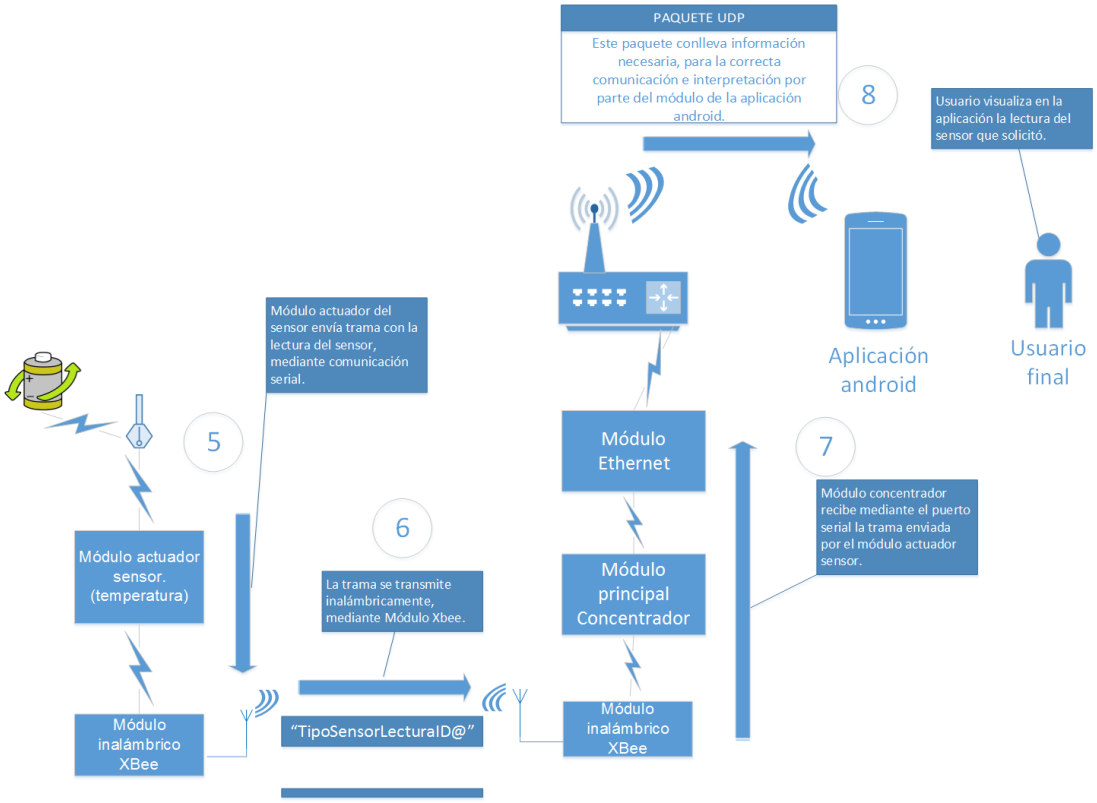


Figura E6 Diagrama de proceso de lectura de sensor (Parte 2)

En la figura E6 se muestra un diagrama de la parte en que el dispositivo sensor, responde la solicitud del usuario con la información que genera la lectura, esta información es receptada por el concentrador y retransmitida a la aplicación para que el usuario pueda verla.

ANEXO F

DIAGRAMAS DE FLUJO DE MÓDULOS DEL SISTEMA DOMÍTICO

Este anexo muestra los diagramas de flujo, que muestran los procesos que realizan cada uno de los módulos que conforman el sistema domótico planteado, estos son, módulo concentrador, módulo actuador y módulo sensor.

Diagrama de flujo de módulo concentrador

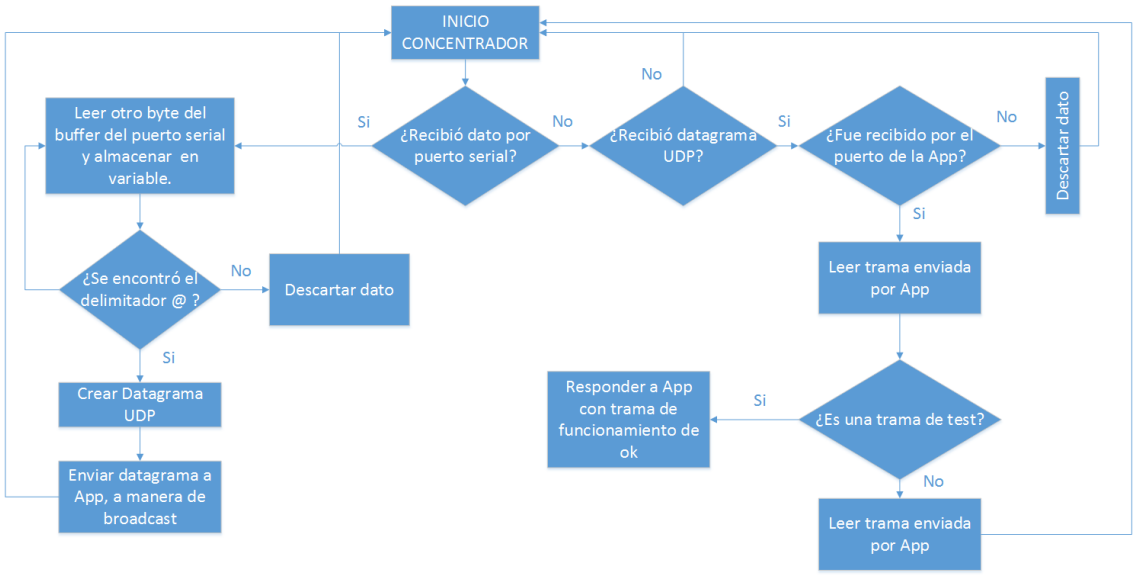


Figura F1 Diagrama de flujo del funcionamiento del módulo concentrador

Diagrama de flujo de módulo actuador

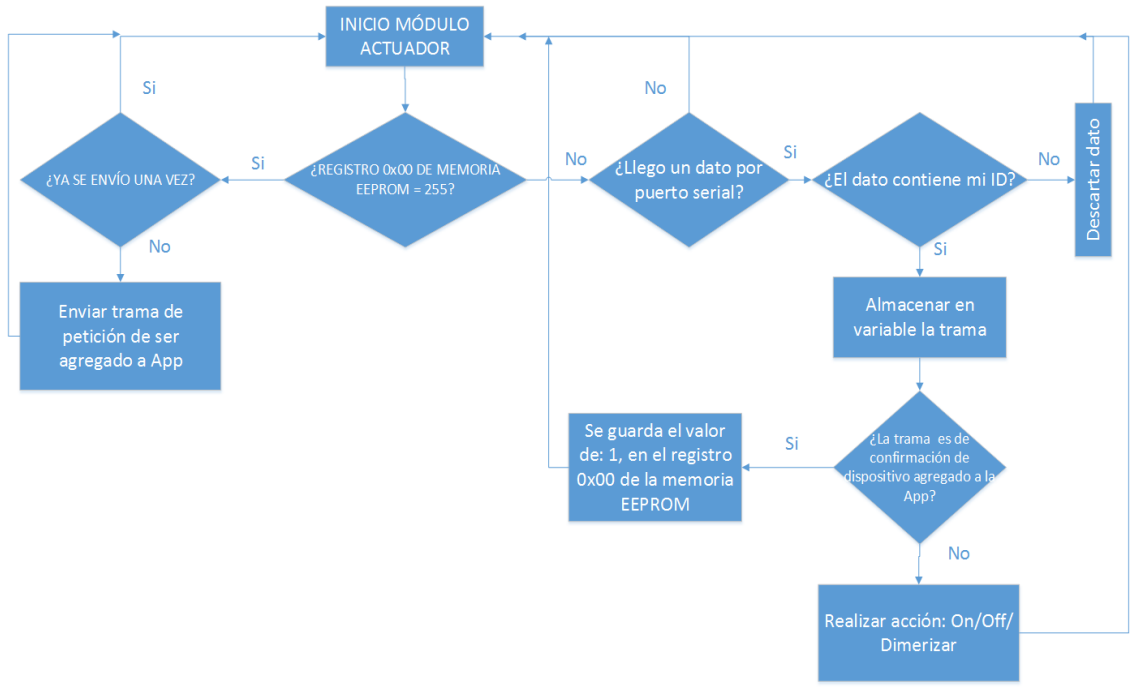


Figura F2 Diagrama de flujo de dispositivo final actuador

Diagrama de flujo de módulo sensor

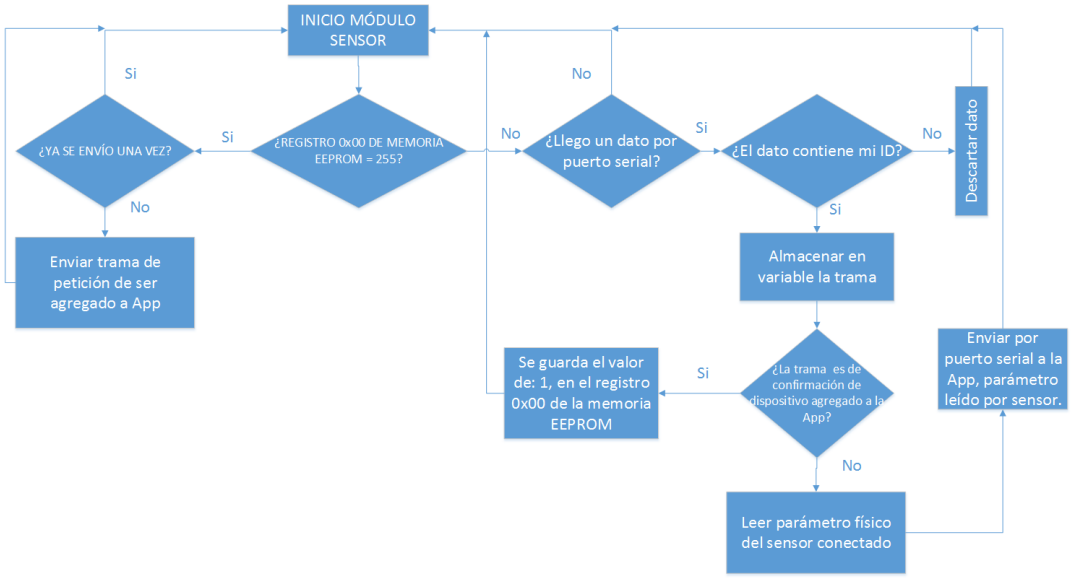


Figura F3 Diagrama de flujo del módulo sensor

ANEXO G

DISEÑO DE INTERFAZ DE USUARIO

Debido a la variedad de dispositivos con S.O. Android el diseño de la interfaz debe mejorar la presentación de los gráficos y el contenido de texto para pantallas con diferentes resoluciones y densidades.

A partir de Android 3.0, los elementos más usados para hacer una interfaz de usuario dinámica y de fácil navegación son:

- Barra de acciones.

  La barra de acciones es el destino de los íconos de acción que representan tareas que el usuario realiza con frecuencia, por este motivo se colocan en la parte superior de la barra de acciones, para que sean de fácil ubicación y acceso.

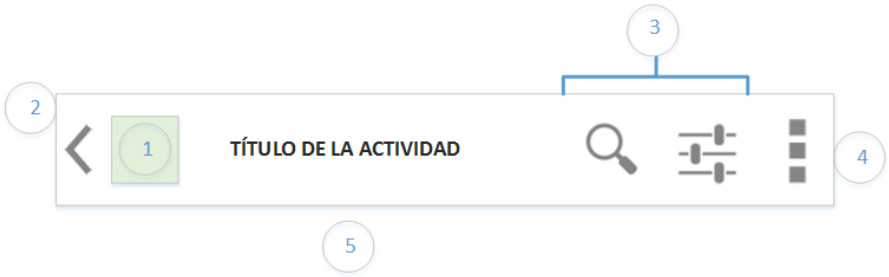


Figura G1 Esquema general de la barra de acciones de Android

1. El ícono de la aplicación debe representar las funciones de la misma, así como tener un diseño apreciable en pequeñas dimensiones.
2. En la barra de acciones se permite colocar el ícono de navegación hacia atrás, su función es regresar a la actividad previa a la actual, sin embargo esto puede ser modificado mediante programación.
3. Los botones de acción, funcionan de manera similar a las opciones de menú, incluso se declaran como menú. Sus funciones no se limitan a abrir nuevas actividades, también puede transformar la barra de acciones en un cajón para ingresar texto, que es el caso del botón buscar.

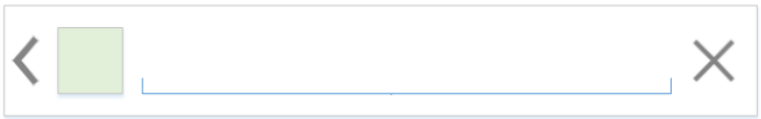


Figura G2 Esquema general de botón de acción, con transformación en la barra de acciones.

4. El ícono de Overflow es usado para que una vez presionado cargue un menú desplegable, se recomienda usar cuando éste menú se forma de pocos items para no sobrecargar la interfaz.

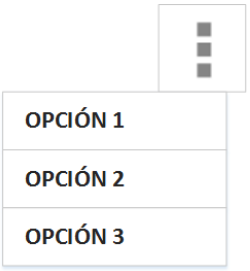


Figura G3 Esquema general de botón de acción overflow.

5. La forma en que se navegará por las distintas actividades de la aplicación, puede ser Tab Navigation, que se usa con pestanas, o con Spinner Navigation que carga una lista desplegable desde el nombre de la actividad.

Figura G4 Esquena de navegación con Spinner Navigation

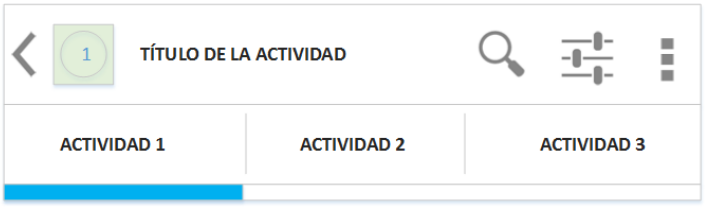


Figura G5 Esquena de navegación con Tab Navigation

- Pestañas con vistas deslizables.

  ViewPager es el elemento de Android que controla las vistas deslizables de las pestañas por lo tanto debe ser agregado en el layout de la actividad principal. Luego de ser creado necesitamos crear las vistas que incluirá ese ViewPager, por lo que necesitamos utilizar un PagerAdapter, para este fin Android ofrece dos alternativas:

  1. FragmentPagerAdapter: Es la clase más indicada cuando desee crear un número determinado de vistas.
  2. FragmentStatePagerAdapter: Es la clase más indicada cuando la cantidad de vistas es indeterminada, Android automáticamente destruirá fragmentos que no estén a la vista reduciendo el consumo de memoria.

