INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

“Sistema de Detección y Prevención de Incendios en una Casa Habitación”

14-2-0003

Que para cumplir con la opción de titulación curricular en la carrera de:

“Ingeniería en Sistemas Computacionales”

Presentan

Márquez Malpica Aura Jessid
Martínez López Yoshio Alexis

Directores

M. en C. Ismael Cervantes de Anda

Ing. José Luis Hernández Aguilar

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

No. de Registro: 14-2-0003 Serie: Amarilla Mayo 2014

Documento Técnico

Sistema de Detección y Prevención de Incendios en una Casa Habitación

Autores:

Márquez Malpica Aura Jessid[1]
Martínez López Yoshio Alexis[2]

RESUMEN

El presente documento describe el desarrollo de un sistema de detección y prevención de incendios a ser implementado en una casa habitación, compuesto por un dispositivo móvil para visualizar la información del sistema y recibir notificaciones cuando éste se encuentre en estado de alerta, un módulo de comunicación central y uno o varios módulos de monitoreo y control automatizado.

Palabras clave.- Desarrollo de aplicaciones para dispositivos móviles, control automático, sistemas embebidos, domótica.

Directores

M. en C. Ismael Cervantes de Anda **Ing. José Luis Hernández Aguilar**

[1] amarquezmalpica@gmail.com
[2] yoshioalexis@gmail.com

ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE FORMACIÓN INTEGRAL E INSTITUCIONAL

ESCOM

COMISIÓN ACADÉMICA DE TRABAJOS TERMINALES

México, D.F. a 11 de junio de 2014

DR. FLAVIO ARTURO SÁNCHEZ GARFIAS
PRESIDENTE DE LA COMISIÓN ACADÉMICA
DE TRABAJOS TERMINALES
P R E S E N T E

Por medio del presente, informamos que el alumno que integra el **TRABAJO TERMINAL:** 14-2-0003, titulado: "Sistema de Detección y Prevención de Incendios en una Casa Habitación" concluyó satisfactoriamente su trabajo.

El empastado del Reporte Técnico Final y el Disco Compacto (CD) fueron revisados ampliamente por sus servidores y corregidos, cubriendo el alcance y objetivo planteados en el protocolo original y de acuerdo a los requisitos establecidos por la Comisión que usted preside.

ATENTAMENTE

M. en C. Ismael Cervantes de Anda

Ing. José Luis Hernández Aguilar

Advertencia

"Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan."

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n al teléfono: 57-29-6000 Ext. 52000.

Agradecimientos

Quiero agradecer en primer lugar a mi compañero de Trabajo Terminal y novio en la vida real, Yoshio Alexis Martínez López, ya que sin él este trabajo no se hubiera podido llevar a cabo. Gracias por las noches de desvelo juntos, las risas histéricas a las cuatro de la mañana, los regaños, las lágrimas, las sonrisas, los abrazos y los besos en momentos de crisis (y también cuando no era momento de crisis) y sobre todo por su amor incondicional en todo momento. Te amo mucho apestoso. ¡Somos libres!

Gracias a mi familia, que han estado ahí en todo momento, queriéndome, apoyándome y no dejando que caiga en desesperación y frustración, dándome ideas y ayudando en todo lo que podían.

Gracias a mi mamá, Aura Estrella Malpica Salinas, por todos esos años de desvelos y cariño, por quererme, comprenderme (y tolerar mi carácter en tiempos de estrés) y por estar ahí todo el tiempo. Te quiero mucho mamucha.

Gracias a mi papá, Javier Gerardo Márquez Ojeda, por la gran ayuda en el diseño y la construcción del prototipo de pruebas, por estar disponible en todo momento y por todo el apoyo que me has dado siempre. Te quiero papi.

Gracias a ese Ángel que yo sé que me está cuidando desde el cielo, que se alegró más que nadie cuando logré entrar a esta carrera y que no pudo ver este logro en vida, pero que siempre estará en mi corazón. Esto va para ti, Chuchis.

Gracias a la familia Martínez López, por recibirme en su casa con los brazos abiertos, por brindarnos apoyo y cariño en todo momento y por dejarnos trabajar y hacer desastres en su casa. Muchas, muchas gracias.

Gracias a nuestros directores de trabajo terminal (Ismael Cervantes de Anda y José Luis Hernández Aguilar) por su ayuda y orientación en la realización de este trabajo terminal.

Gracias a los amigos que he conocido a lo largo de la carrera en ESCOM. No puedo nombrarlos a todos pero espero que se sientan abrazados desde estas líneas.

Gracias a todos mis compañeros del CIDETEC, que estuvieron ahí todo el tiempo apoyando tanto física como moralmente, por ayudarnos a aprender lo que no sabíamos y por hacernos reír para desestresarnos. Mil gracias a cada uno de ustedes.

Gracias al Dr. Gabriel Sepúlveda por aceptarnos en el CIDETEC, por dejarnos quedarnos a trabajar con él y darnos los conocimientos, el tiempo, el espacio y todas las facilidades posibles para que pudiéramos terminar este trabajo.

Aura Jessid Márquez Malpica

Agradecimientos

Quiero otorgar un agradecimiento muy especial para Aura Jessid Márquez Malpica (apestosa) quien además de ser mi compañera de TT también es mi novia, por ayudarme a terminar este trabajo, por no volverse loca con mis ideas, por ayudarme en las situaciones más difíciles, y sobre todo por estar conmigo en todos los momentos (felices y tristes) que pasamos juntos haciendo esto ya que sin ella no hubiera podido realizar este trabajo. Gracias apestosa te amo mucho.

Gracias a mis padres quien me brindaron todo su apoyo, por habernos aguantado todo este tiempo haciendo relajo en la casa, por alimentarnos, por echarnos porras para que termináramos, por darnos nuevas ideas, y sobre todo por quererme tanto. Los amo.

Gracias a mis directores (Ismael Cervantes de Anda y José Luis Hernández Aguilar) por habernos ayudado en el desarrollo de este trabajo y por habernos dado todas las ideas necesarias para llevar a cabo este TT.

Gracias a la familia Márquez por todo el apoyo brindado en este trabajo, por las ideas y sobre todo gracias a Javier Márquez Ojeda por la ayuda en la planeación y desarrollo del prototipo.

Gracias a Aura Malpica Salinas y Aura Salinas Alemán por habernos soportado en su casa largas noches para poder terminar TT1 y TT2, por habernos alimentado y por habernos dado todo su apoyo.

Gracias a todos mis compañeros del CIDETEC por habernos brindado todo su apoyo sin ningún compromiso y sobre todo por su solidaridad. Muchas gracias.

Gracias al Dr. Gabriel Sepúlveda Cervantes por habernos dado el tiempo necesario para terminar este proyecto, por habernos ayudado con algunos de los materiales y sobre todo por el conocimiento adquirido ya que sin ello no hubiéramos podido lograrlo.

Yoshio Alexis Martínez López

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE DIAGRAMAS

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1.1. Problemática

En la actualidad nuestro país carece de una cultura de prevención de incendios en zonas urbanas, esto debido a que no hay estadísticas y hay muy pocos programas gubernamentales que informen a las personas acerca de este tema. A causa de esto, las personas no cuentan con un sistema de prevención de incendios en sus hogares.

El uso de dispositivos móviles ha tenido un gran crecimiento en los últimos años, ya que éstos permiten el envío y recepción de información desde cualquier parte del mundo, gracias a la capacidad que tienen para conectarse a Internet. Este crecimiento ha motivado a diversas empresas y desarrolladores a crear aplicaciones para cubrir las necesidades de una gran gama de usuarios.

Actualmente no existen sistemas que puedan notificar al usuario de manera remota acerca de riesgos como incendios, fugas de gas y contaminación por monóxido de carbono o humo.

1.2. Solución Propuesta

Ante la problemática anterior, se propone el desarrollo de un sistema para detectar y prevenir incendios, notificando del riesgo al usuario permitiendo así aprovechar el uso de dispositivos móviles.

1.3. Justificación

De acuerdo con información de la NFPA[3] y la AMRACI[4] de cada 100 incendios en zonas urbanas el 63.5% se generan en casas habitación multifamiliares, mientras que 22.1% se generan en casas habitación de una sola familia. Gran parte de ellos se suscitan debido a la apatía o el desconocimiento de los habitantes, descuidos y la falta de sistemas de prevención y acción. Las causas más comunes por las que se presenta un incendio son fallas eléctricas, fallas en la instalación de gas, combustión espontánea por exceso de basura y desorden, manejo inadecuado de líquidos inflamables y mantenimiento deficiente de tanques contenedores de gas. Las principales causas de muerte en un incendio son: 62.4% por asfixia debido a inhalación de humo, 26% por quemaduras y 10.7% por lesiones traumáticas [1].

Tomando en cuenta estas cifras, resulta alarmante que en México no se fomente la cultura de prevención de incendios urbanos y que las personas no cuenten con sistemas contra incendios en sus casas.

1.4. Objetivo General

Implementar un sistema que permita el monitoreo de variables físicas relacionadas con la detección y prevención de incendios en casas habitación (temperatura, humedad, presencia de flama y concentración de humo o gas) mediante módulos autónomos, así como la

[3] National Fire Protection Association
[4] Asociación Mexicana de Rociadores Automáticos Contra Incendios

visualización de la información en un dispositivo móvil, incluyendo el envío de notificaciones cuando el sistema se encuentre en estado de alerta.

1.4.1. Objetivos Específicos

- Crear una aplicación para un dispositivo móvil cuya función sea representar los datos que recibe desde el servidor mediante una interfaz gráfica.
- Crear un conjunto de módulos que obtengan mediciones del entorno, tomen decisiones con base en ellas para activar o desactivar actuadores y las envíen al servidor.
- Implementar un algoritmo para determinar las acciones a realizar con base en las mediciones de los sensores y los rangos de peligro de cada variable física.
- Crear una aplicación para un servidor cuya función sea almacenar los datos que le envíen los módulos y otra para enviarlos al dispositivo móvil.

1.5. Contexto

Este sistema está diseñado para ser instalado y utilizado en casas habitación unifamiliares, multifamiliares y departamentos. Tiene gran utilidad en situaciones tales como: detección de fugas de gas, contaminación por monóxido de carbono e incendios. Permite una gran flexibilidad en la distribución de los módulos y la distancia entre ellos gracias a las características de los módulos de comunicación de la red.

1.6. Estado del Arte

En la Tabla 1 se describen los productos comerciales similares a nuestro proyecto.

Nombre	Descripción
Ninja Sphere [2]	Es un proyecto desarrollado por la empresa australiana Ninja Blocks que terminó su fase de recaudación de fondos en la página Kickstarter en enero de 2014. Está orientado a domótica y consiste en los siguientes elementos: • Spheramid: este dispositivo permite controlar otros dispositivos conectados a la red y muestra información relevante de su estado en su matriz de leds, además de enviar información a un dispositivo móvil (teléfono o reloj). • Waypoint: este pequeño dispositivo permite al sistema localizar diversos objetos en la casa. • Socket: este dispositivo contiene un relevador y un medidor de energía, controla los dispositivos conectados a él. Este sistema utiliza las tecnologías ZigBee, Bluetooth y Wi-Fi para comunicarse. Una vez que ha sido conectado, el sistema monitorea temperatura, iluminación, uso de energía, presencia de mascotas, entre otras y aprende las condiciones habituales del lugar en el que está y las preferencias del usuario para así poder notificar cuando algo ha cambiado y permitir al usuario realizar acciones para controlar sus dispositivos.

Cheetah Xi [3]	Es un sistema que incorpora alarmas contra incendio, supresión con agente limpio, supresión con dióxido de carbono, agua nebulizada y sistema de riego. Consiste en un panel de control central interconectado con diversos sistemas como: • Cortes de electricidad. • Sistemas de voceo de evacuación. • Puertas. • Descarga de agente limpio, dióxido de carbono, agua nebulizada, espuma y sistema sprinkler, o pre-acción. • Control de humo. • Elevadores. • Apagadores. • Sistemas de seguridad. Cada uno de estos sistemas se conecta a un dispositivo capaz de generar información por medio de sensores inteligentes que indican que este dispositivo está en alarma y la localización del riesgo. Estos dispositivos pueden comunicarse entre ellos antes de conectarse al panel del control.
Waspmote [4]	Es un módulo desarrollado por Libelium para trabajar con sensores en toda clase de aplicaciones: detección de emisiones y contaminación, control de procesos químicos, detección de incendios forestales, sistemas de seguridad y emergencias, detección de ruido y calidad del aire, asistencia en estacionamiento, sistemas de irrigación de agricultura y casas verdes, medición de niveles de radiación, medición de consumo de agua y luz, etc. Se le pueden conectar accesorios como Wi-Fi, Bluetooth, GSM/GPRS, 3G, RFID/NFC, GPS para comunicarlo con una computadora y así monitorear las mediciones.

Tabla 1. Productos comerciales

En la Tabla 2 se detallan algunos artículos académicos sobre sistemas de prevención de incendios.