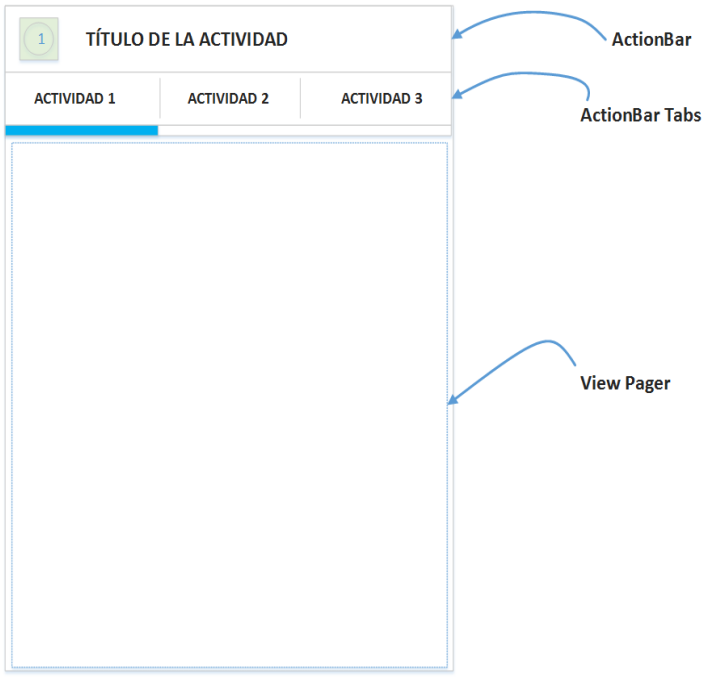


Figura G6 Estructura general de pestanas con ViewPager

ActionBar Tabs es habilitado en la actividad principal, usando la propiedad "setNavigationMode". La vista para cada pestaña necesita un archivo layout y una clase actividad, para poder ser cargado en el ViewPager.

- Fragmentos.

  Son vistas o porciones de interfaz de usuario que se integran a una actividad, son de gran utilidad ya que en una misma actividad se pueden usar varios fragmentos, así como en varias actividades reutilizar un

fragmento, lo que le permite al usuario no perder continuidad, lo que puede suceder cuando se genera un cambio de actividad.

Cada fragmento tiene sus propios eventos de entrada, su ciclo de vida, y puede ser agregado o eliminado mientras la actividad que lo acoge aún está en marcha.

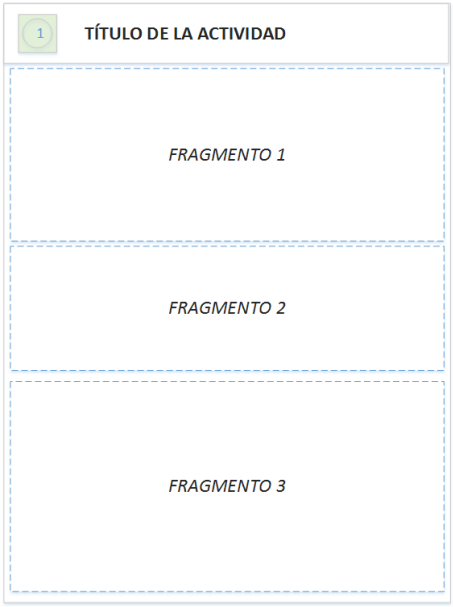


Figura G7 Estructura general de una actividad con fragmentos

En la Figura G7 se muestra un esquema de una actividad que contiene tres fragmentos, lo cual ayuda a reducir la cantidad de actividades de la aplicación, ya que el contenido en cada fragmento puede ser cambiado sin mayor dificultad.

- Diálogos.

  Son ventanas que se muestran delante de las actividades, y pueden recibir acciones del usuario, que generalmente se requieren antes de proceder con el siguiente paso. Si se requiere un diálogo diferente a los predefinidos por Android, entonces se puede extender de la clase Dialog y crear uno propio, pero esto debe evitarse. Se recomienda utilizar una de las siguientes subclases:

  1. AlertDialog: Permite mostrar al usuario un título, una lista con elementos seleccionable e incluso un diseño personalizado, y un máximo de tres botones para toma de decisión.

Figura G8 Estructura general de un Diálogo tipo AlertDialog

El ícono, el título y los botones son opcionales, pero lo más importante del diálogo es el área de contenido, ya que aquí se

mostrará la información necesaria para que el usuario tome una decisión.

2. DatePickerDialog: Permite al usuario seleccionar una fecha, mediante una interface predefinida.
3. TimePickerDialog: Permite al usuario seleccionar una hora, mediante una interface predefinida.

- Galería.

Una galería es una vista de una lista de elementos, los cuales se desplazan horizontalmente, tiene la característica de encerrar a la selección actual en el centro de la vista. Para llenar la galería se usa un adaptador, tal como lo hace ListView. Su estructura es la siguiente:

Figura G9 Estructura general de una galería

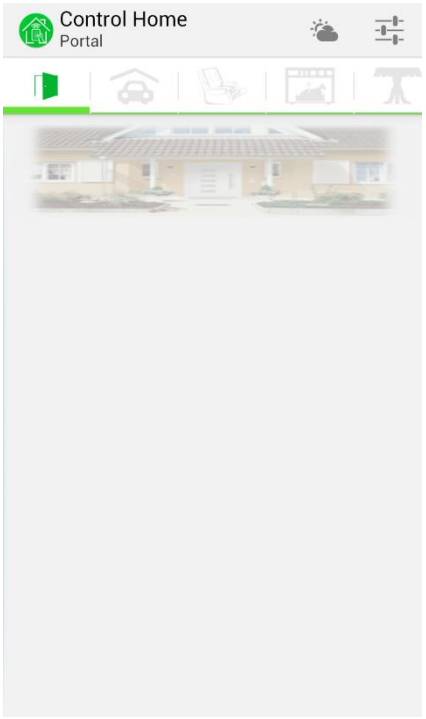


Figura G10 Vista de la actividad de ambientes

En la Figura G10 se puede observar el uso de pestañas y vistas deslizables, para que desde ahí el usuario tenga acceso a los diferentes ambientes de su hogar, sin tener cambios de pantalla, además se usaron íconos distintivos de cada ambiente, esto con la finalidad de no sobrecargar la interfaz con texto.

En la barra de acciones, se muestra como título el nombre de la aplicación y como subtítulo el nombre del ambiente en que se encuentra, esto es de gran utilidad, ya que evita confusiones en los ambientes.

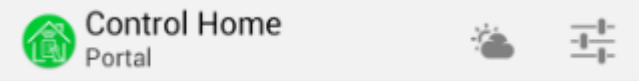


Figura G11 Barra de acciones de la actividad de ambientes

En la Figura G11 se puede ver el uso de botones en la barra de acciones, para lo cual se trató usar los íconos nativos de Android, especialmente para representar opciones de información muy generales o funciones esenciales, como es el caso de ajustes de una aplicación, esto debido a que el usuario ya está familiarizado con la acción que este ícono representa, y podrá tener acceso fácilmente a las por medio de la barra de acciones.

Figura G12 Vista de la lista de opciones de ajustes con privilegios

En la Figura G12 se muestra la vista de la actividad que se inicia una vez presionado el ícono de ajustes de la barra de acciones.

En la opción de seguridad se usó fragmentos para mostrar las opciones de configuración de seguridad y configuración de contraseña, esto para que el usuario no pierda la secuencia de que está en la opción de seguridad de la lista de opciones de Ajustes, para esto también es de gran ayuda el título y subtítulo de la barra de acciones.

Figura G13 Vista de cambio de contenido en fragmento de actividad de seguridad

En la Figura G13 se puede observar el cambio de contenido en el fragmento que se encuentra en la actividad de seguridad, el cual depende de la opción que elija el usuario.

Cuando un nuevo elemento es energizado en el hogar, para notificar este evento, al usuario le llega una notificación, la cual al ser presionada mostrará un diálogo tipo AlertDialog.

Figura G14 Diálogo para agregar un nuevo elemento en la aplicación

En la Figura G14 se podrá observar un Diálogo del tipo AlertDialog, el cual como contenido tiene dos galerías, una con imágenes de los ambientes del hogar y otra con imágenes representativas de él o los posibles nuevos elementos conectados al sistema. Además tendrá dos botones para tomar una decisión respecto al nuevo elemento.

ANEXO H

CÓDIGO JAVA Y XML DE APLICACIÓN ANDROIDE

En el presente anexo se detalla las clases y códigos xml para obtener la vista en la pantalla de usuario, con cada una de sus funciones.

Actividad de Tutorial: En esta sección detallaremos las principales clases y código XML, necesarias para visualizar el tutorial**.**

Código de layout principal activity_tutorial.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent"
   android:orientation="vertical"
   android:background="#FFFFFF">
   <LinearLayout
      android:layout_height="430dp"
      android:layout_width="fill_parent"
      android:orientation="vertical"
      android:background="#FFFFFF">
      <com.Tesis.DOMÓTICA.PageControl
      android:id="@+id/horizontal_pager"
      android:layout_width="fill_parent"
      android:layout_height="match_parent"
      android:layout_weight="0.1"
      android:background="#FFFFFF">
         <TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="Tutorial 1"
            android:textSize="24sp"
            android:textColor="#04B486"
            android:textStyle="bold"/>
         <TextView
```

```
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="Tutorial 2"
            android:textSize="24sp"
            android:textColor="#04B486"
            android:textStyle="bold"
            />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="Tutorial 3"
            android:textSize="24sp"
            android:textColor="#04B486"
            android:textStyle="bold"
            />
    </com.Tesis.DOMÓTICA.PageControl>


    </LinearLayout>

     <LinearLayout
          android:layout_height="fill_parent"
          android:layout_width="fill_parent"
          android:orientation="vertical"
          android:background="#FFFFFF"

          >
       <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Continuar"
            android:clickable="true"
            android:onClick="Continuar"
            android:layout_gravity="right"
            android:layout_weight="0.04"

android:textAppearance="@android:style/TextAppearance.DeviceDefault.Medium"
            />


        </LinearLayout>

</LinearLayout>
```

Código de TutorialActivity.java

```
public class TutorialActivity extends FragmentActivity {
    private MyAdapter mAdapter;
    private ViewPager mPager;
    PageIndicator mIndicator;
    private static final long ANIM_VIEWPAGER_DELAY = 5000;
    private static final long ANIM_VIEWPAGER_DELAY_USER_VIEW = 10000;
    private Runnable animateViewPager;
    private boolean stopSliding=false;
    private Handler handler;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if(getTuto()){
            if(getSeguridadAlta()){
                Intent conf = new Intent(TutorialActivity.this,ConfirmacionContrasena.class);
                startActivity(conf);
                finish();
            }
            else{
                Intent Partes = new
Intent(TutorialActivity.this,AmbientesFragmentActivity.class);
                startActivity(Partes);
                this.finish();
            }
        }

        else{
            setContentView(R.layout.principal_tutorial);
            mIndicator = (CirclePageIndicator)findViewById(R.id.indicator);
            mPager = (ViewPager) findViewById(R.id.pager);

            mAdapter = new MyAdapter(getSupportFragmentManager());
            mIndicator.setOnPageChangeListener(new PageChangeListener());
            mPager.setOnPageChangeListener(new PageChangeListener());

            mPager.setAdapter(mAdapter);
            mPager.setOnTouchListener(new OnTouchListener() {



                @Override
```

```
            public boolean onTouch(View v, MotionEvent event) {
                v.getParent().requestDisallowInterceptTouchEvent(true);
                switch (event.getAction()) {

                case MotionEvent.ACTION_CANCEL:
                    break;

                case MotionEvent.ACTION_UP:
                        stopSliding = false;
                        runnable(4);
                        handler.postDelayed(animateViewPager,
                                ANIM_VIEWPAGER_DELAY_USER_VIEW);

                    break;

                case MotionEvent.ACTION_MOVE:
                    if (handler != null && stopSliding == false) {
                        stopSliding = true;
                        handler.removeCallbacks(animateViewPager);
                    }
                    break;
                }
                return false;
            }
        });
    }
}


@Override
public void onResume() {

        mIndicator.setViewPager(mPager);

        runnable(4);
        handler.postDelayed(animateViewPager, ANIM_VIEWPAGER_DELAY);

    super.onResume();
}

public void runnable(final int size) {
    handler = new Handler();
    animateViewPager = new Runnable() {
        @Override
```

```
        public void run() {
            if (true) {
                if (mPager.getCurrentItem() == size - 1) {
                    mPager.setCurrentItem(0);
                } else {
                    mPager.setCurrentItem(
                            mPager.getCurrentItem() + 1, true);
                }
                handler.postDelayed(animateViewPager, ANIM_VIEWPAGER_DELAY);
            }
        }
    };
}


    private class PageChangeListener implements OnPageChangeListener {

        @Override
        public void onPageScrollStateChanged(int state) {
            if (state == ViewPager.SCROLL_STATE_IDLE) {

            }
        }

        @Override
        public void onPageScrolled(int arg0, float arg1, int arg2) {
        }

        @Override
        public void onPageSelected(int arg0) {
        }
    }

    public static class MyAdapter extends FragmentPagerAdapter {
        public MyAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public int getCount() {
            return 4;
        }

        @Override
```

```
        public Fragment getItem(int position) {
            switch (position) {
            case 0:
                return new Tutorial1Fragment();
            case 1:
                return new Tutorial2Fragment();
            case 2:
                return new Tutorial3Fragment();
            case 3:
                return new Tutorial4Fragment();
            default:
                return null;
            }
        }
    }

    public void finTutorial(View v){
        setTuto(true);
        if(!getAjustes()){
            Intent i= new Intent(this, AmbientesFragmentActivity.class);
            startActivity(i);
            this.finish();
        }
        else{
            this.finish();
        }
    }

    public void setTuto(Boolean valor){
        android.content.SharedPreferences settings =
getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean("eventoTuto", valor);
        editor.commit();
    }

    public boolean getTuto(){
        android.content.SharedPreferences settings =
getSharedPreferences("SETTINGS", 0);
        return settings.getBoolean("eventoTuto",false);
    }

    public boolean getSeguridadAlta(){
```

```
        android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
        return settings.getBoolean("SegAlta", false);
    }

    public boolean getAjustes(){
        android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
        return settings.getBoolean("eventoAjustes", false);
    }
}
```

Código del fragmento Tutorial1Fragment.java

El fragmento es utilizado en el adapter “MyAdapter”, que se encuentra declarado en la programación de la clase principal del tutorial. Esta clase que extiende de Fragment, tiene la misma estructura de programación que las siguientes páginas del tutorial”, por lo que sólo mostraremos la programación de la primera página.

```
public class Tutorial1Fragment extends Fragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.e("Test", "hello");
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.tutorial_uno, container, false);
```

```
        return view;
    }
}
```

Actividad de Ambientes

En esta sección detallaremos la programación necesaria, para que el usuario pueda navegar por los distintos ambientes del hogar, además de tener acceso a otras opciones.

Código de activity_page.xml

```
<android.support.v4.view.ViewPager
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/pager"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

</android.support.v4.view.ViewPager>
```

Actividad de ambientes

Podremos observar que en el siguiente código de la actividad, se crea una instancia del layout activity_page visto en el punto B.2.1.