Título	Descripción
Smart and Automate Fire and Power Monitoring System [5]	Este artículo presenta un sistema que engloba la detección de incendios, la medición de consumo de energía eléctrica y la calidad de la energía eléctrica, tanto en el ámbito doméstico como en el industrial. El sistema es capaz de detectar incendios y enviar alarmas vía SMS a ciertos receptores, así como de distinguir si el fuego generado es eléctrico o no eléctrico.
Simple Schemes of Remote Alerting by Fire and Intruder Detection: A Proposal Favoring Home Security [6]	Este artículo presenta una propuesta para alertar a las personas que hay algún peligro en sus casas, tanto de incendio como de intrusión, a través de llamadas telefónicas automáticas. También contempla el aviso de dichos sucesos a sistemas de emergencia externos (bomberos, policía).

Network-based Fire – Detection System via Controller Area Network for Smart Home Automation [7]	Este artículo presenta un sistema de detección de incendios basado en una red de área de campus, para evaluar la viabilidad de utilizar un protocolo de automatización para el hogar en una casa inteligente. Un sistema de detección de incendio tiene diversas desventajas, tal como la debilidad al ruido. Este artículo presenta una nueva estructura de red para un sistema de este tipo.
Multi-criteria fire detection systems using a probabilistic neural network [8]	Este artículo presenta el proyecto Navy program, Damage Control Automation for Reduced Manning (DC-ARM) que pretende mejorar sus sistemas de detección de incendios, para lograr esto realizan una serie de pruebas con un sistema de detección basado en análisis multi-criterio, utilizando redes neuronales probabilísticas para realizar una detección temprana con base en las mediciones de cinco sensores.

Tabla 2. Artículos académicos

En la Tabla 3 se describen trabajos terminales de la Escuela Superior de Cómputo que tienen relación con nuestro proyecto.

Título	Descripción
Sistema de control para el hogar incorporando tecnología avanzada: CHITA (TT – 20060092)	Es un prototipo de un sistema domótico, el cual controla las puertas, ventanas, sistema de riego y luces de una casa de forma local a través de comandos de voz y de manera remota por medio de una página Web.
Sistema de monitoreo remoto y seguridad integral de inmuebles (TT – 20000226)	Es un sistema que manipula las comodidades, tareas y servicios de seguridad, tareas y servicios de seguridad requeridas en el hogar, tales como abrir el portón del garaje, encender la calefacción, así como el encendido y apagado de luces, mediante una computadora y haciendo uso de tecnología inalámbrica y dispositivos móviles.
Prototipo de sistema de monitoreo y control remoto de un inmueble por medio de un dispositivo móvil (TT – 20060070)	Es un sistema el cual se encarga de monitorear y controlar dispositivos de seguridad colocados en un inmueble de manera local y remota. Su funcionamiento se basa en el uso de una tarjeta de adquisición de datos conectada en una PC y en la generación de una base de datos que reflejara el estado de los dispositivos conectados en el inmueble.
Sistema para la administración de un invernadero (TT – 20060096)	Es un sistema de apoyo para agricultores como herramienta para el control de la humedad y la temperatura de un invernadero de manera remota a través de Internet. Con esta aplicación el usuario podrá llevar un control de la producción de los cultivos del invernadero.

Tabla 3. Trabajos terminales

CAPÍTULO II: MARCO TEÓRICO

2.1. Incendio en Casa Habitación

2.1.1. Fuego

El fuego es una reacción química que consiste en la oxidación violenta de un material combustible; se manifiesta con desprendimientos de luz, calor, humos y gases en grandes cantidades [1]. Es representado simbólicamente con un triángulo equilátero (Figura 1) que describe los tres elementos necesarios para que exista.

Figura 1. Triángulo del fuego

El oxígeno es el primer componente del triángulo del fuego, es un gas no inflamable que se encuentra en la atmósfera en una proporción del 21%, sin embargo, para que se inicie una combustión con flama debe existir como mínimo un 16% de oxígeno en el aire [9].

El calor es el segundo componente del triángulo del fuego, es la energía que se necesita para aumentar la temperatura del combustible, al punto que desprenda suficientes vapores y ocurra la ignición. El calor, por lo tanto, puede ser definido como una fuerza térmica que eleva la temperatura de los cuerpos hasta hacerlos gasificar, volatilizar o dilatar y es el principal causante de la propagación de un incendio [9].

La tercera parte del triángulo del fuego es el combustible o agente reductor, que puede ser sólido, líquido o gaseoso. Para que se inicie la combustión de cualquier material combustible (a excepción del estado gaseoso), este debe sufrir un cambio de estado hasta convertirse en vapor, que en una proporción adecuada con el aire atmosférico y una fuente de calor presente que inicie o mantenga la reacción, dará lugar al fuego [9].

La reacción única, directa y simultánea de los tres elementos del triángulo es la causa del inicio de la reacción de combustión. Si falta alguno de estos tres elementos, o si no están en la proporción o combinación adecuada, el fuego no podrá existir [9].

El fuego puede propagarse por tres medios (o combinaciones entre ellos). Esos medios son: radiación, convección y conducción.

La radiación es el desplazamiento de ondas de calor, partiendo de un fuego, a una materia próxima [10].

La convección es el desplazamiento de los gases y aire calientes. Cuando se calienta el aire (o cualquier otro gas), este se dilata y se vuelve más ligero, esto hace que ascienda. El humo y los gases calientes que se engendran en un fuego suben rápidamente, calentando todos los objetos que están por encima, pudiendo llegar a su temperatura de ignición y arder [10].

La conducción es el avance del calor a través de una sustancia. De forma muy general podemos decir que los metales son buenos conductores del calor y que el hormigón, las piedras o ladrillos son malos conductores [10].

En la Tabla 4 se detallan los siniestros relacionados con fuego.

Nombre	Descripción
Incendios estructurales	Se originan en edificaciones como viviendas unifamiliares, residenciales, departamentos, oficinas, comercios, industrias, almacenes, centros de recreación, de hospedaje, religiosos, etc. También se consideran los incendios en vehículos terrestres y aéreos.
Incendios forestales	Se originan en reservas y parques ecológicos, áreas protegidas, zonas de barrancas, etc.
Incendios de pasto y basura	Se originan en camellones, predios baldíos, casas abandonadas, llanos, etc.
Flamazos (deflagraciones)	Se originan por acumulación de gases, vapores o polvos combustibles en casas habitación, comercios e industrias.
Explosiones	Son ocasionadas por calderas, boilers, tanques de gas, en industrial y comercios, etc.
Fugas de gas	Éstas pueden ser: • Fugas de gas en instalaciones domésticas, comerciales e industriales. • Fugas de gas LP (licuado de petróleo) en cilindros portátiles, tanques estacionarios, tanques de carburación o plantas de almacenamiento. • Fugas de gas natural en redes de distribución y tuberías de alimentación domiciliaria.
Sustancias peligrosas	Dentro de esta categoría se encuentran las fugas o derrames de sustancias inflamables, tóxicas, oxidantes o corrosivas en comercios, depósitos o industrias. También se considera la generación de gases tóxicos por mala combustión que pongan en riesgo la vida humana.
Mezclas inflamables	Dentro de esta categoría se encuentran las sustancias líquidas o sólidas, inflamables, tóxicas, oxidantes o corrosivas, derramadas y/o esparcidas en la vía pública. También se consideran los residuos líquidos de los cilindros y tanques para gas LP, vertidos en la vía pública, drenajes, terrenos baldíos, patios o azoteas.

Tabla 4. Siniestros relacionados con el fuego [11]

El presente trabajo se enfoca en los incendios estructurales (específicamente en casas habitación) y en la detección de fugas de gas como una medida de prevención de incendios.

2.1.2. INCENDIO

Podemos definir un incendio como el fuego no controlado de grandes proporciones que puede presentarse en forma súbita, gradual o instantánea, al que le siguen daños materiales que pueden interrumpir el proceso de producción, ocasionar lesiones o pérdida de vidas humanas y deterioro ambiental [1].

Para que un incendio se sostenga y se propague, es necesario que exista un cuarto factor que complementa el triángulo del fuego y es la reacción química en cadena que se produce entre el combustible y el oxígeno con ayuda del calor. A la figura multidimensional que se forma con este nuevo elemento se le conoce como el tetraedro del fuego [9].

A medida que el fuego arde, las moléculas del combustible se reducen dentro de la llama; mientras el proceso de combustión continúa, el aumento de temperatura provoca que el oxígeno sea atraído al área de flama facilitando la ruptura de moléculas y con ello, éstas entran en reacción alcanzando su punto de ignición, aumentando la temperatura, lo cual a su vez demanda más oxígeno, lo que permite que continúe de esta manera la reacción en cadena a nivel molecular. Este proceso seguirá hasta que las sustancias involucradas se trasladen hacia áreas más frías de la llama [9].

Los incendios se han clasificado para indicar la naturaleza de los materiales que arden y en consecuencia, tratar de extinguirlos con el agente más efectivo. En la Tabla 5 se muestra la clasificación:

Nombre	Origen	Método de extinción
Tipo "A"	Materiales que contienen carbono y que pueden ser: madera, papel, basura, tela, algunos tipos de plásticos, etc.	Agua y extintores con base en polvo químico seco.
Tipo "B"	Sólidos inflamables, etanol, metano, gasolina, aguarrás, thiner, alcohol y los gases derivados del petróleo: propano o butano y natural o metano.	Extintores que contienen CO_2, PQS (polvo químico seco) o AFFF (espuma química).
Tipo "C"	Se producen a partir de la corriente eléctrica y su mecanismo no es una combustión sino ignición.	Extintores con base de PQS, gas halón (hidrocarburo halogenado).
Tipo "D"	Producidos por metales como el sodio, potasio, magnesio, litio y aluminio.	Bicarbonato de potasio K_2CO_3, arenas secas, PQS y nunca agua o extintores a base de CO_2 ya que pueden ocasionar reacciones exotérmicas.

Tabla 5. Clasificación de los incendios [1]

Es frecuente que alguna de las clases antes mencionadas se desarrolle en presencia de corriente eléctrica, como en el caso de incendios de aparatos electrodomésticos y cables eléctricos. En estos casos, al peligro que representa el fuego, se añade el riesgo de

electrocución, por lo que al intentar apagar el fuego se debe considerar esta posibilidad y tomar medidas protectoras oportunas, tales como desconectar el suministro eléctrico [10].

Otra clasificación de los incendios se hace con base en su extensión: [10]

- Conato: el conato de incendio es un fuego que se inicia y puede ser controlado sin mayores dificultades, no representa gran peligro si se maneja a tiempo mediante el uso de extintores portátiles, acción que puede ser realizada sin personal especializado.
- Incendio parcial: es un fuego que abarca parcialmente una instalación o un área geográficamente determinada, tiene la posibilidad de salir de control y causar víctimas o daños mayores, los extintores portátiles frecuentemente son útiles para sofocar estos incendios y se requiere la participación de personal entrenado y equipado.
- Incendio total: es un incendio completamente fuera de control y de alta destructividad, afecta a toda una instalación o área difícil de combatir directamente, en consecuencia deben protegerse vidas y bienes de los alrededores e incluso evacuar la zona.

La combustión en los incendios produce gases. La peligrosidad de estos gases depende de la concentración de los mismos en el aire, el tiempo que dura la inhalación y las condiciones físicas de la persona. Estas condiciones varían a causa del propio incendio, ya que el ritmo respiratorio aumenta debido a la tensión nerviosa, el calor, el esfuerzo y el exceso de anhídrido carbónico. Los principales gases producidos por la combustión son: [10]

- Monóxido de carbono (CO): este gas es tóxico y actúa principalmente al ser inhalado, siendo clasificado como asfixiante químico. La presencia de monóxido de carbono en la sangre impide que el oxígeno del aire se combine con la hemoglobina, ya que esta tiene una afinidad 200 a 300 veces mayor por el monóxido de carbono que por el oxígeno. El grave peligro que presenta una exposición a grandes cantidades de monóxido de carbono radica en que no hay advertencia alguna, como olor o color y existen pocos síntomas iniciales antes de la inconsciencia o la muerte.
- Dióxido de carbono (CO_2): el dióxido de carbono o anhídrido carbónico se produce en los incendios y es mortal en grandes concentraciones.
- Otros gases de la combustión: en los incendios no solo se producen óxidos de carbono, sino muchos otros gases que dependen del tipo de material, cantidad de oxígeno y temperatura. Los más comunes pueden ser los compuestos de azufre, cloro y nitrógeno.

El presente trabajo se enfoca en la detección de monóxido de carbono por ser uno de los productos más peligrosos de la combustión, ya que su inhalación no provoca síntomas en el ser humano hasta que es prácticamente fatal (por esta razón es conocido como el “asesino silencioso”).

2.2. Valores de Riesgo

En un incendio, no solo las llamas son un riesgo para la salud y la vida de una persona, a continuación se presentan los valores de riesgo de diversas variables que intervienen en un incendio:

- Temperatura: el aumento de la temperatura ambiente puede producir un shock térmico, causando la muerte si se sobrepasan los 41°C [10].
- Monóxido de carbono: la concentración de este gas comienza a producir síntomas en cuerpo a partir de las 200 ppm, estos síntomas pueden aumentar su intensidad dependiendo del tiempo de exposición y de la concentración del gas como se muestra en la Tabla 6.

Concentración	Síntomas
50 ppm	Sin efectos adversos durante las primeras 8 horas de exposición.
200 ppm	Leve dolor de cabeza después de 2 a 3 horas de exposición.
400 ppm	Dolor de cabeza y náuseas después de 1 a 2 horas de exposición.
800 ppm	Dolor de cabeza, náuseas y mareo después de 45 minutos; colapso e inconsciencia después de 1 hora de exposición.
1 000 ppm	Pérdida de consciencia después de una hora de exposición.
1 600 ppm	Dolor de cabeza, náuseas y mareo después de 20 minutos de exposición.
3 200 ppm	Dolor de cabeza, náuseas y mareo después de 5 a 10 minutos; colapso e inconsciencia después de 30 minutos de exposición.
6 400 ppm	Dolor de cabeza y mareo después de 1 a 2 minutos; inconsciencia y peligro de muerte después de 10 a 15 minutos de exposición.
12 800 ppm	Efectos fisiológicos inmediatos, inconsciencia y peligro de muerte después de 1 a 3 minutos de exposición.

Tabla 6. Síntomas por envenenamiento de CO [12]

- Gas licuado de petróleo (GLP): en altas concentraciones (más de 1000 ppm), el gas licuado es un asfixiante simple, ya que diluye el oxígeno disponible para respirar. Los efectos de una exposición prolongada pueden ser: dolor de cabeza, náuseas, vómito, tos, signos del sistema nervioso central, dificultad al respirar, mareo, somnolencia y desorientación. En casos extremos pueden presentarse convulsiones, inconsciencia, incluso la muerte como resultado de la asfixia [13].

2.3. Sistema de Detección de Incendios y de Alarma

Un sistema de detección de incendios y de alarma permite detectar un incendio de forma automática y avisar a los ocupantes del edificio de la amenaza. La alarma sonora o visible de un sistema de este tipo es la primera señal que reciben los ocupantes de un edificio para iniciar la evacuación [14].

Un sistema de detección de incendios y de alarma puede incluir todos o algunos de los elementos siguientes: [14]

- Una unidad de control del sistema.
- Un suministro primario o principal de energía eléctrica.
- Un suministro secundario de energía (stand-by), normalmente alimentado por baterías o por un generador de emergencia.
- Dispositivos de activación de la alarma, como detectores automáticos de incendios, pulsadores manuales y/o dispositivos de flujo de sistemas de rociadores, conectados a circuitos de activación de la unidad de control del sistema.
- Dispositivos de alarma, como timbres o luces, conectados a circuitos indicadores de la unidad de control del sistema.
- Controles auxiliares, como funciones de apagado de la ventilación, conectados a circuitos de salida de la unidad de control del sistema.
- Alarmas conectadas a un centro de emergencia externo, como el centro de bomberos.
- Circuitos de control para activar un sistema de protección contra incendios o un sistema de control de humos.

En la Figura 2 se muestra un ejemplo de instalación de un sistema de detección de incendios en una casa. Como se puede observar los módulos del sistema se instalan en el techo de cada habitación, estos módulos cuentan con detectores de humo y rociadores. Además, se colocan extintores de incendios en diversas habitaciones como medida adicional de seguridad.

Figura 2. Casa con sistema de detección de incendios [15]

Los dispositivos de activación de alarma requieren de sensores y los circuitos de activación del sistema de protección contra incendios requieren de actuadores. Los sensores y actuadores son los elementos de un sistema que lo conectan con su entorno físico [16]. A continuación se describen estos elementos.

2.4. SENSORES

La función de los sensores es obtener señales eléctricas en respuesta a magnitudes de entrada no eléctricas [16].

Según el aporte de energía, los sensores se pueden dividir en moduladores y generadores. En los sensores moduladores o activos, la energía de la señal de salida procede, en su mayor parte, de una fuente de energía auxiliar. La entrada sólo controla la salida. En los sensores generadores o pasivos, en cambio, la energía de salida es suministrada por la entrada [17].

Según la señal de salida, los sensores se clasifican en analógicos y digitales. En los analógicos la salida varía de forma continua y la información está en la amplitud. En los sensores digitales, la salida varía en forma de saltos o pasos discretos por lo que no requieren conversión A/D y la transmisión de su salida es más fácil [17].

Atendiendo al modo de funcionamiento, los sensores pueden ser de deflexión o de comparación. En los sensores que funcionan por deflexión, la magnitud medida produce algún efecto físico, que engendra algún efecto similar, pero opuesto, en alguna parte del instrumento, y que está relacionado con alguna variable útil. En los sensores que funcionan por comparación, se intenta mantener nula la deflexión mediante la aplicación de un efecto bien conocido, opuesto al generado por la magnitud a medir [17].

Según el tipo de relación entrada-salida, los sensores pueden ser de orden cero, de primer orden, de segundo orden o de orden superior. El orden está relacionado con el número de elementos almacenadores de energía independientes que incluye el sensor, y repercute en su exactitud y velocidad de respuesta. Esta clasificación es de gran importancia cuando el sensor forma parte de un sistema de control de lazo cerrado [17].

Desde el punto de vista de la ingeniería electrónica, es más atractiva la clasificación de los sensores de acuerdo con la magnitud y el parámetro variable. En la Figura 3 se presentan ambas clasificaciones [17].

Figura 3. Clasificación de los sensores por magnitud y parámetro variable

2.5. ACTUADORES

La función de los actuadores es realizar una acción mecánica en respuesta a una señal de entrega, que en el caso de los actuadores eléctricos es eléctrica, pero que puede ser también neumática, hidráulica o mecánica [16].

Dentro de los actuadores electromecánicos se encuentran los llamados relés o relevadores. Un relé es un interruptor mecánico accionado eléctricamente. Consta de dos o más contactos, correspondientes a uno o varios circuitos independientes, y de un elemento que controla la conmutación. La corriente y tensión necesarias para el control es muy pequeña respecto a la potencia del circuito controlado. Ello permite que el interruptor que cierra el circuito de control, denominado interruptor de control, sea de potencia inferior a la necesaria en un interruptor intercalado directamente en el circuito controlado [16].

Las características de los relés se pueden dividir en tres grupos según sean relativas al contacto, a la conmutación, o al elemento de control.

Algunas características del contacto son: la resistencia del contacto cuando está cerrado, la resistencia y capacidad entre contactos cuando está abierto, la máxima potencia, tensión y corriente que pueden conmutar, la corriente máxima que puede conducir, y la máxima tensión entre contactos que puede aguantar [16].

Las características de la conmutación se refieren, por una parte, a la vida esperada y, por otra, a la dinámica de la conmutación. Esta última se caracteriza por los tiempos de cierre y apertura, por los tiempos que duran los rebotes mecánicos respectivos, especificados para la tensión nominal en el elemento de control, y por la velocidad de repetición del ciclo cierre-apertura, limitada, aparte de los rebotes, por el aumento de temperatura del relé [16].

El elemento de control viene caracterizado por su resistencia e inductancia eléctricas, su tensión nominal, su consumo al accionar el interruptor y para mantenerlo accionado, y los valores de tensión capaces de provocar, respectivamente, la apertura y cierre del circuito controlado [16].

CAPÍTULO III: ANÁLISIS DEL SISTEMA

3.1. Tecnologías de Desarrollo

3.1.1. Sistemas Operativos Móviles

Un sistema operativo móvil es un sistema operativo orientado a dispositivos móviles. A diferencia de los sistemas operativos para computadoras, éstos son más simples debido a que los dispositivos móviles tienen menos recursos de hardware, están orientados a la conexión inalámbrica y al manejo, envío y recepción de archivos multimedia y datos en general.

Algunos ejemplos de sistemas operativos móviles son: Android, iOS, Windows Phone, Symbian y BlackBerry OS.

Para el sistema operativo del móvil, tomamos en cuenta las características de cada sistema operativo (Figura 4) y la cantidad de usuarios que los utilizan (Figura 5).

	Apple iOS 6	**Android 4.2**	**Windows Phone 7**	**BlackBerry OS 7**	**Symbian 9.5**
Compañía	Apple	Open Handset Alliance	Windows	RIM	Symbian Foundation
Núcleo del SO	Mac OS X	Linux	Windows CE	Mobile OS	Mobile OS
Familia CPU soportada	ARM	ARM, MIPS, Power, x86	ARM	ARM	ARM
Lenguaje de programación	Objective-C, C++	Java, C++	C#, muchos	Java	C++
Licencia de *software*	propietaria	*software* libre y abierto	Propietaria	propietaria	*software* libre
Año de lanzamiento	2007	2008	2010	2003	1997
Motor del navegador web	WebKit	WebKit	Pocket Internet Explorer	WebKit	WebKit
Soporte Flash	No	Sí	No	Si	Sí
HTML5	Sí	Sí	Sí	Sí	No
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlackBerry App World	Ovi Store
Número de aplicaciones	400.000	300.000	50.000	30.000	50.000
Coste publicar	$99 / año	$25 una vez	$99 / año	sin coste	$1 una vez
Plataforma de desarrollo	Mac	Windows, Mac, Linux	Windows	Windows, Mac	Windows, Mac, Linux
Actualizaciones automáticas del S.O.	Sí	depende del fabricante	depende del fabricante	Sí	Sí
Soporte memoria externa	No	Sí	No	Sí	Sí
Fabricante único	Sí	No	No	Sí	No
Variedad de dispositivos	modelo único	muy alta	baja	baja	muy alta
Tipo de pantalla	capacitativa	capacitiva /resistiva	capacitativa	/resistiva capacitativa	capacitiva /resistiva
Aplicaciones nativas	Sí	Sí	No	No	Sí

Figura 4. Comparativa de sistemas operativos móviles [18]

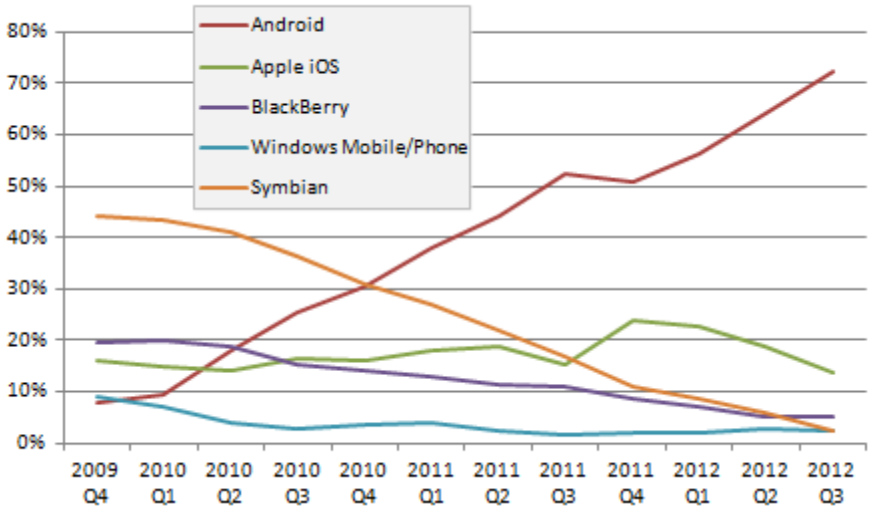


Figura 5. Cuota de mercado de sistemas operativos móviles en el mundo [18]

Considerando las características que ofrece Android y tomando en cuenta que es el sistema operativo más utilizado (hasta el 2012) decidimos utilizarlo como base para desarrollar nuestra aplicación.

3.1.1.1. ANDROID

Es un sistema operativo basado en Linux, desarrollado inicialmente por Android Inc. (que fue comprado por Google en 2005) y que fue diseñado principalmente para dispositivos móviles con pantalla táctil como smartphones y tablets. Android fue develado en 2007 junto con la fundación Open Handset Alliance, un consorcio de compañías para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android se vendió en octubre de 2008.

Este sistema operativo incorpora las siguientes cualidades: [18]

- Plataforma de desarrollo de código abierto.
- Aplicaciones portables desarrolladas en Java.
- Arquitectura basada en componentes inspirados en Internet.
- Filosofía de dispositivo siempre conectado a Internet.
- Gran cantidad de servicios incorporados.
- Programas aislados con permisos que dan un aceptable nivel de seguridad.
- Optimización para baja potencia y poca memoria.
- Alta calidad de gráficos y sonido.

La arquitectura Android está formada por cuatro capas, que son: [18]

- Núcleo Linux: esta capa del modelo actúa como una capa de abstracción entre el hardware y el resto de la pila, por lo que es la única que depende del hardware.

- Runtime de Android: está basado en el concepto de máquina virtual utilizado en Java (pero en una versión para poca memoria y procesador limitado llamada Dalvik), también incluye el "core libraries" con la mayoría de librerías disponibles en lenguaje Java.
- Librerías nativas: incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android, están compiladas en código nativo del procesador y muchas de ellas utilizan proyectos de código abierto.
- Entorno de aplicación: proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones, cuyos servicios más importantes son: Views, Resource Manager, Activity Manager, Notification Manager y Content Providers. A esta capa también pertenece el nivel de Aplicaciones el cual está formado por el conjunto de aplicaciones que son ejecutadas por la máquina virtual Dalvik y que fueron desarrolladas con el Android SDK o con el Android NDK.

En la Figura 6 se muestra cómo está estructurada la arquitectura de Android.