```
public class AmbientesFragmentActivity extends FragmentActivity implements
    ActionBar.TabListener{
```

```
    private ViewPager viewPager;
    private TabsPagerAdapter mAdapter;
    private ActionBar actionBar;
    public int i=0;
    int pos=0;

    public  String[] TabImageListName =
{"Portal","Garage","Sala","Cocina","Comedor","Dormitorio","Jardin"};

    NotificationManager notificationManager;
    Notification myNotification;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_page);

        // Initilización
        viewPager = (ViewPager) findViewById(R.id.pager);
        actionBar = getActionBar();
        mAdapter = new TabsPagerAdapter(getSupportFragmentManager());
        viewPager.setAdapter(mAdapter);
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        actionBar.setSubtitle(TabImageListName[0]);

        Bundle bundle = getIntent().getExtras();
        if(!ServiceUDPListener.isRunning()){
            Intent in = new
Intent(AmbientesFragmentActivity.this,ServiceUDPListener.class);
            startService(in);
        }
        Intent serviceGraph = new Intent(AmbientesFragmentActivity.this,
ServiceGraph.class);
        startService(serviceGraph);

        if(bundle!=null){
        }
        else{

        }
        for (int i = 0;i<7;i++) {

            actionBar.addTab(actionBar.newTab()
                    .setTabListener(this)
                    .setIcon(Helper.TabImageList[i]));
```

```
    }

    /**
     * on swiping the viewpager make respective tab selected
     * */
    viewPager.setOnPageChangeListener(new ViewPager.OnPageChangeListener() {

        @Override
        public void onPageSelected(int position) {
            pos=position;
            actionBar.setSelectedNavigationItem(position);
            actionBar.setSubtitle(TabImageListName[position]);
        }

        @Override
        public void onPageScrolled(int arg0, float arg1, int arg2) {
        }

        @Override
        public void onPageScrollStateChanged(int arg0) {
        }
    });

    if(getEventoGlobal()){
        setEventoGlobal(false);
        actionBar.setSelectedNavigationItem(getAmbiente());
    }
}

@Override
public void onTabReselected(Tab tab, FragmentTransaction ft) {
}

@Override
public void onTabSelected(Tab tab, FragmentTransaction ft) {
    viewPager.setCurrentItem(tab.getPosition());
}

@Override
public void onTabUnselected(Tab tab, FragmentTransaction ft) {
}

 @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```

```
    getMenuInflater().inflate(R.menu.menu_conf, menu);
    return super.onCreateOptionsMenu(menu);
  }

public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.clima:
      Intent ic=new Intent(this, ConsultaClimaActivity.class);
      startActivity(ic);
      return true;
    case R.id.ajustes:
      Intent ia=new Intent(this, ListaOpcionesActivity.class);
      startActivity(ia);
      return true;
    default:
      return super.onOptionsItemSelected(item);
    }
  }

  private BroadcastReceiver receiver = new BroadcastReceiver() {

    @Override
    public void onReceive(Context context, Intent intent) {
      Bundle bundle = intent.getExtras();
      if (bundle != null) {

      }
    }
  };
  @Override
  protected void onResume() {
    super.onResume();

  }
  @Override
  protected void onPause() {
    super.onResume();
  }
  @Override
 protected void onDestroy()
 {
    super.onDestroy()
 }
```

```
    //MÉTODOS QUE AYUDARÁN A SABER SI OCURRIÓ UN EVENTO Y SETEAR LOS VALORES DE AMBIENTE Y DISPOSITIVO
    public void setAmbiente(int valor){

        android.content.SharedPreferences settings = getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putInt("amb_sel", valor);
        editor.commit();
    }

    public int getAmbiente(){

        android.content.SharedPreferences settings =
        getSharedPreferences("SETTINGS", 0);
        return settings.getInt("amb_sel", 0);

    }

    public void setDispositivo(int valor){

        android.content.SharedPreferences settings =
        getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putInt("disp_sel", valor);
        editor.commit();
    }

    public int getDispositivo(){

        android.content.SharedPreferences settings =
        getSharedPreferences("SETTINGS", 0);
        return settings.getInt("disp_sel", 0);

    }

    public void setEventoGlobal(Boolean valor){

        android.content.SharedPreferences settings =
        getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
      editor.putBoolean("eventoGlobal",valor);
        editor.commit();
    }
```

```
        public boolean getEventoGlobal(){

          android.content.SharedPreferences settings =
         getSharedPreferences("SETTINGS", 0);
          return settings.getBoolean("eventoGlobal",false);
        }

        public boolean getSeguridadAlta(){
          android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
          return settings.getBoolean("SegAlta", false);
      }
}
```

B.2.2. Adapter de pestañas

Este adapter es usado en la programación de la actividad de ambientes, a continuación veremos cuál es su función a nivel de programación.

```
public class TabsPagerAdapter extends FragmentPagerAdapter {

   public TabsPagerAdapter(FragmentManager fm) {
      super(fm);
   }

   @Override
   public Fragment getItem(int index) {

      switch (index) {
      case 0:
         // Portal fragment activity
         return new Portal();
      case 1:
         // Garage fragment activity
         return new Garage();
      case 2:
         // Sala fragment activity
         return new Sala();
```

```
            case 3:
                // Cocina fragment activity
                return new Cocina();

            case 4:
                // Comedor fragment activity
                return new Comedor();
            case 5:
                // Dormitorio fragment activity
                return new Dormitorio();
            case 6:
                // Jardin fragment activity
                return new Jardin();

            }

            return null;
        }

        @Override
        public int getCount() {
            // get item count - equal to number of tabs
            return 7;
        }

}
```

Fragmento de partes del hogar.

El siguiente código de la clase Sala.java mostrará cada una de las funciones que se realiza en las pestañas de la actividad de ambientes, ya que todas las partes de la casa tienen las mismas opciones y funciones.

```
public class Sala extends Fragment implements OnClickListener{
    public LinearLayout fila1;
    public LinearLayout fila2;
    public static final int PosAmb=2;
    private String trama;
```

```
    private String consumo;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {

        View rootView = inflater.inflate(R.layout.sala, container, false);
        fila1=(LinearLayout)rootView.findViewById(R.id.linearLayout1);
        fila2=(LinearLayout)rootView.findViewById(R.id.linearLayout2);
        if(getFondoAmb("Sala"))
        {
            LinearLayout fondo=(LinearLayout)rootView.findViewById(R.id.fondoSala);
            File sd = Environment.getExternalStorageDirectory();
            String path= sd.getAbsolutePath()+"/AmbientesTesis/Sala.png";
            Resources res = getResources();
            Bitmap bitmap = BitmapFactory.decodeFile(path);
            BitmapDrawable bd = new BitmapDrawable(res, bitmap);
            fondo.setBackgroundDrawable(bd);
        }
        else{
            LinearLayout fondo=(LinearLayout)rootView.findViewById(R.id.fondoSala);
            fondo.setBackgroundResource(R.drawable.ic_sala_fondo);
        }
        if(getEvento("eventoSala")){

            //agregar botón y guardar en la base de datos
            if(getYaTengoBoton())
            {
                Graficarboton();
            }
            if(getTotalBotonesPorFragment()<=4){
            ImageView b = new ImageView(this.getActivity());
                b.setLayoutParams(new
LayoutParams(android.view.ViewGroup.LayoutParams.WRAP_CONTENT,
android.view.ViewGroup.LayoutParams.WRAP_CONTENT,1));
                b.setId(getTotalBotonesPorFragment());
                b.setOnClickListener(this);
                b.setImageResource(Helper.elementos[getDispositivo()]);
                b.startAnimation(AnimationUtils.loadAnimation(this.getActivity(),
                        R.anim.zoom_forward_in));
                if(getTotalBotonesPorFragment()<=2){

                    fila1.addView(b);
                }
```

```
            else {
                fila2.addView(b);
            }
            DataBase db = new DataBase(getActivity().getApplicationContext());
            db.insertarBOTON(getTotalBotonesPorFragment(),
                    getTotalBotonesPorFragment(),
                    getAmbiente(),
                    getDispositivo(),
                    getTrama(),
                    getConsumo());
            setYaTengoBoton(true);
            setTotalBotonesPorFragment(getTotalBotonesPorFragment()+1);
        }
        else {
            Toast.makeText(getActivity(), "Ha completado los dispositivos",
Toast.LENGTH_LONG).show();
        }
        setEvento("eventoSala",false);
    }
    else if(getYaTengoBoton())
    {

        Graficarboton();

    }

    return rootView;
  }

  private void Graficarboton() {
    // TODO Auto-generated method stub
    DataBase db = new DataBase(getActivity().getApplicationContext());
    for(int i=1;i<getTotalBotonesPorFragment();i++){
        Botones bot= db.recuperarBoton(i,PosAmb);
        ImageView b = new ImageView(getActivity().getApplicationContext());

b.startAnimation(AnimationUtils.loadAnimation(this.getActivity(),R.anim.fade_in));
        b.setLayoutParams(new
LayoutParams(android.view.ViewGroup.LayoutParams.WRAP_CONTENT,
android.view.ViewGroup.LayoutParams.WRAP_CONTENT,1));
        b.setId(bot.getID());
        b.setImageResource(Helper.elementos[bot.getImagen()]);
        b.setOnClickListener(this);
        if(i<=2){
```

```
                fila1.addView(b);
            }
            else {
                fila2.addView(b);
            }
        }

    }

    @Override
    public void onClick(View v) {
        DataBase db = new DataBase(getActivity().getApplicationContext());
        Botones bot= db.recuperarBoton(v.getId(),PosAmb);
        setIDBoton(v.getId());
        trama=bot.getTrama();
        consumo=bot.getConsumo();
        switch (v.getId()) {
        case 1:
            v.startAnimation(AnimationUtils.loadAnimation(this.getActivity(),
R.anim.image_click));
            setDispositivo(bot.getImagen());
            setAmbiente(PosAmb);
            IrActivity(bot.getImagen());
            //saveAnswers();
            break;
        case 2:
            v.startAnimation(AnimationUtils.loadAnimation(this.getActivity(),
R.anim.image_click));
            setDispositivo(bot.getImagen());
            setAmbiente(PosAmb);
            IrActivity(bot.getImagen());
            break;
            //saveAnswers();

        case 3:
            v.startAnimation(AnimationUtils.loadAnimation(this.getActivity(),
R.anim.image_click));
            setDispositivo(bot.getImagen());
            setAmbiente(PosAmb);
            IrActivity(bot.getImagen());
            //saveAnswers();
            break;

        case 4:
```

```
            v.startAnimation(AnimationUtils.loadAnimation(this.getActivity(),
R.anim.image_click));
            setDispositivo(bot.getImagen());
            setAmbiente(PosAmb);
            IrActivity(bot.getImagen());
            //saveAnswers();
            break;
        }
    }
    //MÉTODOS QUE AYUDARÁN A SABER SI OCURRIO UN EVENTO Y SETEAR
LOS VALORES DE AMBIEN E IMAGEN
        public void setAmbiente(int valor){

            android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
            android.content.SharedPreferences.Editor editor = settings.edit();
            editor.putInt("amb_sel", valor);
            editor.commit();
        }

        public int getAmbiente(){

                android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
                return settings.getInt("amb_sel", 0);

        }

        public String getTrama(){

            android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS", 0);
            return settings.getString("trama_disp", "sin");

    }

        public void setDispositivo(int valor){

            android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
            android.content.SharedPreferences.Editor editor = settings.edit();
            editor.putInt("disp_sel", valor);
            editor.commit();
        }
```

```
    public int getDispositivo(){

        android.content.SharedPreferences settings =
this.getActivity().getSharedPreferences("SETTINGS",0);
        return settings.getInt("disp_sel", 0);

    }

    public void setIDBoton(int valor){

        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putInt("ID_bot", valor);
        editor.commit();
    }

    public int getIDBoton(){

        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        return settings.getInt("ID_bot", 0);

    }

    public void setEvento(String name,Boolean valor){

        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean(name, valor);
        editor.commit();
    }

    public boolean getEvento(String name){

        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        return settings.getBoolean(name, false);

    }

    public void setYaTengoBoton(boolean valor){
```

```
        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean("Sala", valor);
        editor.commit();

    }

    public boolean getYaTengoBoton(){

        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        return settings.getBoolean("Sala", false);
    }

    public void setYaSeAgrego(boolean valor){
        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean("YaSala", valor);
        editor.commit();
    }
    public boolean getYaSeAgrego(){
        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        return settings.getBoolean("YaSala", false);
    }
    public void setTotalBotonesPorFragment(int valor){

        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putInt("TotalButtonSala", valor);
        editor.commit();
    }

    public int getTotalBotonesPorFragment(){

        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS", 0);
        return settings.getInt("TotalButtonSala", 1);

    }
```

```
    public void IrActivity(int imagen){
        if(imagen==0||imagen==1||imagen==2)
        {
            Intent accion=new Intent(this.getActivity(),
AccionesGrupo2SensoresActivity.class);
            accion.putExtra("trama_sel", trama);
            accion.putExtra("consumo", consumo);
            startActivity(accion);
        }
        else if(imagen==3||imagen==4||imagen==5)
        {
            Intent accion=new Intent(this.getActivity(),
AccionesGrupoActuadoresActivity.class);
            accion.putExtra("trama_sel", trama);
            accion.putExtra("consumo", consumo);
            startActivity(accion);
        }
        else if(imagen==6)
        {
            Intent accion=new Intent(this.getActivity(), AccionesGrupo4Activity.class);
            accion.putExtra("trama_sel", trama);
            accion.putExtra("consumo", consumo);
            startActivity(accion);
        }
        else if(imagen==7)
        {
            Intent accion=new Intent(this.getActivity(), AccionesGrupo3Activity.class);
            accion.putExtra("trama_sel", trama);
            accion.putExtra("consumo", consumo);
            startActivity(accion);
        }
    }

    private boolean getFondoAmb(String selec)
    {
        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("ConfFondosAmbientes", 0);
        return settings.getBoolean("Fondo_"+selec, false);
    }
    public String getConsumo(){

        android.content.SharedPreferences settings =
getActivity().getSharedPreferences("SETTINGS", 0);
        return settings.getString("consumo_disp", "60");
```

```
    }
}
```