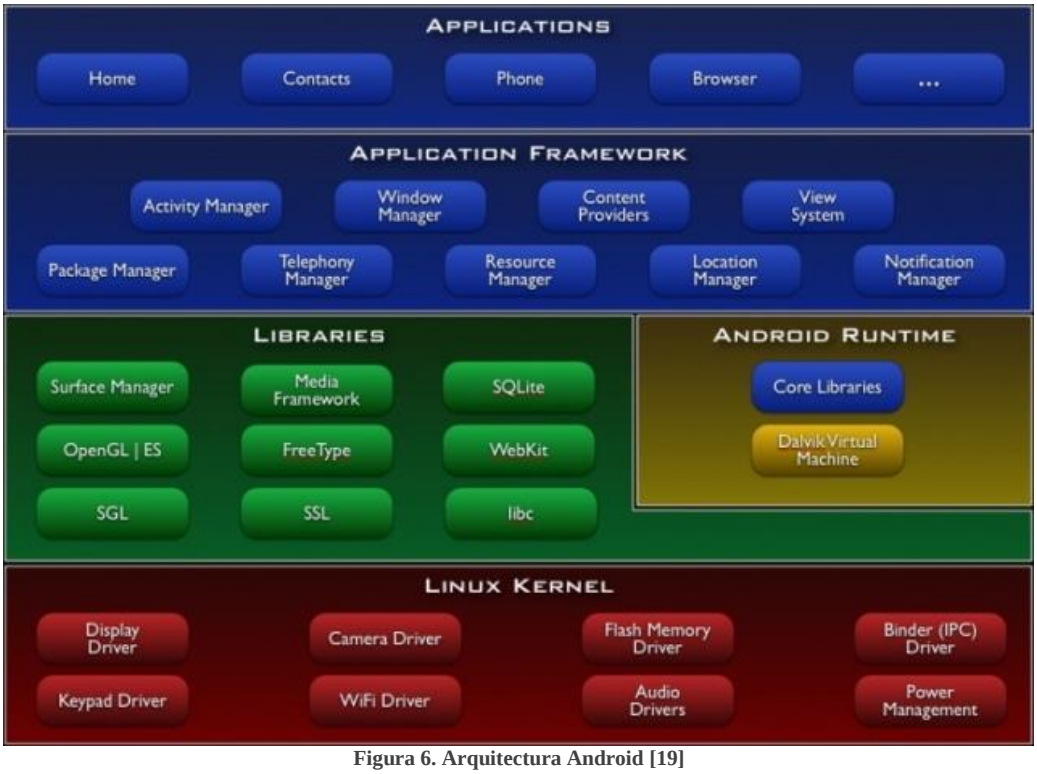


Figura 6. Arquitectura Android [19]

Desde su lanzamiento, Android ha liberado diversas versiones que se ilustran en la Figura 7.

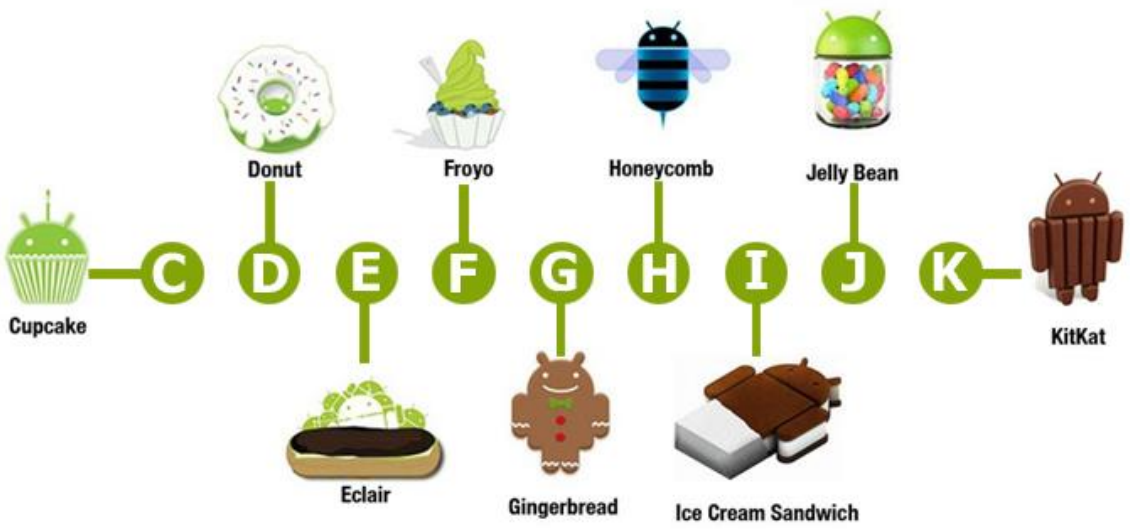


Figura 7. Versiones de Android [20]

En la Tabla 7 se especifican las principales características de cada una de las versiones de Android.

Nombre	Características
Cupcake (v1.5)	Esta versión fue anunciada el 30 de abril de 2009, desde entonces Google comenzó a nombrar sus sistemas operativos como postres. Las principales características de esta versión incluyen soporte para widgets, teclado en la pantalla, Bluetooth, auto rotación, animación de inicio, soporte para AVRCP, funciones para cortar y pegar, cargar fotos y videos y añadió soporte para software de terceras partes.
Donut (v1.6)	Esta versión fue lanzada el 15 de septiembre de 2009. Sus principales características incluyen búsqueda por voz, búsqueda rápida, indicador de batería, lectura de texto en diversos idiomas, soporte para resolución de pantallas WVGA, cambio entre modo de captura y modo quieto y soporte para CDMA.
Éclair (v2.0/2.1)	Esta versión salió el 26 de octubre de 2009. Sus principales características incluyen el navegador beta de Google Maps, un nuevo navegador, soporte para varias cuentas, fondos animados, sincronización de correos y contactos, teclado mejorado, búsqueda de SMS y soporte para Exchange.
Froyo (v2.2)	Esta versión fue lanzada el 20 de mayo de 2010. Sus principales características incluyen funcionalidad para tener un punto de acceso Wi-Fi, mayor cantidad de lenguajes en el teclado y soporte para Adobe Flash.

Gingerbread (v2.3)	Esta versión fue liberada el 6 de febrero de 2010. Sus principales características incluyen mejor control sobre las aplicaciones, mejoras en las funciones cortar y pegar, telefonía por Internet basada en SIP, una API abierta para audio nativo, video chat con Google Talk, etc.
Honeycomb (v3.0)	Esta versión fue lanzada el 22 de febrero de 2011. Sus principales características son la funcionalidad para cifrar todos los datos de usuario, multitarea simplificado, un nuevo teclado, nueva interfaz de usuario virtual y holográfica, acceso rápido a aplicaciones como la cámara.
Ice Cream Sandwich (v4.0)	Esta versión fue liberada el 19 de octubre de 2011. Las mejores características de esta versión son dictado de texto en tiempo real, soporte para NFS, barrido para descartar notificaciones, Wi-Fi Direct, desbloqueo facial y una plataforma consolidada para teléfonos y tablets.
Jelly Bean (v4.1/4.2/4.3)	Esta versión fue lanzada el 9 de julio de 2012. Sus principales características incluyen widgets en la pantalla de bloqueo, soporte para múltiples usuarios y pantallas, audio multicanal y USB y proyecto Butter para mejorar el rendimiento.
KitKat (v4.4)	Esta versión fue liberada el 3 de septiembre de 2013. Sus principales características incluyen función de impresión, detección para soporte de APIs, emulación de cliente NFC y procesamiento de sensores.

Tabla 7. Características de las versiones de Android

3.1.2. COMPUTADORAS DE PLACA ÚNICA

Una computadora de placa única o computadora de placa reducida (Single Board Computer) es una computadora completa en un solo circuito. El diseño se centra en un solo microprocesador con memoria, puertos de entrada y salida y todas las demás características de una computadora funcional en una sola tarjeta que suele ser de tamaño reducido, y que tiene todo lo que necesita en la placa base. Esta arquitectura no es muy utilizada en computadoras personales (aunque las tendencias indican que esto puede cambiar), sino en entornos industriales o en sistemas embebidos.

Para elegir la computadora a utilizar tomamos en cuenta las características más significativas que ayudarán al desarrollo de nuestro sistema, como se muestra en la Tabla 8.

		Cubieboard	A13-OLinuXino	Raspberry Pi	BeagleBone
Imagen					
Precio		US$54.00 (Envío incluido)	€58.00 (Sin envío)	US$35.00 ($25 - modelo A) (Sin envío)	US$89.00 (Envío incluido)
CPU	**Tipo de socket**	Allwinner A10	Allwinner A13	Broadcom BCM2835	TI AM3359
	Tipo de núcleo	Cortex-A8	Cortex-A8	ARM1176JZF-S	
	Frecuencia	1 GHz	1 GHz	700 MHz	720 MHz
Memoria	**RAM**	1 GB	512 MB	512 MB (256 – modelo A)	256 MB
	Flash	4 GB	0 MB	0 MB	0 MB
Conectividad	**Puertos USB**	2	3	2	1
	Ethernet	✓	opcional	✓	✓
	Puerto SATA	✓	✖	✖	✖
	SPI	✓	✓	✓	✓
	I2C	✓	✓	✓	✓
	GPIO	✓	✓	✓	✓
	LCD Panel	✓	✓	✓	✓
	SD/MMC	✓	✓	✓	✓
	Serial	✓	✓	opcional	✓
	Wi-Fi	✖	opcional	✖	✖
	Bluetooth®	✖	✖	✖	✖

Tabla 8. Comparativa de computadoras de placa única [21]

Después de realizar la comparación se decidió utilizar la tarjeta de desarrollo Cubieboard porque es la tarjeta que más se adecua a las necesidades del sistema ya que se puede utilizar un adaptador USB para agregar la conectividad Wi-Fi, tiene mayor cantidad de memoria y se le puede instalar cualquier distribución de Linux diseñada para microprocesadores ARM.

3.1.2.1. CUBIEBOARD

Cubieboard es una pequeña tarjeta de arquitectura ARM desarrollada por CubieTech con un procesador Allwinner A10 [22].

Sus características y especificaciones son muy similares a las de un smartphone, con la diferencia de que es un sistema abierto y posee distintos tipos de conexión cableada para guardar datos, conectarse a Internet, utilizar periféricos como pantallas, teclados, mouse, etc.

La Figura 8 muestra los componentes de una Cubieboard.

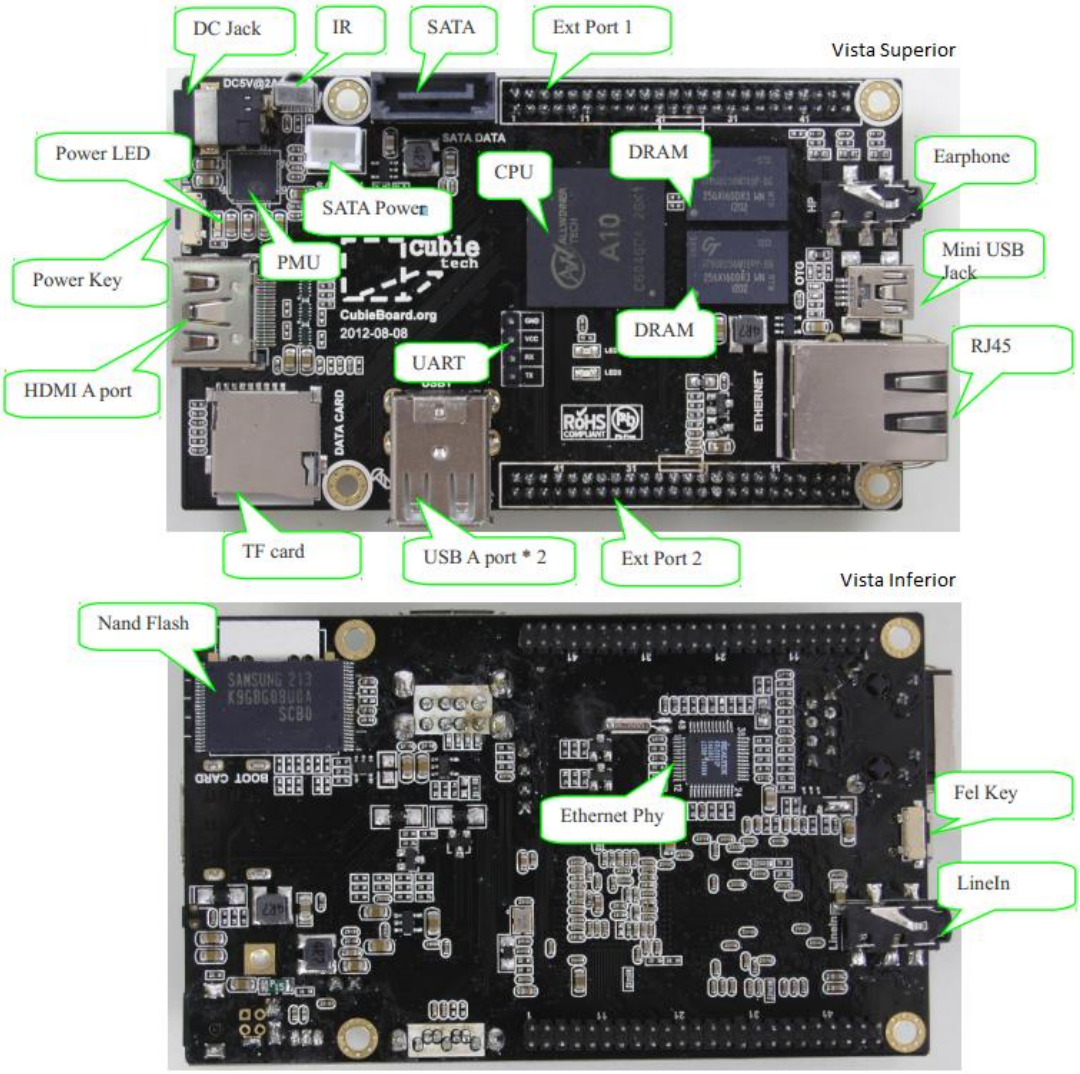


Figura 8. Componentes de una Cubieboard [22]

3.1.3. Sistemas Operativos para Microprocesadores ARM

La arquitectura ARM es una arquitectura RISC (Reduced Instruction Set Computer) de 32 bits desarrollada por ARM Holdings y utilizada principalmente en la electrónica de consumo, como por ejemplo: PDA's, tablets, smartphones, consolas de videojuego portátiles, calculadoras, reproductores de medios y periféricos de computadoras.

Existe una gran variedad de sistemas operativos para dispositivos con un microprocesador ARM y a diferencia de los sistemas operativos para móviles estos no solo están orientados a las conexiones inalámbricas y a la administración y gestión de recursos multimedia sino que también pueden realizar conexiones alámbricas y administrar y gestionar recursos de cualquier tipo al igual que en una computadora común y corriente.

Existen diversas distribuciones que recomienda la página de Cubieboard, entre ellas se encuentran Android y Lubuntu (una distribución de Ubuntu). Sin embargo, el sistema operativo que se instaló en la Cubieboard es Cubian, una distribución de Debian desarrollado para esta tarjeta y que, aunque no es la distribución más difundida, es una muy buena opción una vez que se instala adecuadamente y se le añade el modo gráfico (por defecto Cubian solo tiene modo consola). Esta distribución permite tener acceso a todos los pines de entrada y salida de la Cubieboard, así como la instalación de los paquetes y programas que se necesitan para el desarrollo de este proyecto.

3.1.4. Tarjetas de Desarrollo

Son tarjetas donde el usuario puede realizar de forma rápida y cómoda todo tipo de experimentos, a continuación se describe la más comercial que es Arduino.

3.1.4.1. Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos [23].

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino, basado en Wiring y el entorno de desarrollo Arduino, basado en Processing. Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a una computadora, no obstante tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software como Flash, Processing, MaxMSP, etc [23].

Las placas pueden ser hechas a mano o compradas (montadas de fábrica), el software puede ser descargado de forma gratuita. Los archivos de diseño de referencia (CAD) están disponibles bajo una licencia abierta, así pues el usuario es libre de adaptarlos a sus necesidades [23].

En la Tabla 9 se muestra una comparación entre las diversas placas Arduino.

Nombre	Procesador	Voltaje de Operación / Voltaje de Entrada (V)	Velocidad del CPU (MHz)	E/S Analógicas	ES/PWM Digitales	EEPROM (KB)	SRAM (KB)	Flash (KB)	USB	UART
Uno	ATmega328	5 / 7 – 12	16	6 / 0	14 / 6	1	2	32	Regular	1
Due	AT91SAM3X8E	3.3 / 7 – 12	84	12 / 2	54 / 12	-	96	512	2 Micro	4
Leonardo	ATmega32u4	5 / 7 – 12	16	12 / 0	20 / 7	1	2.5	32	Micro	1
Mega 2560	ATmega2560	5 / 7 – 12	16	16 / 0	54 / 15	4	8	256	Regular	4
Mega ADK	ATmega2560	5 / 7 – 12	16	16 / 0	54 / 15	4	8	256	Regular	4
Micro	ATmega32u4	5 / 7 – 12	16	12 / 0	20 / 7	1	2.5	32	Micro	1
Mini	ATmega328	5 / 7 – 9	16	8 / 0	14 / 6	1	2	32	-	-
Nano	ATmega168 ATmega328	5 / 7 – 9	16	8 / 0	14 / 6	0.512 1	1 2	16 32	Mini-B	1
Ethernet	ATmega328	5 / 7 - 12	16	6 / 0	14 / 4	1	2	32	Regular	-
Esplora	ATmega32u4	5 / 7 – 12	16	-	-	1	2.5	32	Micro	-
Arduino BT	ATmega328	5 / 2.5 – 12	16	6 / 0	14 / 6	1	2	32	-	1
Fio	ATmega328P	3.3 / 3.7 – 7	8	8 / 0	14 / 6	1	2	32	Mini	1
Pro (168)	ATmega168	3.3 / 3.35 - 12	8	6 / 0	14 / 6	0.512	1	16	-	1
Pro (328)	ATmega328	5 / 5 – 12	16	6 / 0	14 / 6	1	2	32	-	1
Pro Mini	ATmega168	3.3 / 3.35 - 12 5 / 5 – 12	8 16	6 / 0	14 / 6	0.512	1	16	-	1
LilyPad	ATmega168V ATmega328V	2.7 - 5.5 / 2.7-5.5	8	6 / 0	14 / 6	0.512	1	16	-	-
LilyPad USB	ATmega32u4	3.3 / 3.8 – 5	8	4 / 0	9 / 4	1	2.5	32	Micro	-
LilyPad Simple	ATmega328	2.7 - 5.5 /2.7 - 5.5	8	4 / 0	9 / 4	1	2	32	-	-
LilyPad SimpleSnap	ATmega328	2.7 - 5.5 /2.7 - 5.5	8	4 / 0	9/4	1	2	32	-	-

Tabla 9. Comparación de tarjetas Arduino [23]

Con base en la tabla anterior, se decidió utilizar la Arduino Uno para desarrollar los módulos tanto de sensores como de actuadores, ya que cuenta con las características necesarias para este proyecto. Se descartó la Arduino Fio (aunque tiene integrado el adaptador para Xbee) porque su voltaje de operación es 3.3 V y tanto los sensores como el relé trabajan con 5 V.

En la Figura 9 se muestra la vista superior de una tarjeta Arduino Uno.

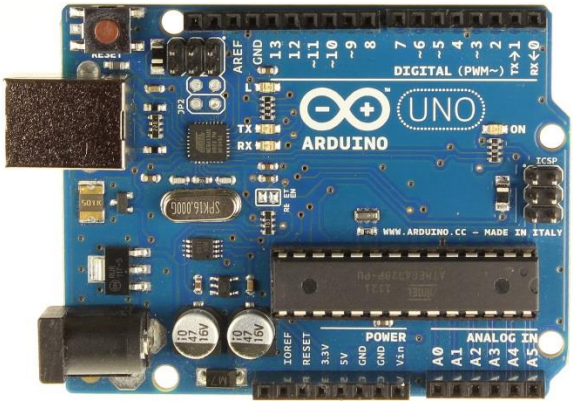


Figura 9. Vista superior de tarjeta Arduino Uno [23]

3.2. Tecnologías de Comunicación

A continuación se describen las tecnologías de comunicación que se utilizarán en el presente trabajo terminal que son Wi-Fi para la comunicación entre la Cubieboard y el dispositivo móvil y ZigBee para la comunicación entre los módulos y la Cubieboard.

3.2.1. Wi-Fi

Se llaman redes inalámbricas de área local, WLAN (Wireless Local Area Network), a aquellas redes que tienen una cobertura de unos cientos de metros. Estas redes están pensadas para crear un entorno local entre computadoras o terminales situadas en un mismo edificio o grupo de edificios. En el mercado existen distintas tecnologías que dan respuesta a esta necesidad. Entre estas tecnologías están las siguientes: Wi-Fi, HiperLan, HiSWAN [24].

Durante bastantes años, las redes inalámbricas de computadoras se llevaban a cabo utilizando soluciones particulares de cada fabricante. Estas soluciones, llamadas propietarias, tenían el gran inconveniente de no permitir interconectar equipos de distintos fabricantes [24].

La única forma de resolver este problema es desarrollar un sistema normalizado que sea aceptado por todos los fabricantes como sistema común. Este sistema es el propuesto por la asociación WECA (Wireless Ethernet Compatibility Alliance, Alianza para la compatibilidad Ethernet inalámbrica) y normalizado por la IEEE con el estándar 802.11b. A esta norma se le conoce más habitualmente como Wi-Fi o Wireless Fidelity (Fidelidad Inalámbrica) [24].

Con el sistema Wi-Fi se pueden establecer comunicaciones a una velocidad máxima de 11Mbps, alcanzándose distancias de hasta varios cientos de metros. No obstante, versiones más recientes de esta tecnología permiten alcanzar los 22, 54 y hasta 100 Mbps [24].

Una red Wi-Fi puede estar formada por dos computadoras o por miles de ellas. Para que una computadora pueda comunicarse de forma inalámbrica necesita tener instalado un adaptador de red. Un adaptador de red es un equipo de radio (con transmisor, receptor y antena) que puede ser insertado o conectado a una computadora, PDA (personal digital assistant, asistente digital personal) o cualquier equipo susceptible de formar parte de la red. De forma general, a los equipos que forman parte de una red inalámbrica se les conoce como terminales [24].

Además de los adaptadores de red, las redes Wi-Fi pueden disponer también de unos equipos que reciben el nombre de puntos de acceso (AP o Access Points). Un punto de acceso es como una estación base utilizada para gestionar las comunicaciones entre distintas terminales. Los puntos de acceso funcionan de forma autónoma sin necesidad de ser conectados directamente a ninguna computadora [24].

Tanto a las terminales como a los puntos de acceso se les conoce por el nombre general de estación [24].

Las estaciones se comunican entre sí gracias a que utilizan la misma banda de frecuencias y a que internamente tienen instalado el mismo conjunto de protocolos. Aunque los protocolos que utiliza Wi-Fi están basados en las siete capas del modelo de referencia OSI, el estándar IEEE 802.11b sólo define las dos primeras capas (física y enlace); el resto de las capas son idénticas a las empleadas en las redes locales cableadas e Internet y se conoce con el nombre de conjunto de protocolos IP (Internet Protocol o Protocolo de Internet). En la Figura 10 se ilustran los protocolos de red local en el modelo OSI [24].

Modelo OSI			Protocolos
7	Aplicación	Común	HTTP, FTP, POP3
6	Presentación		DNS, LDAP, XML
5	Sesión		
4	Transporte		UDP, TCP
3	Red		IP, ICMP, RSVP
2	Enlace	IEEE 802	LLC, MAC
1	Físico		Coaxial, FO, radio

Figura 10. Protocolos de red local en el modelo OSI

Se determinó utilizar Wi-Fi porque es el único protocolo de conexión inalámbrica que soportan la mayoría de los routers domésticos, además de su largo alcance y su velocidad de transmisión de datos.

3.2.2. ZigBee

ZigBee es un protocolo estándar de comunicaciones para redes inalámbricas en malla de bajo consumo. No debe confundirse con Xbee que es una marca de módulos de radiofrecuencia que soporta una gran variedad de protocolos de comunicación, incluyendo ZigBee, 802.15.4 y Wi-Fi [25].

La capa de red bajo ZigBee que soporta todas sus características es conocida como IEEE 802.15.4. Este es un conjunto de estándares que definen la administración de energía, direccionamiento, corrección de errores, formatos de mensaje y otras especificaciones de redes punto a punto necesarias para la correcta comunicación de un módulo de radio a otro [25].

En la Figura 11 se muestran las capas del protocolo ZigBee.

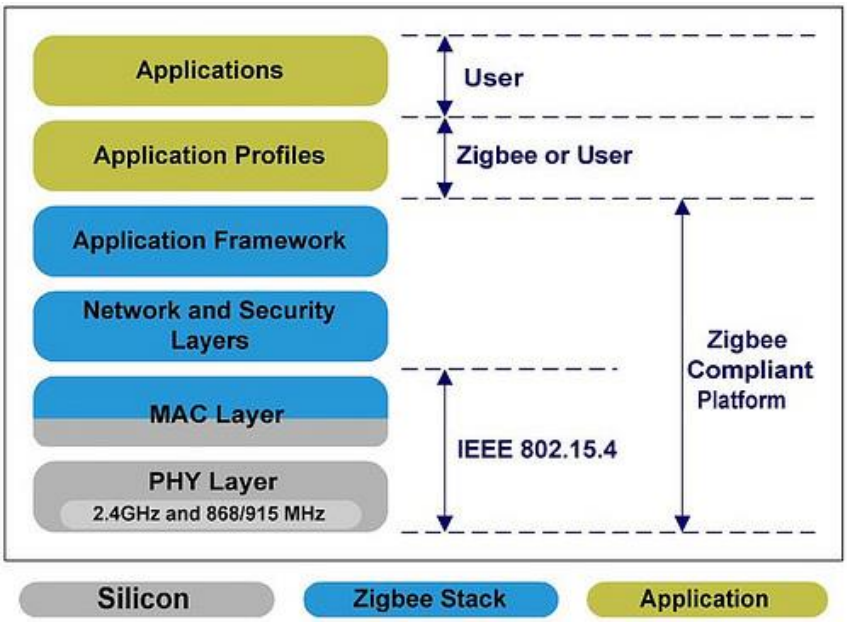


Figura 11. Capas del protocolo ZigBee [26]

Las características de las redes / dispositivos ZigBee serían las siguientes: [26]

- Velocidad de transmisión entre 25-250 kbps.
- Protocolo asíncrono, half duplex y estandarizado, permitiendo a productos de distintos fabricantes trabajar juntos.
- Se pueden formar redes que contengan desde dos dispositivos hasta cientos de ellos.
- Los dispositivos de estas redes pueden funcionar en un modo de bajo consumo, lo que supone años de duración de sus baterías.
- Opera en la frecuencia de 2.4 GHz (16 canales) y también en las frecuencias de 868 MHz y 915 MHz.
- Es un protocolo fiable, la red se organiza y se repara de forma automática y se rutean los paquetes de manera dinámica.
- Es un protocolo seguro ya que se puede implementar encriptación y autentificación.