Actividad de consulta de clima

Debemos aclarar que la programación de esta clase implementa funciones de una librería proporcionada por el servidor de Yahoo, para consultas de clima.

```
public class ConsultaClimaActivity extends Activity implements
YahooWeatherInfoListener,
YahooWeatherExceptionListener {

private ImageView mIvWeather0,clima;
private TextView fecha,tclima,viento,humedad,temp;
private TextView mTvErrorMessage;
private TextView mTvTitle;
private View linea;
private YahooWeather mYahooWeather = YahooWeather.getInstance(5000, 5000,
true);
private ActionBar actionbar;
private ProgressDialog mProgressDialog;
private int temp_i;
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_clima_ch);
   actionbar=getActionBar();
   actionbar.setSubtitle("Información del Clima");

   mTvTitle = (TextView) findViewById(R.id.infociudad);
   mTvErrorMessage = (TextView) findViewById(R.id.textview_error_message);
   fecha= (TextView) findViewById(R.id.fecha);
   tclima= (TextView) findViewById(R.id.textclima);
   viento= (TextView) findViewById(R.id.textviento);
   humedad= (TextView) findViewById(R.id.texthumedad);
   temp= (TextView) findViewById(R.id.temptitle);
   linea= (View) findViewById(R.id.dividerView);
   mIvWeather0 = (ImageView) findViewById(R.id.imageView1);
   clima= (ImageView) findViewById(R.id.clima);
```

```
    searchByGPS();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate de menú; este agrega items en la barra de acción.
    getMenuInflater().inflate(R.menu.refresh, menu);
    return super.onCreateOptionsMenu(menu);
}

public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.clima_refresh:
        searchByGPS();
        return true;
    default:
        return super.onOptionsItemSelected(item);
    }
}
@Override
protected void onDestroy() {
    hideProgressDialog();
    mProgressDialog = null;
    super.onDestroy();
}

@Override
public void gotWeatherInfo(WeatherInfo weatherInfo) {
    hideProgressDialog();
    if (weatherInfo != null) {
        Log.i("clima", "Obtenido");
        setNormalLayout();
        mTvTitle.setText(
                weatherInfo.getWOEIDCounty() + " - "
                + weatherInfo.getWOEIDState() + ", "
                + weatherInfo.getWOEIDCountry());
        temp.setText(weatherInfo.getCurrentTempC() + "ºC");
        fecha.setText(weatherInfo.getCurrentConditionDate());
        tclima.setText(weatherInfo.getCurrentText());
        viento.setText(weatherInfo.getWindSpeed()+ "km/h");
        humedad.setText(weatherInfo.getAtmosphereHumidity()+ "%");

        temp_i= Integer.parseInt(temp.getText().subSequence(0,2).toString());
```

```
        Log.i("temp", String.valueOf(temp_i));
        if(temp_i<=0){
            linea.setBackgroundColor(getResources().getColor(R.color.celeste_claro));
            linea.invalidate();
        }
        else if(temp_i<=15 && temp_i>0){
            linea.setBackgroundColor(getResources().getColor(R.color.celeste_oscuro));
            linea.invalidate();
        }
        else if(temp_i<=25 && temp_i>15){
            linea.setBackgroundColor(getResources().getColor(R.color.amarillo));
            linea.invalidate();
        }
        else if(temp_i<=37 && temp_i>25){
            linea.setBackgroundColor(getResources().getColor(R.color.naranja));
            linea.invalidate();
        }
        else if(temp_i>37){
            linea.setBackgroundColor(getResources().getColor(R.color.rojo));
            linea.invalidate();
        }

        if (weatherInfo.getCurrentConditionIcon() != null) {
            mIvWeather0.setImageBitmap(weatherInfo.getCurrentConditionIcon());
            clima.setImageBitmap(weatherInfo.getCurrentConditionIcon());
        }
        else {
            setNoResultLayout();
        }
    }
}


@Override
}

@Override
public void onFailParsing(final Exception e) {
}

@Override
public void onFailFindLocation(final Exception e) {

}
```

```
private void setNormalLayout() {
    mTvTitle.setVisibility(View.VISIBLE);
    mTvErrorMessage.setVisibility(View.INVISIBLE);
}

private void setNoResultLayout() {
    mTvTitle.setVisibility(View.INVISIBLE);
    mTvErrorMessage.setVisibility(View.VISIBLE);
    mTvErrorMessage.setText("Disculpa, no se obtuvo resultados");
    mProgressDialog.cancel();
}

private void searchByGPS() {
    mYahooWeather.setExceptionListener(this);
    mProgressDialog = new ProgressDialog(this);
    mProgressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
    mProgressDialog.setTitle("Consultando clima");
    mProgressDialog.show();
    mYahooWeather.setNeedDownloadIcons(true);
    mYahooWeather.setSearchMode(SEARCH_MODE.GPS);
    mYahooWeather.queryYahooWeatherByGPS(getApplicationContext(), this);
}

private void searchByPlaceName(String location) {
    mYahooWeather.setNeedDownloadIcons(true);
    mYahooWeather.setSearchMode(SEARCH_MODE.PLACE_NAME);
    mYahooWeather.queryYahooWeatherByPlaceName(getApplicationContext(),
location, ConsultaClimaActivity.this);
}

private void showProgressDialog() {
    if (mProgressDialog != null && mProgressDialog.isShowing()) {
        mProgressDialog.cancel();
    }
    mProgressDialog = new ProgressDialog(ConsultaClimaActivity.this);
    mProgressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
    mProgressDialog.setTitle("Consultando cLima");
    mProgressDialog.show();
}

private void hideProgressDialog() {
    if (mProgressDialog != null && mProgressDialog.isShowing()) {
        mProgressDialog.cancel();
```

```
    }
}
}
```

Actividad de ajustes.

En esta actividad se le muestra al usuario una lista de opciones, ya sea de consulta, personalización, pruebas, etc. Es por eso el nombre de su clase ListaOpcionesActivity, cuya programación mostramos a continuación

```
public class ListaOpcionesActivity extends Activity {

    MyCustomAdapter dataAdapter = null;
    private ActionBar actionBar;
    public static Handler Handler;
    public String dato="";
    public static final String SERVERIP = "192.168.0.60";
    public static final int SERVERPORT = 5505;
    public static final int PuertoLocalRecepcion=5507;
    public static final int PuertoLocalenvio=5506;

     @Override
     public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.lista_opciones);
      actionBar = getActionBar();
      actionBar.setTitle("Ajustes");
      actionBar.setDisplayHomeAsUpEnabled(true);
      displayListView();

     }

     private void displayListView() {

        ArrayList<Opcion> opcionList = new ArrayList<Opcion>();
         Opcion opciones = new Opcion(R.drawable.ic_about,"Acerca
de",R.drawable.navigation_next_item);
         opcionList.add(opciones);
```

```
        opciones = new
Opcion(R.drawable.ic_tutorial,"Tutorial",R.drawable.navigation_next_item);
        opcionList.add(opciones);
        opciones = new Opcion(R.drawable.change_picture,"Fondos de
Ambiente",R.drawable.navigation_next_item);
        opcionList.add(opciones);
        opciones = new
Opcion(R.drawable.security,"Seguridad",R.drawable.navigation_next_item);
        opcionList.add(opciones);
        opciones = new Opcion(R.drawable.consumo,"Consumo
Energético",R.drawable.navigation_next_item);
        opcionList.add(opciones);
        opciones = new Opcion(R.drawable.chip,"Prueba
Conexión",R.drawable.navigation_next_item);
        opcionList.add(opciones);
        opciones = new Opcion(R.drawable.restart,"Reiniciar
Sistema",R.drawable.navigation_next_item);
        opcionList.add(opciones);

    //create an ArrayAdaptar from the String Array
    dataAdapter = new MyCustomAdapter(this,
      R.layout.configuraciones, opcionList);
    ListView listView = (ListView) findViewById(R.id.listView1);
    // Assign adapter to ListView
    listView.setAdapter(dataAdapter);


    listView.setOnItemClickListener(new OnItemClickListener() {
     public void onItemClick(AdapterView<?> parent, View view,
       int position, long id) {
      // When clicked, show a toast with the TextView text
      Opcion op = (Opcion) parent.getItemAtPosition(position);

       switch (position) {
          case 0:
             DialogoAcercaDe();
             break;
          case 1:
             setTuto(false);
             setAjustes(true);
             Intent OpcTuto= new Intent(ListaOpcionesActivity.this,TutorialActivity.class);
             startActivity(OpcTuto);
             break;
          case 2:
```

```
            Intent ChangeFondoAmb= new
Intent(ListaOpcionesActivity.this,FondosAmbientesActivity.class);
            startActivity(ChangeFondoAmb);
            overridePendingTransition(R.anim.left_in, R.anim.left_out);
            break;
          case 3:
            Intent OpcSeguridad= new
Intent(ListaOpcionesActivity.this,Seguridad_Acivity.class);
            startActivity(OpcSeguridad);
            break;
          case 4:
            Intent OpcGrafico= new
Intent(ListaOpcionesActivity.this,GraficoConsumoActivity.class);
            startActivity(OpcGrafico);
            break;
          case 5:
            IniciarCliente("%&isok?!@");
            break;
          }

      }
     });

    }

    private class MyCustomAdapter extends ArrayAdapter<Opcion> {

     private ArrayList<Opcion> opList;

     public MyCustomAdapter(Context context, int textViewResourceId,
       ArrayList<Opcion> opList) {
      super(context, textViewResourceId, opList);
      this.opList = new ArrayList<Opcion>();
      this.opList.addAll(opList);
     }

     private class ViewHolder {
      ImageView code;
      TextView name;
      ImageView other_ima;
     }

     @Override
     public View getView(int position, View convertView, ViewGroup parent) {
```

```
        ViewHolder holder = null;
        Log.v("ConvertView", String.valueOf(position));

        if (convertView == null) {
        LayoutInflater vi = (LayoutInflater)getSystemService(
          Context.LAYOUT_INFLATER_SERVICE);
        convertView = vi.inflate(R.layout.configuraciones, null);

        holder = new ViewHolder();
        holder.code = (ImageView) convertView.findViewById(R.id.imageView1);
        holder.name = (TextView) convertView.findViewById(R.id.textView1);
        holder.other_ima= (ImageView) convertView.findViewById(R.id.imageView2);
        convertView.setTag(holder);


        }
        else {
         holder = (ViewHolder) convertView.getTag();
        }
        Opcion op = opList.get(position);
        holder.code.setImageResource(op.getCode());
        holder.name.setText(op.getName());
        holder.other_ima.setImageResource(op.getOther());
        holder.name.setTag(op);

        return convertView;

      }
    }

     public void DialogoAcercaDe(){
        LayoutInflater li =  (LayoutInflater) ListaOpcionesActivity.this
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        final View prompt = li.inflate(R.layout.acerca_de, null);
        Button cerrar;

        Builder alertDialogBuilder = new AlertDialog.Builder(this);
        alertDialogBuilder.setView(prompt);
        alertDialogBuilder.setCancelable(true);
        alertDialogBuilder.setNegativeButton("Cerrar", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialogo1, int id) {
```

```
            dialogo1.cancel();
        }
    });
    AlertDialog alertDialog = alertDialogBuilder.create();
    alertDialog.show();
}


public void IniciarCliente(String comando)
  {
     new Thread(new Client(comando)).start();
     Handler = new Handler()
     {
        @Override
        public void handleMessage(Message msg)
        {

        }
     };
  }

     class Client implements Runnable {

        private String message;



        public Client(String comando) {
           dato = comando;
        }



        @Override
        public void run() {

           try {
                 Thread.sleep(500);
              } catch (InterruptedException e1) {
                 e1.printStackTrace();
              }
           try {
              InetAddress serverAddr = InetAddress.getByName(SERVERIP);
              DatagramSocket socket = new DatagramSocket(PuertoLocalenvio);
```

```
                byte[] buf;
                byte[] bufReplay;
                bufReplay = new byte[6];
                buf=dato.getBytes();
                DatagramPacket packet = new DatagramPacket(buf, buf.length,
serverAddr, SERVERPORT);
                DatagramPacket packetResponsePic = new
                DatagramPacket(bufReplay, bufReplay.length);
                socket.send(packet);
                socket.receive(packetResponsePic);
                 message = new String(packetResponsePic.getData());
                  Log.i("UDP","Se envio el paquete: "+dato + "y se recibio de replay: "+
                message);
                 runOnUiThread(new Runnable() {

                    @Override
                    public void run() {
                       if(message.equals("PIC_OK")){
                         Toast.makeText(getApplicationContext(), "El controlador funciona
                         correctamente", Toast.LENGTH_LONG).show();
                       }

                    }
                });


                socket.disconnect();
                socket.close();
            } catch (Exception e)
            {

                runOnUiThread(new Runnable() {

                    @Override
                    public void run() {

                Toast.makeText(getApplicationContext(), "Existe un error con el
                controlador", Toast.LENGTH_LONG).show();

                    }
                });
            }
            }
```

```
    }

    public void setTuto(Boolean valor){
        android.content.SharedPreferences settings =
getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean("eventoTuto", valor);
        editor.commit();
    }

    public void setAjustes(Boolean valor){
        android.content.SharedPreferences settings =
getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean("eventoAjustes", valor);
        editor.commit();
    }
}
```

Actividad de acciones del grupo de actuadores.