Toda red ZigBee contiene los siguientes componentes: [25]

- Coordinador: las redes ZigBee siempre tendrán un dispositivo coordinador. Este módulo de radio es responsable de formar la red, repartir las direcciones y manejar otras funciones

que definen la red, la aseguran y la mantienen en buen estado. Ninguna red puede tener más de un coordinador.

- Router: es un nodo con todas las características de ZigBee. Se puede unir a redes existentes y puede enviar, recibir y encaminar información, es decir, actuar como mensajero en comunicaciones de otros dispositivos donde éstos se encuentran demasiado alejados como para transmitir la información por su cuenta. Una red puede tener más de un router que debe estar encendido todo el tiempo.
- Dispositivo final: es un nodo que posee las mismas características del router, exceptuando la función de mensajero. Puede apagarse intermitentemente para ahorrar energía, pero necesita un router o un coordinador como padre para que lo ayude a conectarse a una red y guarde sus mensajes cuando él esté dormido. Una red puede estar compuesta por un coordinador, muchos dispositivos finales y ningún router.

En la Figura 12 se muestran algunas de las topologías que se pueden configurar con ZigBee, como son par, estrella, grupo de árboles y malla.

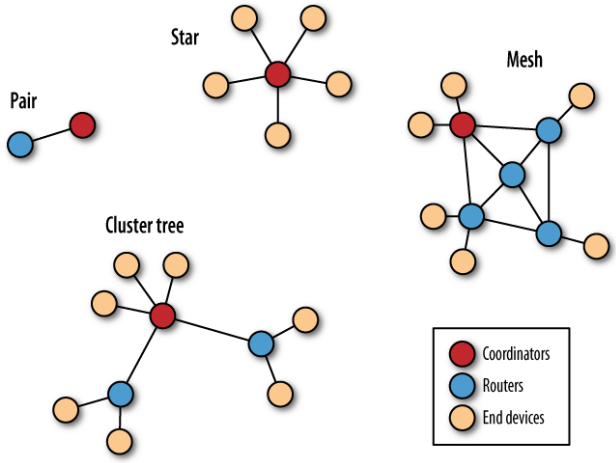


Figura 12. Topologías de ZigBee [25]

Se decidió trabajar con módulos Xbee serie 2 (que implementan el protocolo ZigBee) ya que son los únicos que soportan la implementación de un red en malla, tienen un menor consumo de energía respecto a Bluetooth y soportan una mayor distancia entre dispositivos.

3.3. Componentes de Hardware

3.3.1. Sensores

Para elegir los sensores de un sistema se debe de tomar en cuenta los siguientes aspectos: [27]

- Exactitud: se entiende como el error máximo que se permite dentro de una medición, generalmente se le relaciona con la linealidad y la histéresis que presenta el sensor.
- Repetibilidad: se entiende como la diferencia que presentan las mediciones, bajo las mismas condiciones del instrumento, el mismo operador en un periodo de tiempo corto; mientras menores sean las diferencias, mejor será el sensor.

- Tiempo de respuesta: velocidad con la que responde el sensor cuando se le somete a un cambio, en este punto influyen entre otros factores la temperatura, el flujo de aire y el tipo de filtro que se utiliza.
- Intervalos de operación: no es recomendable tener un sensor con un intervalo de medición muy grande para procesos de intervalos pequeños ya que las mediciones se volverán burdas y quizás no muestren los cambios como se requieren.
- Resistencia a contaminantes y ambientes extremos: si el proceso al que se someterá el instrumento de medición cuenta con ambientes poco usuales, se deberá tener esto en cuenta, eligiendo sensores especializados para dichos ambientes, colocando filtros especiales o protecciones a los sensores.
- Costo & Efectividad: tener el mejor sensor implica también su costo de operación comparado con su aplicación.

Para este proyecto se decidió utilizar sensores de temperatura, humedad, gas, monóxido de carbono y flama. En la Tabla 10 se describen los diferentes tipos de sensores que existen para cada una de dichas variables.

Variable	Tipo de Sensor	Características
Temperatura [28]	Termistor	• Está basado en que el comportamiento de la resistencia de los semiconductores es variable en función de la temperatura. • Existen los termistores tipo NTC y PTC. En los primeros, al aumentar la temperatura, disminuye la resistencia. En los segundos, al aumentar la temperatura, aumenta la resistencia. • Su principal problema es que no son lineales por lo que es necesario aplicar fórmulas complejas para determinar la temperatura.
	RTD	• Resistance Temperature Detector • Se basa en la variación de la resistencia de un conductor con la temperatura. • Los metales empleados normalmente son platino, cobre, niquel y molibdeno. • Tienen mejor linealidad, rapidez y mayor margen de temperatura.
	Termopar	• También conocido como termocupla, recibe este nombre por estar formado por dos metales que genera una tensión que está en función de la temperatura aplicada al sensor. • Los termopares tienen un amplio rango de medida, son económicos y están muy extendidos en la industria.

		• El principal inconveniente estriba en su precisión, que es pequeña en comparación con los otros tipos de sensores.
Humedad [29]	Capacitivo	• En principio en el cual se basa este tipo de sensores es en el cambio que sufre la capacidad de un condensador al variar la constante dieléctrica del mismo. • Se utiliza como una de las placas un alambre conductor, mientras la otra es una malla fina de oro que permite el paso del gas, pero retiene impurezas, como dieléctrico se utiliza un material higroscópico poroso (cerámico) que rodea el alambre, el cual absorbe el agua de la muestra, aumentando la constante dieléctrica del condensador. • Son los más difundidos pues son de fácil producción, bajo costo y alta fidelidad. • Son robustos y tienen excelente precisión.
	Infrarrojo	• Se dispone de 2 fuentes infrarrojas idénticas, la primera se toma como referencia y es medida por una foto-resistencia, la segunda atraviesa la muestra con vapor de agua, el cual absorbe parte de la radiación e incide en el otro detector, ambos valores resistivos son transformados a voltaje por puentes de Wheastone, para finalmente ser comparados con un amplificador diferencial. La diferencia entre ambos va a ser proporcional a la cantidad de humedad presente en la muestra. • Son muy sensibles y logran gran precisión.
	Piezoeléctrico	• Se trata de un cristal cubierto con un material higroscópico, éste aumenta la cantidad de agua sobre el cristal en forma proporcional a la humedad absoluta presente. A mayor masa, menor es la frecuencia de oscilación. • Son sensores robustos y muy sensibles, son de transiciones cortas y entregan una medida en forma de frecuencia, la cual puede ser utilizada para control con PLL o convertida a voltaje.
Gas [30]	Electroquímico	• Estos sensores están formados por dos electrodos sumergidos en un medio electrolítico común. El electrolito es aislado de las influencias externas mediante una barrera, que puede ser una membrana permeable al gas, un medio de difusión o un capilar. • Se pueden utilizar para detectar la mayoría de los gases tóxicos comunes.

		• Son compactos, requieren de poca energía, muestran una gran linealidad y repetibilidad y generalmente tienen una larga vida útil.
	Por semiconductor	• El sensor es fabricado con materiales semiconductores. • Opera por la propiedad de adsorción de gas en la superficie de un óxido calentado depositada en una base de silicio. La absorción de la muestra de gas en la superficie de óxido, seguida por la oxidación catalítica, termina en un cambio de la resistencia eléctrica del material oxidado que puede relacionarse con la concentración del gas.
	De conductividad térmica	• Estos sensores consisten en la disposición de al menos dos termistores formando parte de un puente de Wheastone. • En la célula de referencia se encierra una cantidad determinada de un gas estándar, en la célula de medida penetra el gas a detectar; su conductividad térmica, diferente del gas de referencia, hace que la temperatura del filamento se altere y desequilibre el circuito.
	Catalítico	• Consiste en un pequeño elemento denominado "perla" que está formado por un filamento de platino calentado eléctricamente. Este filamento está recubierto primeramente con una base cerámica y posteriormente por una dispersión catalítica de paladio o rodio. • El cambio de resistencia está directamente relacionado con la concentración de gas presente.
	Infrarrojo	• Se basa en el hecho de que muchos gases combustibles tienen bandas de absorción en el espectro infrarrojo. • Un detector compara las fuerzas de las señales de los haces de referencia y muestra y, por medio de una resta, se proporciona una medida de la concentración de gas.
Humos (CO) [31]	Fotoeléctrico	• Existen de dos tipos: de haz de rayos proyectado y de haz de rayos reflejados. • El humo visible que penetra el aparato afecta el haz de rayos luminosos generado por una fuente de luz, de forma que varía la luz recibida en una célula fotoeléctrica. • Tienen respuesta rápida ante fuegos con humos, particularmente con humos claros producidos por la combustión de madera, algodón, papel y el recalentamiento de cables aislados con PVC.

		• El inconveniente es que si el humo es negro no lo detecta ya que no hay dispersión de luz.
	Iónico	• Existen de dos tipos: de partículas alfa y de partículas beta. • Se basan en la disminución que experimenta el flujo de corriente eléctrica formada por moléculas de oxígeno y nitrógeno ionizadas por una fuente radiactiva entre dos electrodos, al penetrar los productos de combustión de un incendio. • Son detectores aptos para toda gama de humos, son estables ante variaciones de presión, temperatura y corrientes de aire. • El principal inconveniente es que da falsas alarmas en ambientes con aerosoles, polvo, aire en movimiento, humedad elevada, humo de cigarros, etc.
	De puente de resistencia	• Estos sensores se activan ante la presencia de partículas de humo sobre una rejilla con puente eléctrico. Esas partículas al caer sobre la rejilla aumentan su conductividad. • Estos detectores reaccionan con cualquier gas o humo. • Son empleados generalmente como detectores de monóxido de carbono.
	Por semiconductor	• El cristal semiconductor de tipo negativo lleva embebidas dos resistencias calefactoras que mantienen el semiconductor para que aumente el número de electrones libres. • Cuando el sensor está expuesto a una atmósfera que contenga un gas oxidable, sus moléculas reaccionan con el oxígeno atrapado, originando una liberación de electrones en la superficie conductora, entonces disminuye la resistencia de esa superficie y se dispara una alarma. • No discrimina bien entre gases o vapores de ciertas sustancias y humos.
Flama [32]	Termocupla	• Estos sensores emiten un voltaje muy pequeño, cuando el tubo que contiene las tiras bi-metálicas de las termocuplas toma contacto con una fuente de calor, se genera un potencial eléctrico. • Estos sensores son económicos y extremadamente precisos para detectar la llama.
	Óptico	• Este tipo de sensores puede “ver” la llama y generar un voltaje mucho más grande.

		• Los lectores ópticos usan fotoceldas, que cuando son expuestas a la luz ultravioleta generan voltaje a partir de la fuente de luz. • Estos sensores son muy sensibles a cualquier polvo y la ventana del sensor debe tener una exposición total a la llama.

Tabla 10. Tipos de sensores

Para este proyecto se escogieron los siguientes sensores:

- AM2302: es un sensor capacitivo de humedad con un termistor para medir temperatura. Tiene la capacidad de medir de 0 a 100% de humedad y de -40 a 125°C de temperatura. La única desventaja de este sensor es que solamente puede obtener lecturas cada dos segundos [33].
- MQ2: es un sensor semiconductor para gas combustible. Tiene una alta sensibilidad al LPG, propano e hidrógeno, pero también puede usarse para medir metano y otros gases combustibles. Tiene la capacidad de medir concentraciones de 300 a 10 000 ppm [34].
- MQ7: es un sensor semiconductor para monóxido de carbono. Tiene la capacidad de medir concentraciones de 10 a 10 000 ppm [35].
- Grove – Flame Sensor: puede ser usado para detectar fuego y otras fuentes de luz de una longitud de onda en el rango de 760 nm a 1100 nm. Está basado en el sensor YG1006, que es un fototransistor NPN de silicón, de alta velocidad y sensibilidad [36].

3.3.2. Actuadores

En un sistema de detección de incendios además de los sensores que realizan las mediciones ambientales, es importante contar con un circuito de control para activar un sistema de protección contra incendios o un sistema de control de humos. Está fuera de alcance de este proyecto la instalación de sistemas de este tipo, sin embargo se utilizaron relevadores para activar dispositivos que emulan el comportamiento de aspersores de agua (sistema de protección contra incendios) y extractores de aire (sistema de control de humos) para efecto de pruebas. Además, se decidió agregar un relevador al suministro de energía eléctrica con el fin de suspenderlo cuando la casa se encuentre en estado de alerta para evitar riesgos de electrocución. Finalmente, se agregaron alarmas auditivas (bocinas) y visuales (LED) que alertan del riesgo a los habitantes de la casa.