Esta actividad es la misma para un grupo de elementos del hogar, que son: foco, persiana, regadera. Han sido agrupados en la misma actividad debido a que las acciones que realizan son similares entre sí, además de que todos permiten programar una hora de inicio y rango de tiempo de ejecución de activación o desactivación.

```
public class AccionesGrupoActuadoresActivity extends Activity implements IRefresh {
    private RadioGroup opcIntensidad;
    private RadioButton op1,op2,op3;
    private ImageView elemento;
    private Switch swOn_Off;
    private String ID_trama;
    private String comando;
    private String consumoCurrentDisp;
    private int ValorCurrentConsumo;
```

```
private TextView desc;
Bitmap bmp;
private String elementoID;

private MockRefreshTask mTask;

private MenuItem mRefreshMenuItem;
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.acciones_grupo_uno_activity);
    Bundle b = getIntent().getExtras();
    ID_trama=b.getString("trama_sel");
    consumoCurrentDisp=b.getString("consumo");
    ValorCurrentConsumo=Integer.parseInt(consumoCurrentDisp);
    elementoID="elemento".concat(ID_trama);
    desc=(TextView)findViewById(R.id.textView1);
    opcIntensidad=(RadioGroup)findViewById(R.id.opcIntensidad);
    op1=(RadioButton)findViewById(R.id.radiobajo);
    op2=(RadioButton)findViewById(R.id.radiomedio);
    op3=(RadioButton)findViewById(R.id.radioalto);
    swOn_Off = (Switch)findViewById(R.id.switchOnOff);
    elemento = (ImageView)findViewById(R.id.element);

    if(getDispositivo()==3){//foco
        if(getEstado(elementoID)==0){
            elemento.setImageResource(R.drawable.foco_g);
        }
        else if(getEstado(elementoID)==1){
            swOn_Off.setChecked(true);
            op1.setChecked(true);
            elemento.setImageResource(R.drawable.foco_ab);
        }
        else if(getEstado(elementoID)==2){
            swOn_Off.setChecked(true);
            op2.setChecked(true);
            elemento.setImageResource(R.drawable.foco_am);
        }
        else if(getEstado(elementoID)==3||getEstado(elementoID)==4){
            swOn_Off.setChecked(true);
            op3.setChecked(true);
            elemento.setImageResource(R.drawable.foco_a);
        }
```

```
}
else if(getDispositivo()==4){//persiana
    desc.setText("Luminosidad");
    if(getEstado(elementoID)==0){
        elemento.setImageResource(R.drawable.persiana_c);
    }
    else if(getEstado(elementoID)==1){
        swOn_Off.setChecked(true);
        op1.setChecked(true);
        elemento.setImageResource(R.drawable.persiana_sa);
    }
    else if(getEstado(elementoID)==2){
        swOn_Off.setChecked(true);
        op2.setChecked(true);
        elemento.setImageResource(R.drawable.persiana_sa);
    }
    else if(getEstado(elementoID)==3||getEstado(elementoID)==4){
        swOn_Off.setChecked(true);
        op3.setChecked(true);
        elemento.setImageResource(R.drawable.persiana_a);
    }
}
else if(getDispositivo()==5){//regadera
    desc.setText("Caudal");
    op1.setText("Bajo");
    op2.setText("Medio");
    op3.setText("Alto");
    if(getEstado(elementoID)==0){
        elemento.setImageResource(R.drawable.regadera_g);
    }
    else if(getEstado(elementoID)==1){
        swOn_Off.setChecked(true);
        op1.setChecked(true);
        elemento.setImageResource(R.drawable.regadera_b);
    }
    else if(getEstado(elementoID)==2){
        swOn_Off.setChecked(true);
        op2.setChecked(true);
        elemento.setImageResource(R.drawable.regadera_m);
    }
    else if(getEstado(elementoID)==3||getEstado(elementoID)==4){
        swOn_Off.setChecked(true);
        op3.setChecked(true);
        elemento.setImageResource(R.drawable.regadera_a);
```

```
        }
    }
    swOn_Off.setOnCheckedChangeListener(new
Switch.OnCheckedChangeListener() {

        public void onCheckedChanged(CompoundButton buttonView,
          boolean isChecked) {

        if(isChecked){
            setEstadoSW(1);
            setCurrentConsumo(getCurrentConsumo()+ValorCurrentConsumo);
            if(getEstado(elementoID)==1){
                comando=("2001end").concat(ID_trama).concat("@");
                op1.setChecked(true);
                op2.setChecked(false);
                op3.setChecked(false);
            }
            else if (getEstado(elementoID)==2){
                comando=("2010end").concat(ID_trama).concat("@");
                op1.setChecked(false);
                op2.setChecked(true);
                op3.setChecked(false);
            }
            else if (getEstado(elementoID)==3){
                comando=("2030end").concat(ID_trama).concat("@");
                op1.setChecked(false);
                op2.setChecked(false);
                op3.setChecked(true);
            }
            else{
                setEstado(elementoID, 4);
                comando=("1000end").concat(ID_trama).concat("@");
            }
            if(getDispositivo()==3){
                elemento.setImageResource(R.drawable.foco_a);
            }
            else if(getDispositivo()==4){
                elemento.setImageResource(R.drawable.persiana_a);
            }
            else if(getDispositivo()==5){
                elemento.setImageResource(R.drawable.regadera_a);
            }
            opcIntensidad.setEnabled(false);
            swOn_Off.setEnabled(false);
```

```
                swOn_Off.setChecked(true);

                mTask = new MockRefreshTask(AccionesGrupoActuadoresActivity.this);
                mTask.execute();
                swOn_Off.setText("Encendido");
                //AÑADIR MÉTODOS PARA ENVIAR TRAMA...
                EnviarDatoUDP envioON = new EnviarDatoUDP();
                envioON.IniciarClienteUDP(comando);
                Toast.makeText(getApplicationContext(), comando,
Toast.LENGTH_LONG).show();
            }else if(!isChecked){
                setEstadoSW(2);
                setCurrentConsumo(getCurrentConsumo()-ValorCurrentConsumo);
                op1.setChecked(false);
                op2.setChecked(false);
                op3.setChecked(false);
                swOn_Off.setChecked(false);
                setEstado(elementoID,0);
                if(getDispositivo()==3){
                    elemento.setImageResource(R.drawable.foco_g);
                }
                else if(getDispositivo()==4){
                    elemento.setImageResource(R.drawable.persiana_c);
                }
                else if(getDispositivo()==5){
                    elemento.setImageResource(R.drawable.regadera_g);
                }
                mTask = new MockRefreshTask(AccionesGrupoActuadoresActivity.this);
                opcIntensidad.setEnabled(false);
                mTask.execute();
                swOn_Off.setText("Apagado");
                comando=("0000end").concat(ID_trama).concat("@");
                //AÑADIR MÉTODOS PARA ENVIAR TRAMA...
                EnviarDatoUDP envioOFF = new EnviarDatoUDP();
                envioOFF.IniciarClienteUDP(comando);
                Toast.makeText(getApplicationContext(), comando,
Toast.LENGTH_LONG).show();
            }

            }


        });
```

```
        opcIntensidad.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener(){

                public void onCheckedChanged(RadioGroup group, int checkedId) {

                  if(getEstadoSW()==2){
                     if(checkedId==R.id.radiobajo){
                        setEstado(elementoID,1);

                        if(getDispositivo()==3){//foco
                             elemento.setImageResource(R.drawable.foco_ab);
                        }
                        else if(getDispositivo()==4){//persiana
                              elemento.setImageResource(R.drawable.persiana_c);
                       }
                        else if(getDispositivo()==5){//regadera
                           elemento.setImageResource(R.drawable.regadera_b);
                        }
                        swOn_Off.setChecked(true);
                     }
                     if(checkedId==R.id.radiomedio){
                        setEstado(elementoID,2);
                        if(getDispositivo()==3)
                        {//foco
                           elemento.setImageResource(R.drawable.foco_am);
                        }
                        else if(getDispositivo()==4)
                        {//persiana
                           elemento.setImageResource(R.drawable.persiana_sa);
                        }
                        else if(getDispositivo()==5){//regadera
                           elemento.setImageResource(R.drawable.regadera_m);
                        }
                        swOn_Off.setChecked(true);
                     }
                     if(checkedId==R.id.radioalto){
                        setEstado(elementoID,3);
                        if(getDispositivo()==3){//foco
                           elemento.setImageResource(R.drawable.foco_a);
                        }
                        else if(getDispositivo()==4){//persiana
                           elemento.setImageResource(R.drawable.persiana_a);
                       }
                        else if(getDispositivo()==5){//regadera
```

```
                    elemento.setImageResource(R.drawable.regadera_a);
                }
                swOn_Off.setChecked(true);
            }
        }
        else if(getEstadoSW()==1)
        {
            setCurrentConsumo(getCurrentConsumo()+ValorCurrentConsumo);

            if(checkedId==R.id.radiobajo){
                setEstado(elementoID,1);
                comando=("2001end").concat(ID_trama).concat("@");
                op1.setChecked(true);
                op2.setChecked(false);
                op3.setChecked(false);
                if(getDispositivo()==3){//foco
                     elemento.setImageResource(R.drawable.foco_ab);
                }
                else if(getDispositivo()==4){//persiana
                    elemento.setImageResource(R.drawable.persiana_c);
                }
                else if(getDispositivo()==5){//regadera
                    elemento.setImageResource(R.drawable.regadera_b);
                }
                swOn_Off.setChecked(true);
                mTask = new
MockRefreshTask(AccionesGrupoActuadoresActivity.this);
                EnviarDatoUDP envioON = new EnviarDatoUDP();
                envioON.IniciarClienteUDP(comando);
                Toast.makeText(getApplicationContext(), comando,
Toast.LENGTH_LONG).show();
                mTask.execute();
            }
            if(checkedId==R.id.radiomedio){
                setEstado(elementoID,2);
                comando=("2010end").concat(ID_trama).concat("@");
                op1.setChecked(false);
                op2.setChecked(true);
                op3.setChecked(false);
                if(getDispositivo()==3)
                {//foco
                    elemento.setImageResource(R.drawable.foco_am);
                }
                else if(getDispositivo()==4)
```

```
                    {//persiana
                        elemento.setImageResource(R.drawable.persiana_sa);
                    }
                    else if(getDispositivo()==5){//regadera
                        elemento.setImageResource(R.drawable.regadera_m);
                    }
                    swOn_Off.setChecked(true);
                    mTask = new
MockRefreshTask(AccionesGrupoActuadoresActivity.this);
                    EnviarDatoUDP envioON = new EnviarDatoUDP();
                    envioON.IniciarClienteUDP(comando);
                    Toast.makeText(getApplicationContext(), comando,
Toast.LENGTH_LONG).show();
                    mTask.execute();
                }
                if(checkedId==R.id.radioalto){
                    setEstado(elementoID,3);
                    comando=("2030end").concat(ID_trama).concat("@");
                    op1.setChecked(false);
                    op2.setChecked(false);
                    op3.setChecked(true);
                    if(getDispositivo()==3){//foco
                        elemento.setImageResource(R.drawable.foco_a);
                    }
                    else if(getDispositivo()==4){//persiana
                        elemento.setImageResource(R.drawable.persiana_a);
                    }
                    else if(getDispositivo()==5){//regadera
                        elemento.setImageResource(R.drawable.regadera_a);
                    }
                    swOn_Off.setChecked(true);
                    mTask = new
MockRefreshTask(AccionesGrupoActuadoresActivity.this);
                    EnviarDatoUDP envioON = new EnviarDatoUDP();
                    envioON.IniciarClienteUDP(comando);
                    Toast.makeText(getApplicationContext(), comando,
Toast.LENGTH_LONG).show();
                    mTask.execute();
                }
            }
        }
```

```
    });


    }

    //MÉTODOS PARA ELEGIR LA IMAGEN DEL ELEMENTO SELECCIONADO Y
HACER EL QUERY DE LA TRAMA A ENVIAR
    public int getAmbiente(){

            android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
            return settings.getInt("amb_sel", 0);

    }


    public int getDispositivo(){

            android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
            return settings.getInt("disp_sel", 0);

    }



    public void setEstado(String elementoID, int estado){
        android.content.SharedPreferences settings =
getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putInt(elementoID,estado);
        editor.commit();
    }

    public int getEstado(String elementoID){
        android.content.SharedPreferences settings =
getSharedPreferences("SETTINGS", 0);
        return settings.getInt(elementoID, 0);
    }

    public void setEstadoSW(int estado){
        android.content.SharedPreferences settings =
getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
```

```
        editor.putInt("estadoSW",estado);
        editor.commit();
    }

    public int getEstadoSW(){
        android.content.SharedPreferences settings =
getSharedPreferences("SETTINGS", 0);
        return settings.getInt("estadoSW", 2);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.action_bar, menu);

        mRefreshMenuItem = menu.findItem(R.id.action_refresh);

        return super.onCreateOptionsMenu(menu);

    }
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.program:
            Intent i =new Intent(this, AccionesProgramadasActivity.class);
            startActivity(i);
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }
    }



    @Override
    public void setRefresh(boolean refresh) {

        if (refresh){
            mRefreshMenuItem.setActionView(R.layout.actionbar_indeterminate_progress);
        }else{
            mRefreshMenuItem.setActionView(null);
            mTask = null;
        }
    }
```

```
@Override
protected void onDestroy() {

    super.onDestroy();

    if (mTask != null){
        mTask.cancel(true);
    }
}

public class MockRefreshTask extends AsyncTask<Void, Void, Void> {

    private IRefresh refreshImpl;

    public MockRefreshTask(IRefresh ref) {
        refreshImpl = ref;
    }

    @Override
    protected void onPreExecute() {
        refreshImpl.setRefresh(true);
    }

    @Override
    protected Void doInBackground(Void... arg0) {
        try {
            Thread.sleep(3000);

        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        refreshImpl.setRefresh(false);
        runOnUiThread(new Runnable() {
            public void run() {
                op1.setEnabled(true);
                op2.setEnabled(true);
```

```
                op3.setEnabled(true);

                swOn_Off.setEnabled(true);
            }
        });
    }
}

    public void setCurrentConsumo(int valor){

        android.content.SharedPreferences settings =
getSharedPreferences("CONSUMO",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putInt("current_consumo", valor);
        editor.commit();
    }

    public int getCurrentConsumo(){

        android.content.SharedPreferences settings =
getSharedPreferences("CONSUMO",0);
        return settings.getInt("current_consumo", 0);
    }

}
```

B.6. Actividad de acciones programadas.

El acceso a esta actividad se encuentra en la barra de acciones de la actividad de actuadores, y se encarga de programar acciones para los elementos del hogar de esa actividad.

```
public class AccionesProgramadasActivity extends Activity {

    private PendingIntent pendingIntent,pendingIntent2;
    private TimePicker timepick;
    private EditText intervalo;
    private TextView ver;
```

```
    private Spinner acciones;
    private RadioGroup accion_sel;
    private int ID_alarmaHA;
    private int ID_alarmaTA;
    private int duracion;
    private int hour;
    private int minute;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_programacion);
        accion_sel=(RadioGroup)findViewById(R.id.radioGroup1);

ID_alarmaHA=Integer.parseInt(String.valueOf(getIDBoton()).concat(String.valueOf(getD
ispositivo())).concat(String.valueOf(getAmbiente()).concat("1")));

ID_alarmaTA=Integer.parseInt(String.valueOf(getIDBoton()).concat(String.valueOf(getDi
spositivo())).concat(String.valueOf(getAmbiente()).concat("0")));
        Intent onIntent = new Intent(this, TramaActivacionBroadcast.class);
        onIntent.putExtra("alarma",String.valueOf(ID_alarmaHA));
        Toast.makeText(this.getApplicationContext(),""+ID_alarmaHA,
                Toast.LENGTH_LONG).show();
        Intent offIntent = new Intent(this, TramaDesactivacionBroadcast.class);
        offIntent.putExtra("alarma_des",String.valueOf(ID_alarmaTA));
        /*PendingIntent permite a otra aplicación usar los permisos de nuestra aplicación
        para ejecutar trozos de códigos predefinidos.*/
        pendingIntent = PendingIntent.getBroadcast(this.getApplicationContext(),
ID_alarmaHA, onIntent, 0);
        pendingIntent2 = PendingIntent.getBroadcast(this.getApplicationContext(),
ID_alarmaTA, offIntent, 0);
        timepick=(TimePicker)findViewById(R.id.timePicker1);
        intervalo=(EditText)findViewById(R.id.interval);
        acciones=(Spinner)findViewById(R.id.spinner1);
        ArrayAdapter spinner_adapter = ArrayAdapter.createFromResource( this,
R.array.acciones , android.R.layout.simple_spinner_item);

spinner_adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdow
n_item);
        acciones.setAdapter(spinner_adapter);
        findViewById(R.id.start).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(intervalo.length()!=0){
                    start();
```

```
            }
            else{
                Toast.makeText(getApplicationContext(), "Por favor ingrese tiempo de
ejecución", Toast.LENGTH_LONG).show();
            }
        }
    });
}

    public void start() {
        AlarmManager manager = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        AlarmManager manager2 = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        int interval = 30000;
        int tipo = (int) acciones.getSelectedItemId();
        duracion= Integer.parseInt(intervalo.getText().toString());
        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.HOUR_OF_DAY,timepick.getCurrentHour());
        calendar.set(Calendar.MINUTE, timepick.getCurrentMinute());
        calendar.set(Calendar.SECOND,0);
        calendar.set(Calendar.MILLISECOND,0);
        setCalendarvalue(HoraFin(duracion, tipo, timepick));
        manager.set(AlarmManager.RTC_WAKEUP,
calendar.getTimeInMillis(),pendingIntent);
        manager2.set(AlarmManager.RTC_WAKEUP,getCalendarvalue(),pendingIntent2);
    }

    public void cancel() {
        AlarmManager manager = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        Calendar calendar = Calendar.getInstance();
        manager.cancel(pendingIntent);
        Toast.makeText(this, "Alarm Canceled", Toast.LENGTH_SHORT).show();
    }

    private TimePickerDialog.OnTimeSetListener timePickerListener = new
TimePickerDialog.OnTimeSetListener() {
        public void onTimeSet(TimePicker view, int selectedHour,
                int selectedMinute) {
            hour = selectedHour;
            minute = selectedMinute;

            // set current time into timepicker
```

```
            timepick.setCurrentHour(hour);
            timepick.setCurrentMinute(minute);

        }
    };

    public void setCalendarvalue(long valor){

        android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
        android.content.SharedPreferences.Editor editor = settings.edit();
        editor.putLong("calendar", valor);
        editor.commit();
    }

    public long getCalendarvalue(){

        android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS", 0);
        return settings.getLong("calendar",0);

    }

    public long HoraFin(int cantidad,int tipo, TimePicker timepick){
        Calendar calendar = Calendar.getInstance();
        int new_hour=0;
        int new_min=0;
        if(tipo==0){
            new_hour=timepick.getCurrentHour()+cantidad;
            if(new_hour>=24){
                new_hour=new_hour%24;
                calendar.set(Calendar.HOUR_OF_DAY,new_hour);
                calendar.set(Calendar.MINUTE, timepick.getCurrentMinute());
                calendar.set(Calendar.SECOND,0);
                calendar.set(Calendar.MILLISECOND,0);
            }
            else{
                calendar.set(Calendar.HOUR_OF_DAY,new_hour);
                calendar.set(Calendar.MINUTE, timepick.getCurrentMinute());
                calendar.set(Calendar.SECOND,0);
                calendar.set(Calendar.MILLISECOND,0);
            }
        }
```

```
        if(tipo==1){
            new_min=timepick.getCurrentMinute()+cantidad;
            if (new_min>=60){
                new_hour=new_min/60;
                new_min=new_min%60;
                calendar.set(Calendar.HOUR_OF_DAY,new_hour);
                calendar.set(Calendar.MINUTE,new_min);
                calendar.set(Calendar.SECOND,0);
                calendar.set(Calendar.MILLISECOND,0);
            }
            else{
                calendar.set(Calendar.HOUR_OF_DAY,timepick.getCurrentHour());
                calendar.set(Calendar.MINUTE, new_min);
                calendar.set(Calendar.SECOND,0);
                calendar.set(Calendar.MILLISECOND,0);
            }
        }
        return calendar.getTimeInMillis();
    }

    public int getDispositivo(){

        android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
        return settings.getInt("disp_sel", 0);
    }

    public int getAmbiente(){

        android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
        return settings.getInt("amb_sel", 0);
    }

    public int getIDBoton(){

        android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
        return settings.getInt("ID_bot", 0);

}
}
```

Actividad de acciones de puerta.

En esta actividad el usuario puede visualizar quien se encuentra en la entrada de su hogar, para decidir si abre la puerta a no, para lo cual por motivos de seguridad se le solicitará una contraseña previamente configurada. Todas estas validaciones las veremos en la programación de esta actividad.

```
public class AccionesActuadorPuertaActivity extends Activity{

   private Button abrir,verCamara;
   Bitmap bmp;
   private ImageView elemento;
   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.acciones_grupo_tres_activity);
      elemento=(ImageView)findViewById(R.id.element);
      elemento.setImageResource(R.drawable.puerta_on);
   }

   @Override
   public void onBackPressed(){
      Intent amb=new
Intent(AccionesActuadorPuertaActivity.this,AmbientesFragmentActivity.class);
      startActivity(amb);
      super.onBackPressed();
   }

   public void ConfirmacionAbrir(){
      LayoutInflater li =  (LayoutInflater) AccionesActuadorPuertaActivity.this
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
      final View prompt = li.inflate(R.layout.conf_contrasena, null);//cambiar layout
      Button aceptar;
      final EditText pass;
      pass=(EditText)prompt.findViewById(R.id.pass1);
      aceptar=(Button)prompt.findViewById(R.id.acept);

      aceptar.setOnClickListener(new OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            if(getContrasena().equals(pass.getText().toString())){
                // Envío trama para abrir
                Toast.makeText(getApplicationContext(), "Contrasena correcta",
Toast.LENGTH_SHORT);
                finish();
            }
            else{
                Toast.makeText(getApplicationContext(), "Contrasena incorrecta",
Toast.LENGTH_SHORT);
                finish();
            }
            c
        }
    });

    Builder alertDialogBuilder = new AlertDialog.Builder(this);
    alertDialogBuilder.setView(prompt);
    alertDialogBuilder.setCancelable(true);

    AlertDialog alertDialog = alertDialogBuilder.create();
    alertDialog.show();
  }

  public String getContrasena(){
    android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS", 0);
    return settings.getString("contrasena", "1234*");
  }

  public void VerCamara(View v){
    //ACCEDER A CAMARA IP
    v.startAnimation(AnimationUtils.loadAnimation(this, R.anim.image_click));
  }

  public void AbrirPuerta(View v){
    //MANDAR TRAMA DE ABRIR PUERTA AL CONCENTRADOR
    v.startAnimation(AnimationUtils.loadAnimation(this, R.anim.image_click));
    ConfirmacionAbrir();
  }
```

```
   //MÉTODOS PARA ELEGIR LA IMAGEN DEL ELEMENTO SELECCIONADO Y
HACER EL QUERY DE LA TRAMA A ENVIAR
   public int getAmbiente(){

         android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
         return settings.getInt("amb_sel", 0);

   }


   public int getDispositivo(){

         android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
         return settings.getInt("disp_sel", 0);

   }

}
```

B.8. Actividad de acciones de garaje.

Esta actividad permite enviar las tramas correspondientes para abrir o cerrar la puerta del garaje.

```
public class AccionesActuadorGarajeActivity extends Activity{

   Bitmap bmp;


   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.acciones_grupo_cuatro_activity);
   }

   public void CerrarGarage(View v){
```

```
        //MANDAR TRAMA AL CONCENTRADOR DE CERRAR EL GARAGE
    }

    public void AbrirGarage(View v){
        //MANDAR TRAMA AL CONCENTRADOR DE ABRIR EL GARAGE

    }


    //MÉTODOS PARA ELEGIR LA IMAGEN DEL ELEMENTO SELECCIONADO Y
HACER EL QUERY DE LA TRAMA A ENVIAR
    public int getAmbiente(){

            android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
            return settings.getInt("amb_sel", 0);

    }


    public int getDispositivo(){

            android.content.SharedPreferences settings =
this.getSharedPreferences("SETTINGS",0);
            return settings.getInt("disp_sel", 0);
    }

}
```

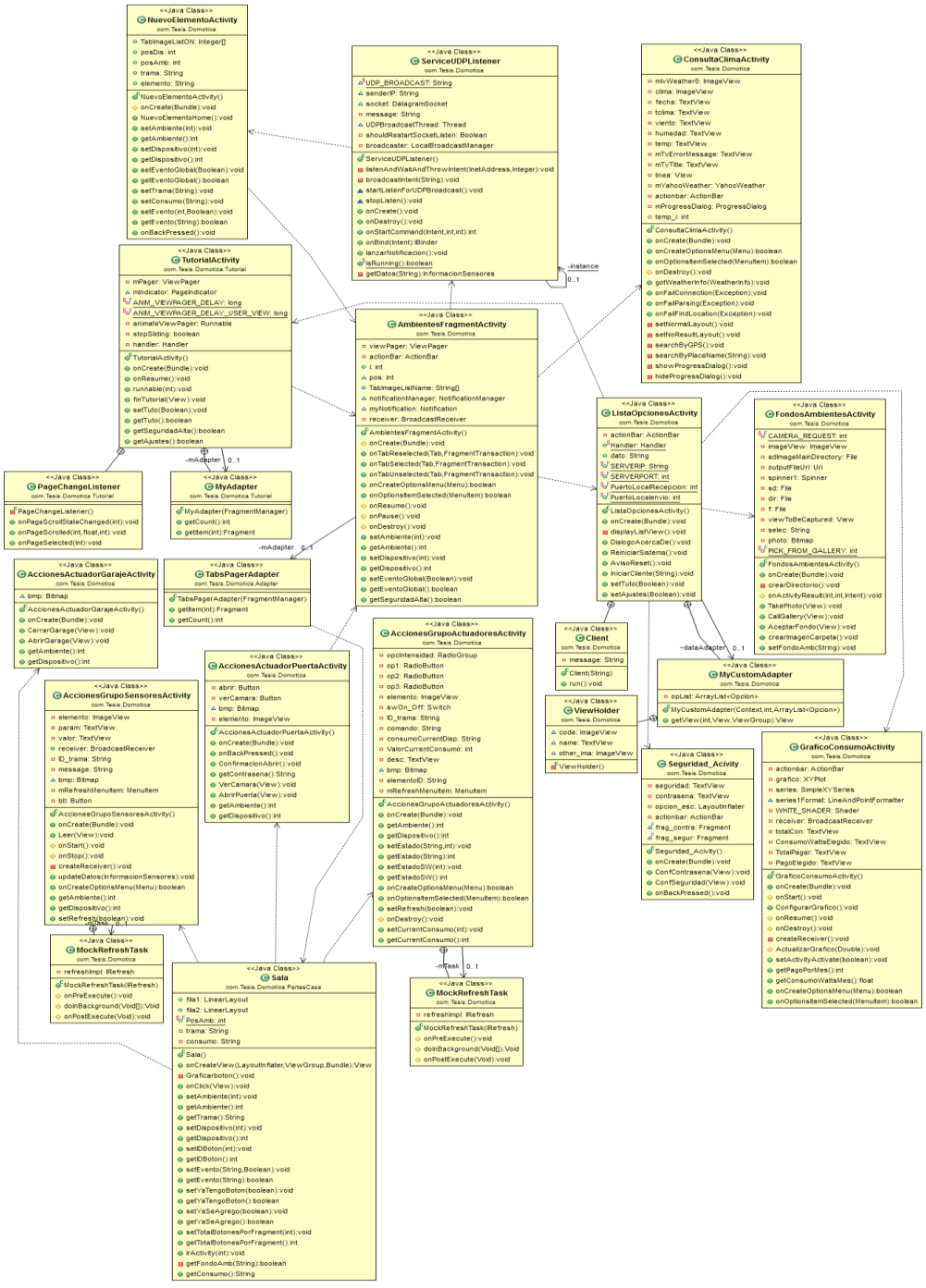


Figura H1 Diagrama de clases completo de la aplicación.

ANEXO I

MANUAL DE USUARIO

Se dará al usuario una noción de lo que puede hacer dentro de la aplicación.

1. **Inicio de Aplicación.**

   Una vez iniciada la aplicación el usuario podrá visualizar un tutorial dinámico con las principales funciones del sistema y la manera para acceder a cada una de ellas, también se le indica las notificaciones que recibirá en caso de que un evento se presente, esto referente a alarmas o nuevos dispositivos agregados al sistema domótico. Una vez presionado “Finalizar tutorial”, se procederá con la autenticación del usuario.

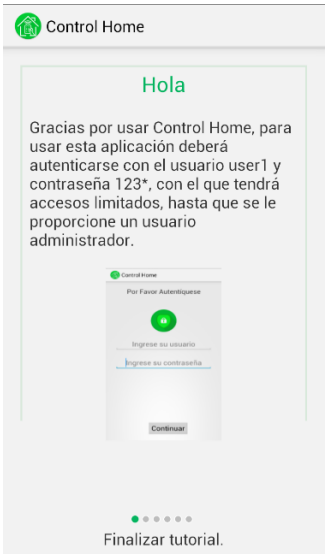


Figura I1 Pantalla de tutorial

2. Autenticación del usuario

Esta pantalla a parte de dar acceso a los ambientes del hogar y las distintas opciones, permite marcar un nivel de privilegios, dependiendo el usuario con el que se autentique. En la pantalla de ajustes se verán reflejadas las limitaciones y accesos que tienen el perfil de usuario general y el perfil de usuario administrador.

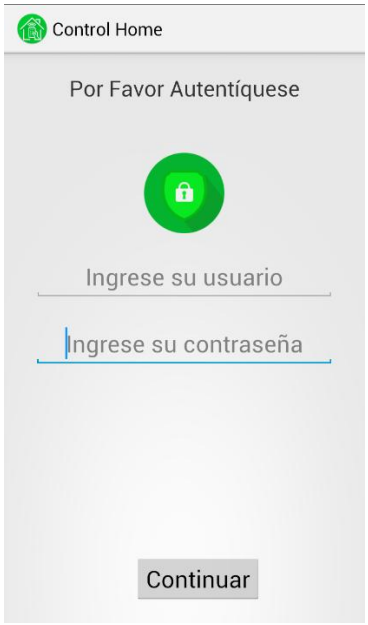


Figura I2 Pantalla de autenticación

3. Actividad de ambientes

En esta pantalla el usuario podrá acceder a los diferentes ambientes del hogar, en este caso al ser la primera vez que inicia la aplicación, los ambientes no tendrán ningún elemento del hogar para controlar. Desde esta pantalla se puede acceder a una lista de opciones e información del clima de la ciudad local, esto mediante los botones en la barra de acción.

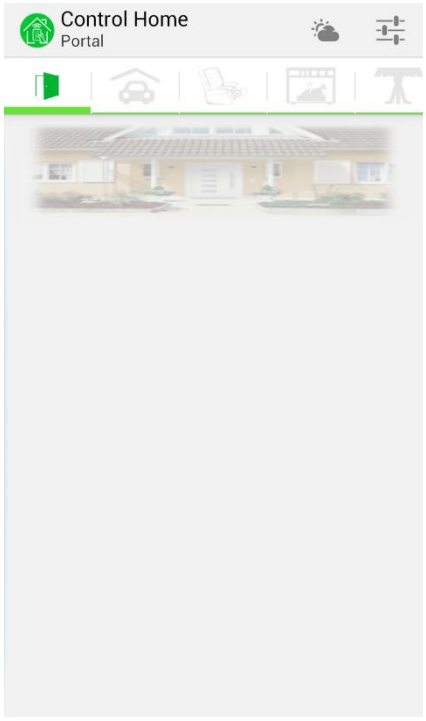


Figura I3 Pantalla de ambientes

4. Actividad de consulta de clima

El usuario accede a esta pantalla de información mediante un botón en la barra de acciones de la pantalla de ambientes. Una vez en esta pantalla se generará una consulta de parámetros de clima, mediante una consulta a los servidores de Yahoo.

Figura. I4 Pantalla de consulta de clima

El botón en la barra de acciones de la actividad consulta de clima, permite refrescar la información presentada al usuario.

Al inicio de la actividad de consulta de clima o cuando se presione el botón de refrescar, el usuario tendrá la siguiente vista.

Figura I5 Pantalla de actualización de información del clima

5. Actividad de ajustes

Esta pantalla le muestra al usuario una lista de opciones de configuración a nivel de aplicación, como cambiar los fondos de ambientes, u opciones de información, como poder visualizar en un gráfico su consumo energético. También podrá realizar acciones generales a nivel de sistema, como un reinicio o una prueba de conexión con el concentrador.

Figura I6 Pantalla con lista de opciones de ajustes

Se detallará las acciones y vistas de cada una de las opciones de la lista de ajustes.

- Acerca de: Si el usuario escoje esta opción, podrá observar mediante un diálogo, donde se le informa quienes desarrollaron la aplicación y el fin de hacerlo.

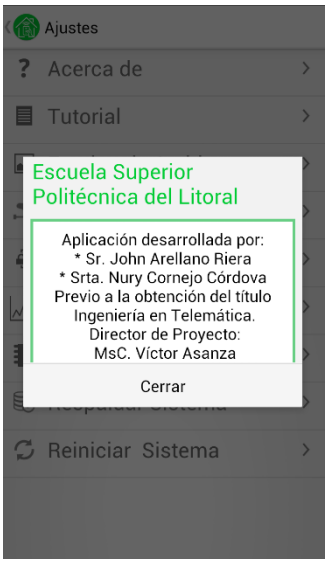


Figura I7 Pantalla con AlertDiálogo de información de desarrolladores.

- Tutorial: Como se explicó en puntos anteriores esta es la pantalla que visualiza el usuario cuando igresa por primera vez a la aplicación, sin embargo esta opción está disponible en la lista de opciones de ajustes.

Figura I8 Pantalla de tutorial dinámico.

Recordar que el usuario puede apreciar el tutorial de manera manual o automática, dado que la aplicación detecta si no se toca la pantalla, procede a cambiar de pestaña automáticamente, después de 5 segundos.

- Fondos de ambiente: Si el usuario escoge esta opción, podrá personalizar las pestañas de ambientes con imágenes de fondo o fotos de los ambientes de sus hogares.

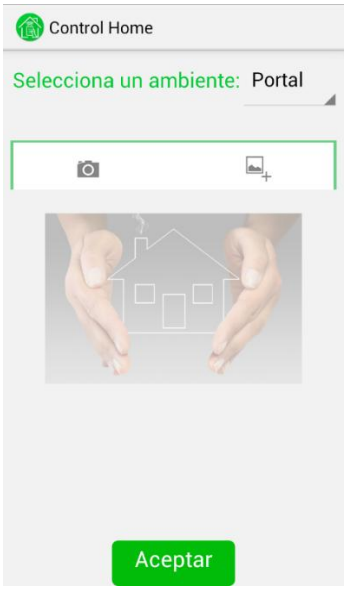


Figura I9 Pantalla de cambio de fondo de ambientes

En la figura I8 se muestra la actividad que corresponde a la configuración de los fondos de los ambientes, el usuario aquí podrá escoger el ambiente y luego seleccionar de que manera desea cargar la foto, pudiendo ser por la cámara o una guardada previamente en la galería.

- Cámara IP: Esta opción le permite al usuario, visualizar en tiempo real las capturas de la cámara IP instalada en su sistema domótico.

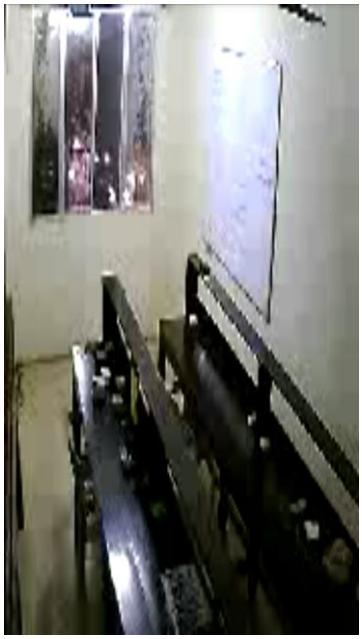


Figura I10 Pantalla de capturas de cámara IP

- Seguridad: Esta sección de ajustes permite al usuario modificar el tipo de seguridad predeterminada con la que está configurada la aplicación, además le permitirá al usuario configurar la contraseña que usará cuando abra la puerta de su entrada principal.