3.4. Análisis de Factibilidad

3.4.1. Factibilidad en Tiempos

Para asegurar que se cumplieran los tiempos de este proyecto se creó un calendario con las actividades correspondientes a la investigación, diseño y desarrollo del sistema, el cual se muestra en la Figura 13.

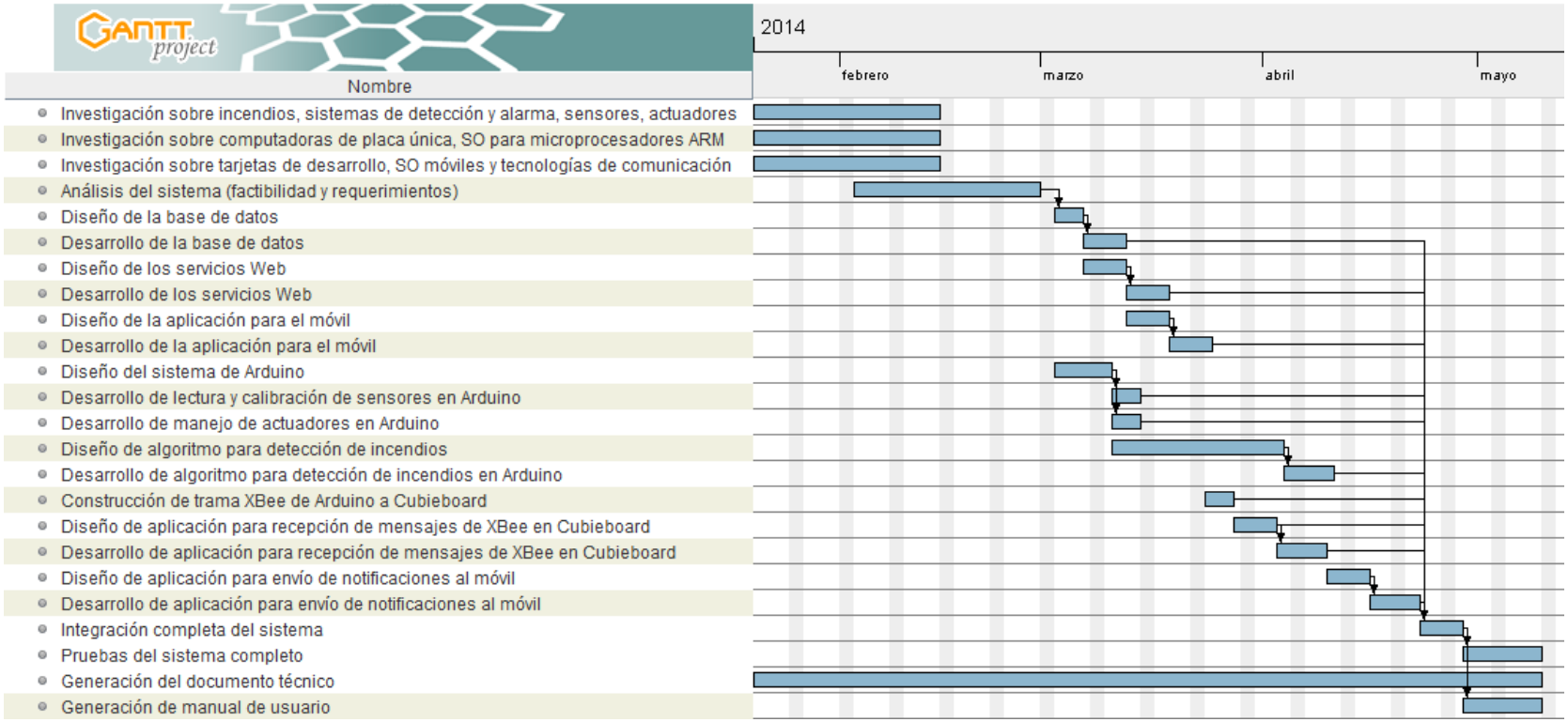


Figura 13. Calendario de actividades

Para cumplir con el tiempo designado se decidió que el sistema fuera desarrollado por un equipo de dos personas con conocimientos de Ingeniería en Sistemas Computacionales, tanto en el área de programación como en electrónica.

3.4.2. FACTIBILIDAD TÉCNICA

Los conocimientos mínimos que se requieren para el desarrollo del sistema son:

- Programación Avanzada
- Electrónica Analógica y Digital
- Métodos Cuantitativos para la Toma de Decisiones
- Matemáticas Avanzadas
- Ingeniería de Software
- Sistemas Distribuidos

Por otro lado con respecto a las herramientas se tomó la decisión de trabajar con las siguientes:

- Software
  - IDE Eclipse y lenguaje de programación Java para servicios Web y aplicación en Android
  - IDE Arduino y lenguaje de programación C para programación de tarjeta Arduino
  - XCTU para configuración de Xbee
  - Sistema operativo Cubian para Cubieboard
- Hardware
  - Arduino UNO
  - Computadora de placa única Cubieboard

3.4.3. FACTIBILIDAD ECONÓMICA

En la Tabla 11 se detallan los componentes que se adquirieron para el desarrollo del sistema.

Concepto	P.U.	Piezas	Subtotal
Cubieboard	$709.12	1	$709.12
Antena Wi-Fi	$200.00	1	$200.00
Módulo Xbee Series 2	$470.00	4	$1,880.00
Arduino UNO	$380.00	3	$1,140.00
Sensor de temperatura y humedad (AM2302)	$161.73	3	$485.19
Sensor de gas combustible (MQ2)	$80.00	3	$240.00
Sensor de CO (MQ7)	$80.00	3	$240.00
Sensor de flama (Grove Flame Sensor)	$89.00	3	$267.00
Relevador (RAS 510)	$17.00	8	$136.00
Buzzer Grove	$24.95	3	$74.85
Ventilador	$25.00	3	$75.00
		Total	**$5,447.16**

Tabla 11. Precio de los componentes comprados

3.5. ANÁLISIS DE RIESGOS

La gestión de riesgos comprende las siguientes etapas: [37]

1. Identificación de riesgos: identificar los posibles riesgos para el proyecto, el producto y los negocios.
2. Análisis de riesgos: valorar las probabilidades y consecuencias de estos riesgos.
3. Planeación de riesgos: crear planes para abordar los riesgos, ya sea para evitarlos o minimizar sus efectos en el proyecto.
4. Supervisión de riesgos: valorar los riesgos de forma constante y revisar los planes para la mitigación de éstos tan pronto como la información de los riesgos esté disponible.

En la Figura 14 se muestra el proceso de gestión de riesgos [37].

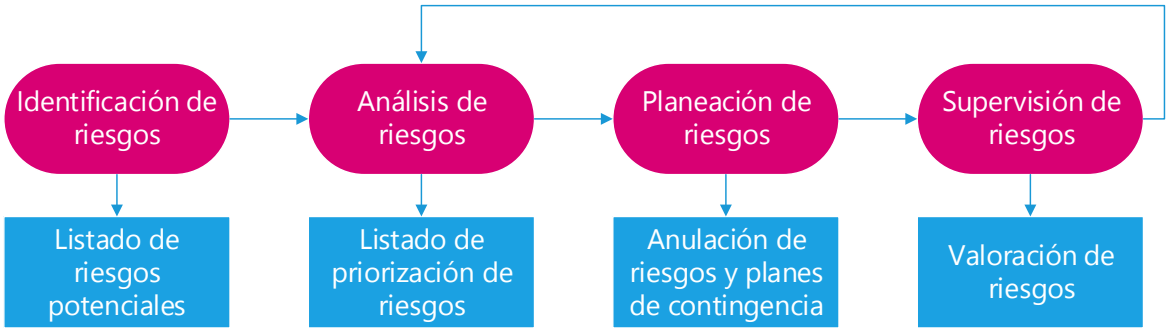


Figura 14. Proceso de gestión de riesgos

En la Tabla 12 se describen los riesgos que pueden presentarse durante el desarrollo del proyecto.

Categoría	Riesgos
Tecnología	El software necesario para el desarrollo del sistema no tiene soporte para arquitectura ARM.
	La base de datos que se utiliza en el sistema no puede procesar una gran cantidad de transacciones por segundo como se esperaba.
	El coordinador de Xbee no puede recibir una gran cantidad de peticiones por segundo como se esperaba.
	La cantidad de memoria de programa sobrepasa la capacidad de la tarjeta Arduino.
	La conexión a Internet del sistema es inestable.
	El tiempo de entrega de los componentes de hardware es más largo de lo que se esperaba.
Personal	Uno de los integrantes abandona el equipo.
	La curva de aprendizaje de los integrantes del equipo se alarga en el tiempo.
Estimación	El tiempo requerido para desarrollar el sistema está subestimado.
	El tiempo requerido para desarrollar la documentación está subestimado.
	El tamaño del software está subestimado.

Tabla 12. Identificación de riesgos

En la Tabla 13 se describen los riesgos del proyecto, la probabilidad de que ocurran y los efectos que podrían producir de llegarse a presentar.

Descripción del Riesgo	Probabilidad	Efecto
El software necesario para el desarrollo del sistema no tiene soporte para arquitectura ARM.	Bajo	Serio
Uno de los integrantes abandona el equipo.	Muy bajo	Serio
El tiempo requerido para desarrollar el sistema está subestimado.	Moderado	Serio
El tiempo requerido para desarrollar la documentación está subestimado.	Moderado	Serio
La base de datos que se utiliza en el sistema no puede procesar una gran cantidad de transacciones por segundo como se esperaba.	Bajo	Tolerable
El coordinador de Xbee no puede recibir una gran cantidad de peticiones por segundo como se esperaba.	Moderado	Tolerable
La cantidad de memoria de programa sobrepasa la capacidad de la tarjeta Arduino.	Bajo	Tolerable
La conexión a Internet del sistema es inestable.	Bajo	Tolerable
El tiempo de entrega de los componentes de hardware es más largo de lo que se esperaba.	Alto	Tolerable
La curva de aprendizaje de los integrantes del equipo se alarga en el tiempo.	Alto	Tolerable
El tamaño del software está subestimado.	Moderado	Tolerable

Tabla 13. Análisis de riesgos

En la Tabla 14 se describen las estrategias de gestión de los riesgos más importantes para este sistema.

Riesgo	Estrategia de gestión
Tiempo requerido para desarrollo subestimado	Dedicar tiempo de otras actividades menos importantes o que se hayan terminado antes para compensar retrasos. En un caso extremo, pedir una prórroga para la entrega del sistema funcional.
Tiempo requerido para documentación subestimado	Dedicar tiempo de otras actividades menos importantes o que se hayan terminado antes para compensar retrasos. En un caso extremo, pedir una prórroga para la entrega del documento.
Tiempo de entrega de componentes de hardware retardado	Buscar otros proveedores aunque el precio de los componentes se eleve.
La curva de aprendizaje se alarga en el tiempo	Dedicar tiempo de otras actividades menos importantes o que se hayan terminado antes para compensar retrasos.

Tabla 14. Estrategia de gestión de riesgos

3.6. Análisis de Requerimientos

3.6.1. Requerimientos Funcionales

Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares [37].

Los requerimientos funcionales del presente sistema son:

- El sistema deberá permitir al usuario registrarse antes de ingresar a la aplicación.
- El usuario deberá tener la posibilidad de recuperar su contraseña, cambiarla y cambiar su dirección de correo electrónico.
- El sistema deberá validar un nombre de usuario y una contraseña para poder ingresar a la aplicación.
- El sistema deberá obtener los valores de los sensores conectados en cada uno de los módulos.
- El sistema deberá controlar los actuadores conectados basándose en el resultado del algoritmo de detección de incendios.
- El sistema deberá ser capaz de mostrar si alguno de los módulos se encuentra desconectado o si los sensores y actuadores tienen alguna falla.
- El sistema deberá guardar la información emitida por los módulos.
- El sistema deberá enviar notificaciones al usuario cuando el sistema se encuentre en estado de alerta.

3.6.2. Requerimientos No Funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Pueden incluir características de fiabilidad, tiempo de respuesta, capacidad de almacenamiento, etc.

Los requerimientos no funcionales del presente sistema son:

- La interfaz de usuario se implementará como una aplicación para un dispositivo móvil con sistema operativo Android.
- El tiempo de respuesta del sistema no deberá exceder 5 segundos.
- La capacidad de almacenamiento del módulo central estará determinada por la capacidad de la memoria externa.
- Los módulos de sensores estarán alimentados por una batería de 9 V no recargable.

3.7. Reglas del Negocio

Las reglas de negocio describen políticas, normas, operaciones, definiciones y restricciones presentes en la definición de un sistema.

Las reglas de negocio del presente sistema son:

- Información requerida para agregar un usuario: la información que se debe almacenar de los usuarios es: nombre de usuario, contraseña y dirección de correo electrónico. Para poder completar el usuario también debe proporcionar un número de serie único que se encuentra en el módulo central, aunque éste no se va a guardar.
    - El nombre de usuario debe estar formado únicamente por números y letras (no se permiten caracteres especiales), debe comenzar con una letra y debe tener una longitud máxima de 15 caracteres.
    - La contraseña puede contener cualquier carácter, debe contener al menos 1 número, su longitud mínima debe ser de 4 caracteres y su longitud máxima de 15 caracteres.
    - La dirección de correo válida debe tener el formato:

$$nombre\ de\ usuario + @ + \ servidor + dominio$$

- Definición de datos de usuario: los datos de usuario son aquellos que permiten autenticar a un usuario en el sistema y son: nombre de usuario y contraseña.
- Información requerida para modificar un usuario: la información que puede modificarse de un usuario es: contraseña y dirección de correo electrónico.
- Definición de datos de módulos de sensores: los datos de módulos de sensores son aquellos que reflejan el estado del módulo y los valores de los diversos sensores que lo integran, estos datos son: nombre de módulo, lista con los nombres de los sensores y los últimos valores de éstos.