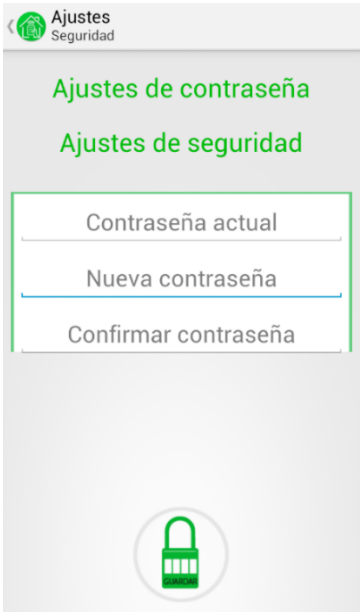


Figura I11 Pantalla de ajustes de contraseña

- Consumo energético: Esta sección de ajustes, le permite al usuario configurar el monto en dólares que desea pagar, por los dispositivo finales conectados al sistema. Además se mostrará un gráfico de consumo energético en tiempo real e información sobre los datos de consumo energético máximo, según la cantidad máxima en dólares que configuró el usuario. Cuando este consumo energético se acerca al límite, envía una notificación al usuario, para que tome medidas correctivas y cambie la tendencia de su consumo.

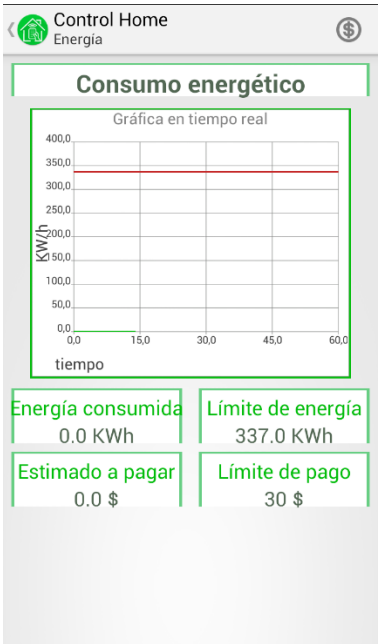


Figura I12 Pantalla de Consumo energético

La figura I10 muestra la pantalla de visualización del consumo energético, en tiempo real, cabe recalcar que para el cálculo de este consumo se considera solo los elementos que estén enlazadaos a la aplicación. Dentro de esta opción el usuario también podrá configurar el límite de pago que desea realizar por mes.

- Prueba de conexión

Figura I11 Prueba Conexión

El usuario podrá realizar una prueba de conexión hacia el concentrador, verificando la funcionalidad correctamente del mismo. Esta opción será útil para futuras revisiones de problemas que pueda existir en el sistema.

- Respaldar Sistema: Esta opción permite guardar o importar una copia de las configuraciones del sistema domótico, esto respecto a los a los elementos agregados en cada ambiente. Lo descrito es de mucha utilidad al momento de que se de una mudanza no volver a configurar y agregar

todo, o si existe alg{un problema adicional que proboque la perdida de datos del dispositivo, se puede realizar una importaci{on de lo que ya estaba previamente respaldado.

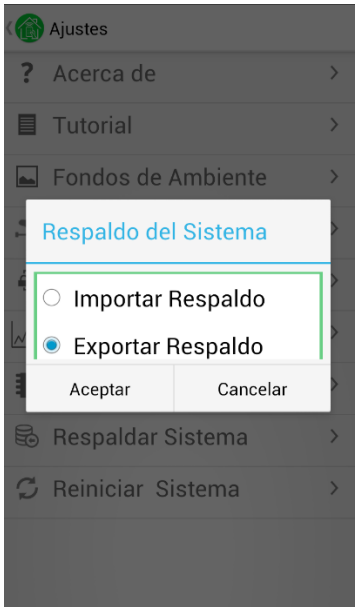


Figura I12 Respaldo del Sistema Domótico

- Reiniciar sistema: Con esta opción el usuario podrá realizar un borrado de todos sus datos, volviendo a reiniciar el sistema y por ende volver a enlazar los dispositivos finales a la aplicación. Esta opción servirá en caso de que el usuario desee volver a conectar los dispositivos en otro sitio de su casa, o cuando realice una mudanza de hogar, de esta manera en su nuevo hogar, podrá volver a configurar los dispositivos finales a su conveniencia.

6. Nuevo elemento conectado

Figura I12 Notificación de nuevo elemento conectado

Esta pantalla se visualizara únicamente cuando un nuevo elemento por primera vez, sea conectado al hogar. Según el tipo de dispositivo que se conecte, la aplicación lo detectará y cargará las respectivas opciones. El usuario deberá ingresar el valor de consumo de wats, que todo equipo trae.

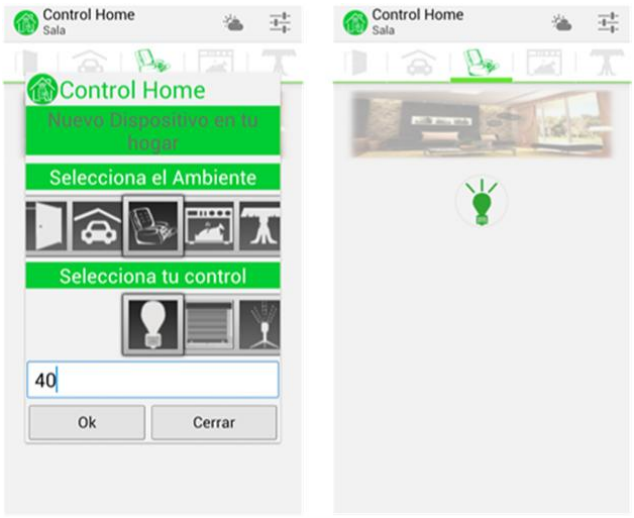


Figura I13 Nuevo elemento agregado

Una vez que el usuario acepte el elemento, este se procederá a cargar automáticamente al ambiente correspondiente.

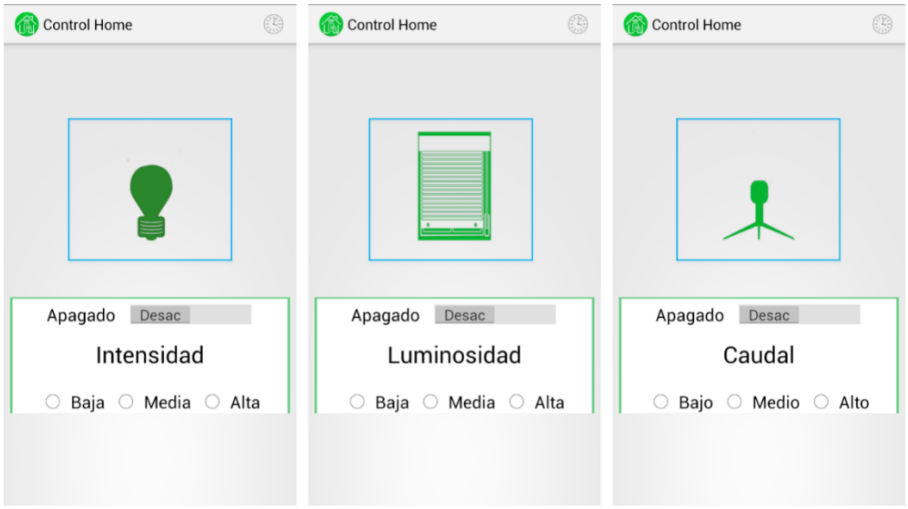


Figura I14 Pantalla de Dispositivos finales actuadores

EL usuario al presionar algún elemento de tipo actuador, para el caso de la iluminaria (Foco), persiana o regadera, se procederá a cargar una nueva pantalla con las respectivas opciones que trae el dispositivo presionado.

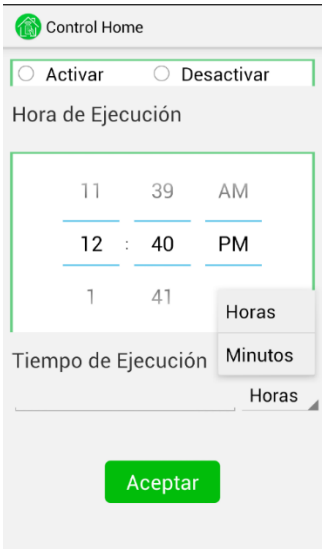


Figura I15 Programación de Acciones automáticas

El usuario podrá acceder a la opción de acciones programadas automáticas, presionando el botón que se encuentra en la esquina superior derecha de la pantalla de la figura I13. Aquí el podrá decidir si la acción que desea realizar es de activación o desactivación, la hora en que desea que ocurra el evento y el tiempo en que estará activo dicho evento.

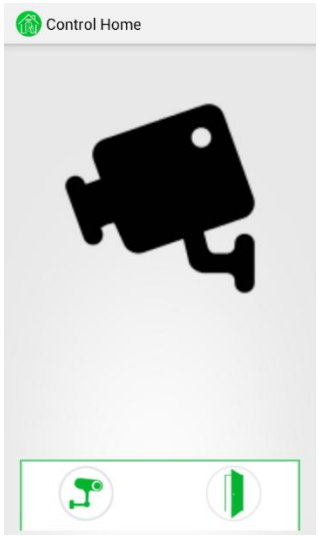


Figura I16 Abrir puerta desde aplicación.

Cuando la aplicación detecte que se conecte un dispositivo final de tipo modulo actuador para chapa eléctrica, le dará como opción al usuario poder abrir la puerta de su hogar, internamente o remotamente, y a su vez poder verificar previamente que persona desea entrar al hogar, accediendo a una cámara IP que por lo general todo hogar trae consigo en su portal.

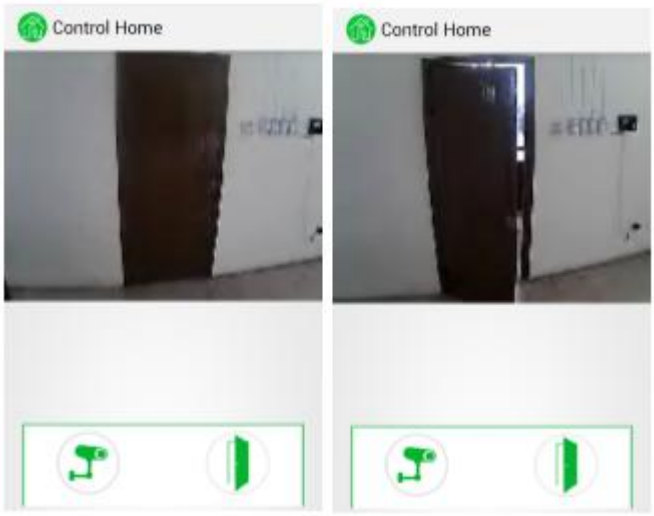


Figura I17 Acceso a cámara IP

El usuario presionando el botón de la cámara, de la aplicación, podrá acceder a su cámara IP y verificar la persona quien desea entrar.

Esta opción es útil dado que sirve a la vez como videovigilancia.

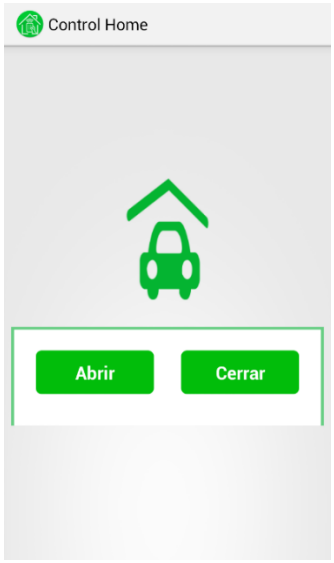


Figura I18 Abrir/Cerrar garaje

Cuando la aplicación detecte que se conecte un dispositivo final de tipo modulo actuador de apertura de garage, le dará como opción al usuario poder abrir y/o cerrar la puerta de su garage, internamente o remotamente, desde su aplicación móvil.

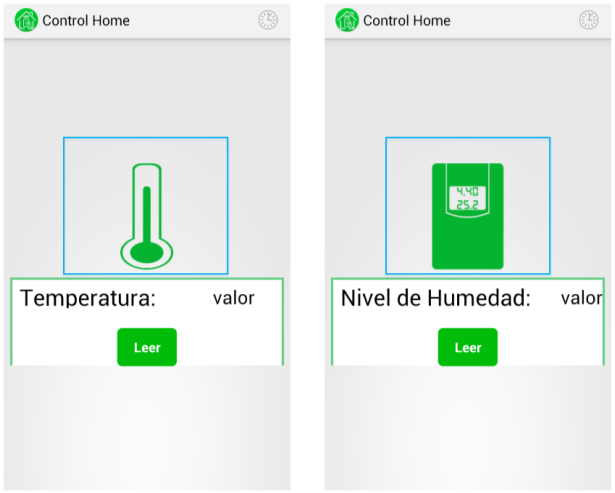


Figura I19 Lectura de sensores

Cuando la aplicación detecte que se conecte un dispositivo final de tipo modulo sensor de parámetros ambientales, le dará como opción al usuario poder leer dicho valor, para luego poder actuar ante según sea el caso y las necesidades del mismo usuario.

ANEXO J

MÓDULOS IDETEC USADOS

En esta sección se menciona y detalla a breves rasgos de los módulos Idetec usados en el sistema domótico.

Módulo Idetec – adaptador TTL.

Este es un módulo de comunicación serial que sirve para ajustar los niveles de voltaje de cualquier módulo XBEE a niveles TTL. Sirve para interface de comunicación serial para niveles TTL con módulo XBEE; adquisición y envío de datos seriales mediante módulo XBEE.

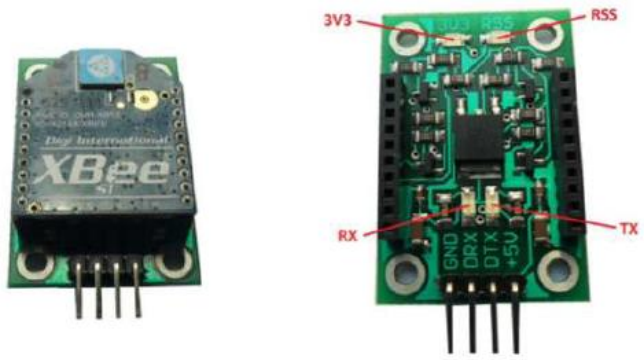


Figura J1 Modulo Idetec TTL para XBee

Especificaciones

- Este módulo cuenta con un LED indicador de energizado, RX, TX y RSS
- Bus de 4 pines para polarización y comunicación serial.
- Soporta de 7 a 8 bits de datos, 1 o 2 bits de stop, y odd/even/mark/space/ no parity.

Módulo controlador para actuadores dimerizadores

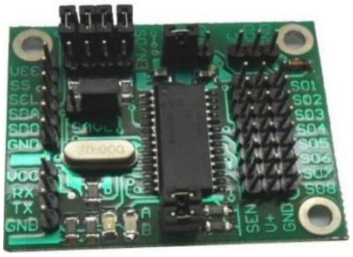


Figura J2 Modulo Idetec controlador 16F886

En la figura J2 podemos observar el módulo controlador, cuyo principal componente es el microcontrolador PIC16F886.

Este módulo es usado para actuadores, específicamente de dispositivos finales que pueden ser dimerizados. Para este fin se usa el PWM del microcontrolador PIC16F886.

Frecuencia de trabajo	8 MHz a 32 kHz, seleccionables.
Comparadores	2 comparadores, 2 canales PWM
Comunicaciones serie	Modo SPI módulo de serie Maestro síncrono Puerto (MSSP). Modo I2C con capacidad de máscara de dirección.
Temporizadores	TMR0: 8-bit del temporizador / contador con 8 bits prescaler. TMR1 mejorada: 16-bit del temporizador / contador con prescaler. TMR2: 8-bit del temporizador / contador con 8 bits periodo de registro, prescaler y postscaler
Memoria EEPROM	256 bytes EEPROM no volatil (como un periférico)
Memoria Flash	32K memoria flash de programa reprogramable (16K instrucciones)

Tabla J1 Características del PIC16F886

El PIC16F886 es ideal para aplicaciones industriales, automoción, aparatos o aplicaciones de consumos, de nivel avanzado, debido a las características descritas.

La finalidad de este microcontrolador en nuestro sistema domótico inalámbrico, es usar sus salidas analógicas para ejercer acciones sobre los dispositivos finales mencionados, para lo cual se comunica de manera serial con el módulo XBee, por el cual recibe las instrucciones enviadas por el usuario mediante la aplicación y luego receptadas y procesadas por el concentrador.

ANEXO K

CARACTERÍSTICAS DE LA APLICACIÓN ANDROIDE

Características funcionales de la aplicación androide

1. Enlazar y conectar dinámicamente los dispositivos, de manera inalámbrica, lo que permite guardar la integridad de la infraestructura del hogar.
2. Permite encender, apagar o controlar intensidad con que serán activados ciertos elementos del hogar, como foco, regadera, persiana.
3. Control de climatización, gracias a información de parámetros ambientales como temperatura, humedad, que envían los respectivos sensores.
4. Control de riego de jardín, mediante programación de tiempo de riego y cantidad de agua que fluye desde la bomba.
5. Información del consumo de energía en tiempo real de los dispositivos agregados al sistema, mediante gráficos estadísticos
6. Permite ingresar un valor máximo a pagar por consumo energético de los dispositivos agregados, alertando automáticamente al usuario cuando su consumo se acerque al valor que ingreso.

7. Permite personalizar la aplicación, a nivel de fondos de los distintos ambientes del hogar.
8. Verifica el correcto funcionamiento del concentrador, y notifica en caso de error.
9. Presenta información sobre el clima de la ciudad en la que se encuentra el usuario.
10. Permite programar horas de vigilancia, y alerta al usuario en caso de detección de intrusos.
11. Acceso a una cámara IP que le permite al usuario verificar quién desea ingresar a su hogar.
12. Abrir puerta de manera segura, por medio de una contraseña de verificación, previamente configurada por el usuario.
13. Aabrir o cerrar puerta de garaje.
14. Programar hora de inicio y rango de tiempo de ejecución de acciones de encendido o apagado, sobre ciertos elementos del hogar.
15. Configurar niveles de seguridad para uso de la aplicación y control de los elementos del hogar.
16. Opción de reinicio del sistema.

Características técnicas de la aplicación androide

1. Funcional a partir de la versión 4.0 en adelante
2. Espacio mínimo requerido para la instalación: 5MB
3. Conexión a internet, en caso de controlar remotamente el hogar
4. Consumo de batería del móvil 2%
5. Memoria ram del móvil, usada 8MB
6. Consumo de paquetes de datos por día: Aproximadamente 15KB

ANEXO L

CARACTERÍSTICAS TÉCNICAS DE HARDWARE

MODULO CONCENTRADOR
1. Caja negra, dimensiones 14x9x6 cm
2. Peso <100 gr
3. Voltaje de operación 5V, 800 mA
4. Consumo de Energía: 4W
5. Rango de operación temperatura: -40° a 85° C

DISPOSITIVOS FINALES MODULO ACTUADOR
Caja ploma , dimensiones 8x8x5 cm
Peso <40 gr
Voltaje de operación 12V, 1ª
Consumo de Energia: 12W
Rango de operación temperatura: -40° a 85° C

DISPOSITIVOS FINALES MODULO SENSOR
Caja ploma , dimensiones 8x8x5 cm
Peso: <40 gr
Voltaje de operación : 5V
Consumo de Energia: 6W
Rango de operación temperatura: -40° a 85° C

ANEXO M

COSTOS DE IMPLEMENTACIÓN DE SISTEMA DOMÓTICO INALÁMBRICO

En esta sección indicaremos los costos de todos los elementos usados para la implementación del sistema domótico funcional.

Se detallará los elementos usados para los diferentes módulos del sistema, para luego obtener un valor total de inversión de este proyecto.

1. Costos de módulo concentrador.
2. Costos de módulo de dispositivo final actuador.
3. Costos de módulo de dispositivo final sensor.
4. Costos de desarrollo.
5. Costo total del sistema.

Costos para implementación de módulo concentrador.

Elemento	Cantidad	Costo Unitario	Costo Total
Cristal 10 Mhz	1	0,7	0,7
Capacitor 22 pF	2	0,15	0,3
LED azul	1	0,25	0,25
Resistencia 220 ohm	1	0,05	0,05
Fuente de 5V	1	3	3
Módulo Ethernet	1	11	11

PIC18F4520	1	7	7
Módulo XBee	1	70	70
Módulo Idetec TTL	1	28	28
Caja plástica	1	12	12
Circuito impreso	1	8	8
Total			140,3

Tabla M1 Costos del concentrador

En la tabla M1 se detalló cada uno de los módulos y elementos usados para conformar el concentrador, cuyo costo final fue de $140,30 dólares.

Costos de módulo de dispositivo final actuador

Elemento	Cantidad	Costo Unitario	Costo Total
Jack AC hembra	1	0,70	0,70
Módulo Idetec disparador de relé	1	16,80	16,80
Módulo Idetec controlador con PWM	1	23,00	23,00
Módulo Idetec Servomotor	1	21,00	21,00
Módulo Idetec regulador de voltaje	1	5	5
Fuente de 12V	1	5,00	5,00
Módulo XBee	1	50,00	50,00

Módulo Idetec TTL	1	1	28.00
Caja plástica	1	2,20	2,20
Total			146,7

Tabla M2 Costos de módulo de dispositivos actuadores

En la tabla M2, se detalló cada uno de los módulos y elementos usados para conformar módulo actuador, cuyo costo final fue de $146,7dólares.

Costos de módulo de dispositivo final sensor

Elemento	Cantidad	Costo Unitario	Costo Total
Baterías Lithium Ion	2	5	10
Módulo Idetec regulador de voltaje	1	5,00	5,00
Módulo Idetec Sensores (humedad, presencia, temperatura)	1	11,00	11,00
Módulo Idetec controlador	1	7,00	7,00
Módulo XBee	1	50,00	50,00
Módulo Idetec TTL para XBee	1	1	28.00
Total			111,00

Tabla M3 Costos de módulo de dispositivos sensores.

En la tabla M3, se detalló cada uno de los módulos y elementos usados para conformar módulo sensor, cuyo costo final fue de $111 dólares.

Costos de desarrollo.

En esta sección se detalla los costos de mano de obra, y del firmware tanto del hardware como del software, para el caso de la aplicación.

Tipo	Costo
Mano de obra de ensamblaje.	200
Firmware de hardware	30
Aplicación Android	5
Total	235,00

Tabla M4 Costos de Desarrollo.

En la tabla M4, se detalla los costos por mano de obra, firmware del hardware y del desarrollo de la aplicación android.

Costo total del sistema.

En esta sección se pensó en un kit básico del sistema, en el cual contenga los siguientes elementos o dispositivos finales:

- 1 Concentrador
- Dispositivos finales Actuadores:
  - 1 Módulo actuador iluminaria
  - 1 Módulo actuador persiana
  - 1 Módulo actuador chapa eléctrica
- Dispositivos finales sensores:
  - 1 Módulo sensor de presencia.
  - 1 Módulo sensor de humedad.
  - 1 Módulo sensor de Temperatura.
- Aplicación androide

Elemento	Cantidad	Costo Unitario	Costo Total
Módulo concentrador	1	140,30	140,30
Dispositivos finales actuadores	3	146,7	440,1
Dispositivos finales sensores	3	111,0	333,0
Costos de desarrollo	1	235,0	235,0
Total			1148,4

Tabla M5 Costos Total del sistema

En la tabla M5, se detalla el costo total del sistema, pensando en un kit básico del sistema domótico.

ANEXO N

RESUMEN DE FOTOS DEL SISTEMA DISEÑADO

Figura N1 Concentrador del sitema.

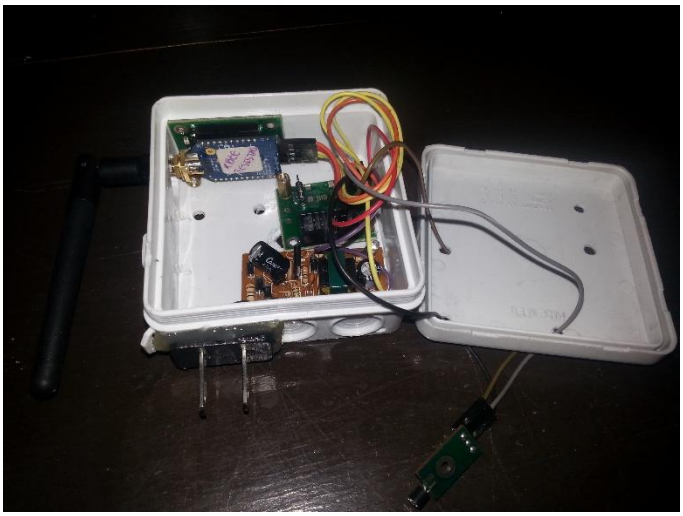


Figura N2 Módulo sensor de temperatura

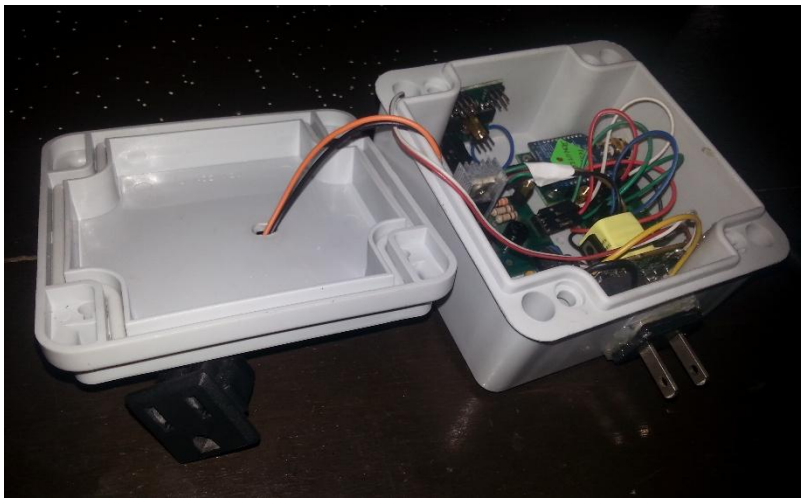


Figura N3 Módulo actuador de iluminaria dimerizable.

Figura N4 Vista completa de los componentes del sistema domótico.

Figura N5 Cámara Ip.

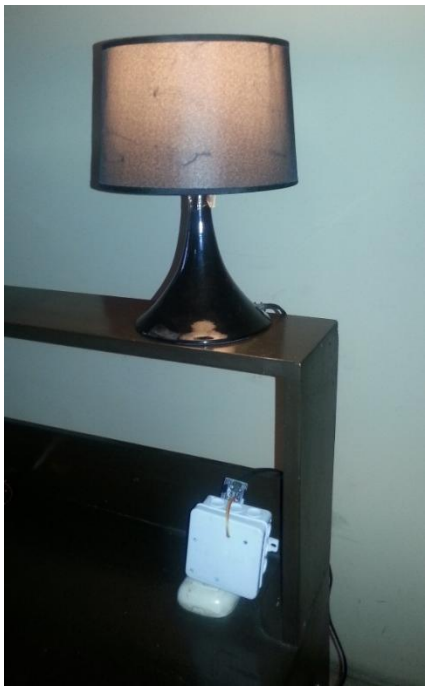


Figura N6 lámpara dimerizable con módulo actuador

GLOSARIO DE TERMINOS

SQLite: es un sistema de gestión de bases de datos relacional compatible con

ACID, contenida en una relativamente pequeña biblioteca escrita en C, es usado por Android.

Bluetooth: es una especificación industrial para Redes Inalámbricas de Área Personal que transmite datos y voz entre diferentes dispositivos mediante un enlace por radiofrecuencia punto a punto.

Smartphone: Es un teléfono móvil que disponen de un hardware y un sistema operativo propio, que permite instalación de aplicaciones, juegos, etc.

Modularidad: propiedad que permite subdividir una aplicación en partes más pequeñas, cada una debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

Polimorfismo: capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación.

MIMO: Se refiere específicamente a la forma como son manejadas las ondas de transmisión y recepción en antenas para dispositivos inalámbricos como enrutadores.

BIBLIOGRAFÍA

[1] Jara Antonio, Introducción a la domótica viviendas inteligentes, http://www.um.es/aulasenior/saavedrafajardo/apuntes/doc/introduccion_DOMÓTICA_vivienda_inteligente.pdf, fecha de consulta diciembre 2014.

[2] Rodríguez María, Hardware para domótica, Tema II Sensores y Actuadores, fecha de consulta diciembre 2014.

[3] Idetec, Catálogo de productos, http://www.ideastechnology.com/?q=portfolio-4, fecha de consulta noviembre 2014.

[4] Fiori Santa María, Sistema operativo Android, http://blog.staffcreativa.pe/android-ventajas-desventajas/http://blog.staffcreativa.pe/android-ventajas-desventajas/, febrero 2014.

[5] GitHub, Características de Android, http://androidos.readthedocs.org/en/latest/data/caracte, fecha de consulta diciembre 2014.

[6] Heméndez Roberto, Introducción a Java e historia, Programación orientada a objetos (Parte II), enero 2015.

[7] Pérez Damian, ¿Qué son las bases de datos?, http://www.maestrosdelweb.com/que-son-las-bases-de-datos/, octubre 2007.

[8] WiFiClub, WiFi: historia, evolución, aplicaciones, http://www.wificlub.org/featured/wifi-historia-evolucion-aplicaciones-desarrollos/#, enero 2010.

[9] Arqhys, Sistema domótico inalámbrico, http://www.arqhys.com/arquitectura/domotica-historia.html, fecha de consulta enero 2015.

[10] Electronicspk, PIC18F4520, http://electronicspk.com/xcart/PIC18F4520.html, fecha de consulta noviembre 214.

[11] Microchip, MikroC, http://www.mikroe.com/mikroc/pic/, 2015, fecha de consulta enero 2015.

[12] Cirett Federico, Logo de Java, http://www.k-bits.com/2013/01/12/hola-mundo/logo-java/, enero 2013.

[13] Eclipse, descarga de IDE eclipse, https://eclipse.org/, fecha consulta enero 2015.

[14] Álvarez Eric, Capas de Android, http://androideric.blogspot.com/2013/02/13-arquitectura-de-android.html, febrero de 2013.

[15] Idea Digital, Módulo Ethernet ENC28J60, http://idelectronica.com.mx/arduino/modulo-ethernet-enc28j60/, fecha consulta diciembre 2014.

[16] Trujillo P., Conexión entre módulo Ethernet y microcontrolador, http://www.mipsandchips.com.es/2012/09/servidor-web-desde-cero-placa-v.html, septiembre 2012.