3.8. Modelado del Sistema

En la Figura 15 se muestra el modelo general del sistema.

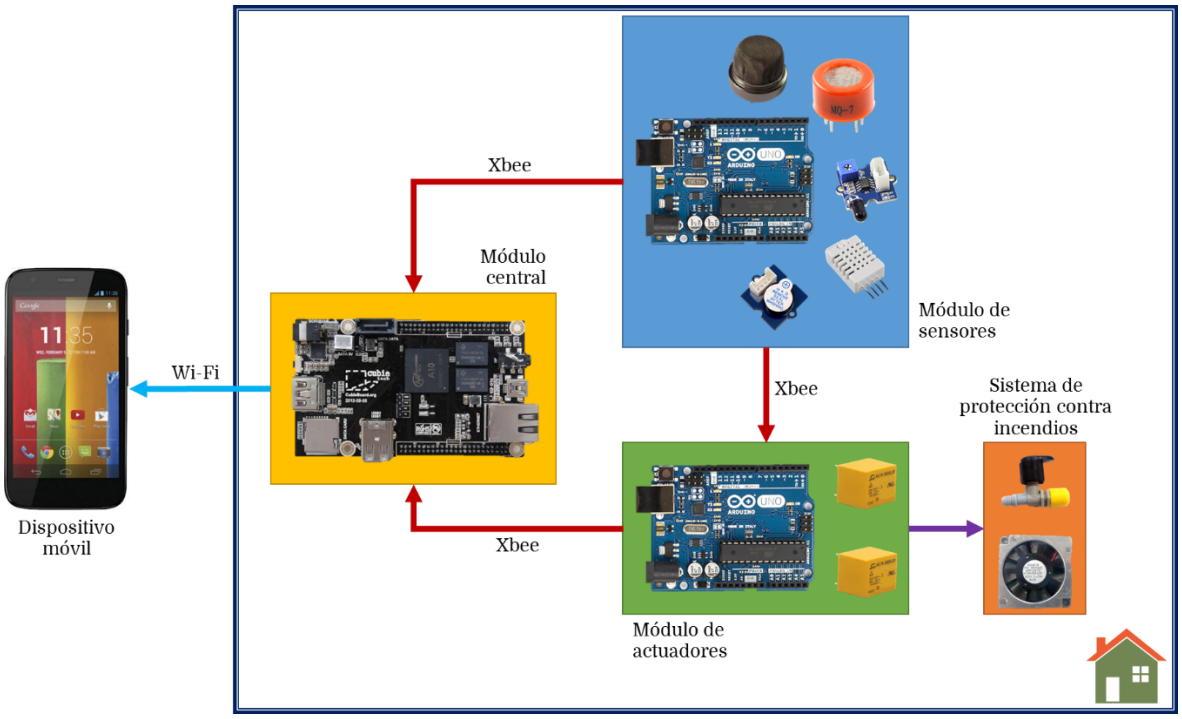


Figura 15. Modelo general del sistema

CAPÍTULO IV: INCREMENTOS DEL SISTEMA

4.1. INCREMENTO 1

En el primer incremento del sistema se desarrollaron la base de datos, los servicios Web que se encuentran en el módulo central (Cubieboard) y la aplicación para Android con las funcionalidades básicas para visualizar datos.

4.1.1. BASE DE DATOS

La base de datos almacena las direcciones de los módulos que se encuentran presentes en el sistema, los sensores y actuadores que se relacionan con cada uno de ellos, así como los historiales que muestran los cambios de las variables o del estado de alerta a través del tiempo. Además, se almacenan los datos de los usuarios que tienen acceso al sistema y las notificaciones que se enviarán al dispositivo móvil cuando se presente una emergencia.

En el Diagrama 1 se muestra el modelo entidad relación de la base de datos y en el Diagrama 2 se muestra el modelo relacional de la primera versión de la base de datos. En el Diagrama 3 se muestra el modelo relacional de la versión final de la base de datos, después de llevar la primera versión de la base de datos a su Tercera Forma Normal.

La normalización es un concepto que hace referencia a las relaciones de una base de datos. Básicamente, indica que las tablas de las bases de datos eliminarán las incoherencias y redundancias y minimizarán la ineficacia [38]. A continuación se describen las tres primeras formas normales:

- Primera Forma Normal: un esquema de relación está en primera forma normal (1FN) sí, y sólo sí, los dominios de todos los atributos de la relación son atómicos, es decir, prohíbe los atributos multivalorados, compuestos y sus combinaciones.
- Segunda Forma Normal: un esquema de relación está en segunda forma normal (2FN) sí, y solo sí, está en 1FN y, además cada atributo del esquema de relación que no está en la clave primaria depende funcionalmente de la clave primaria completa y no solo de una parte de esta.
- Tercera Forma Normal: un esquema de relación está en tercera forma normal (3FN) sí, y sólo sí, está en 2FN y, además cada atributo del esquema de relación que no está en la clave primaria solo depende funcionalmente de la clave primaria, y no de ningún otro atributo.

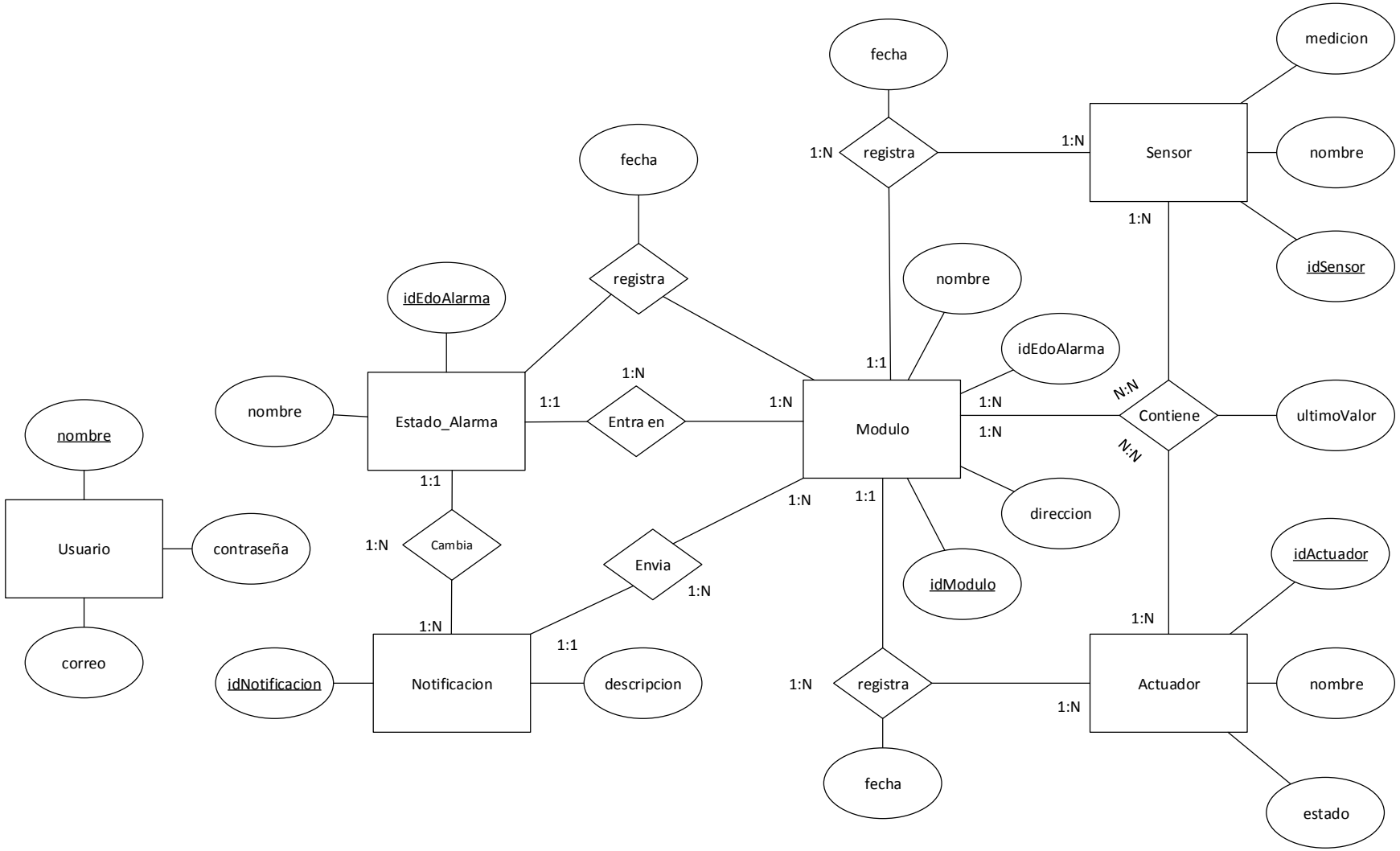


Diagrama 1. Modelo entidad relación de la base de datos

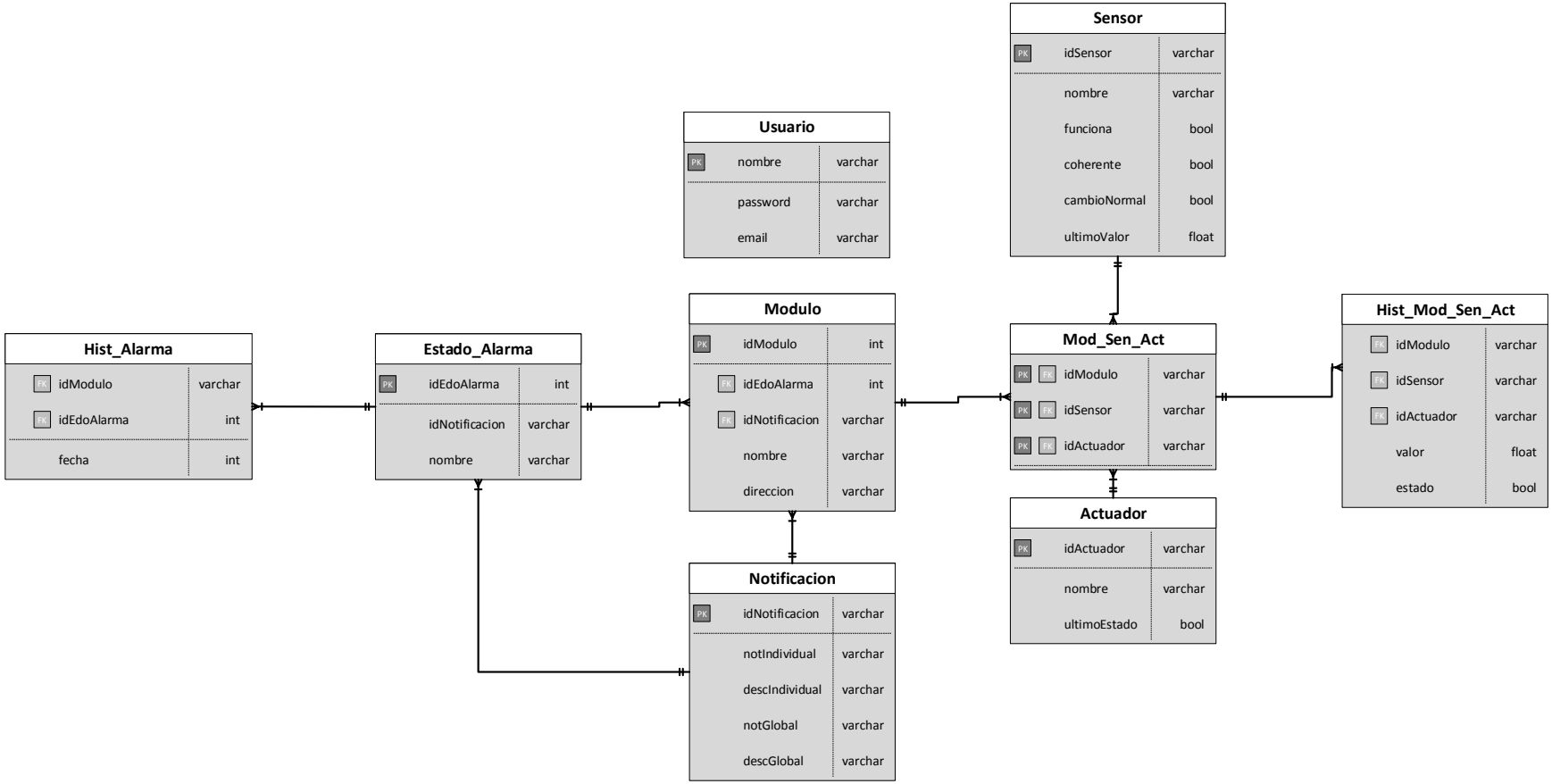


Diagrama 2. Modelo relacional de la base de datos (primera versión)

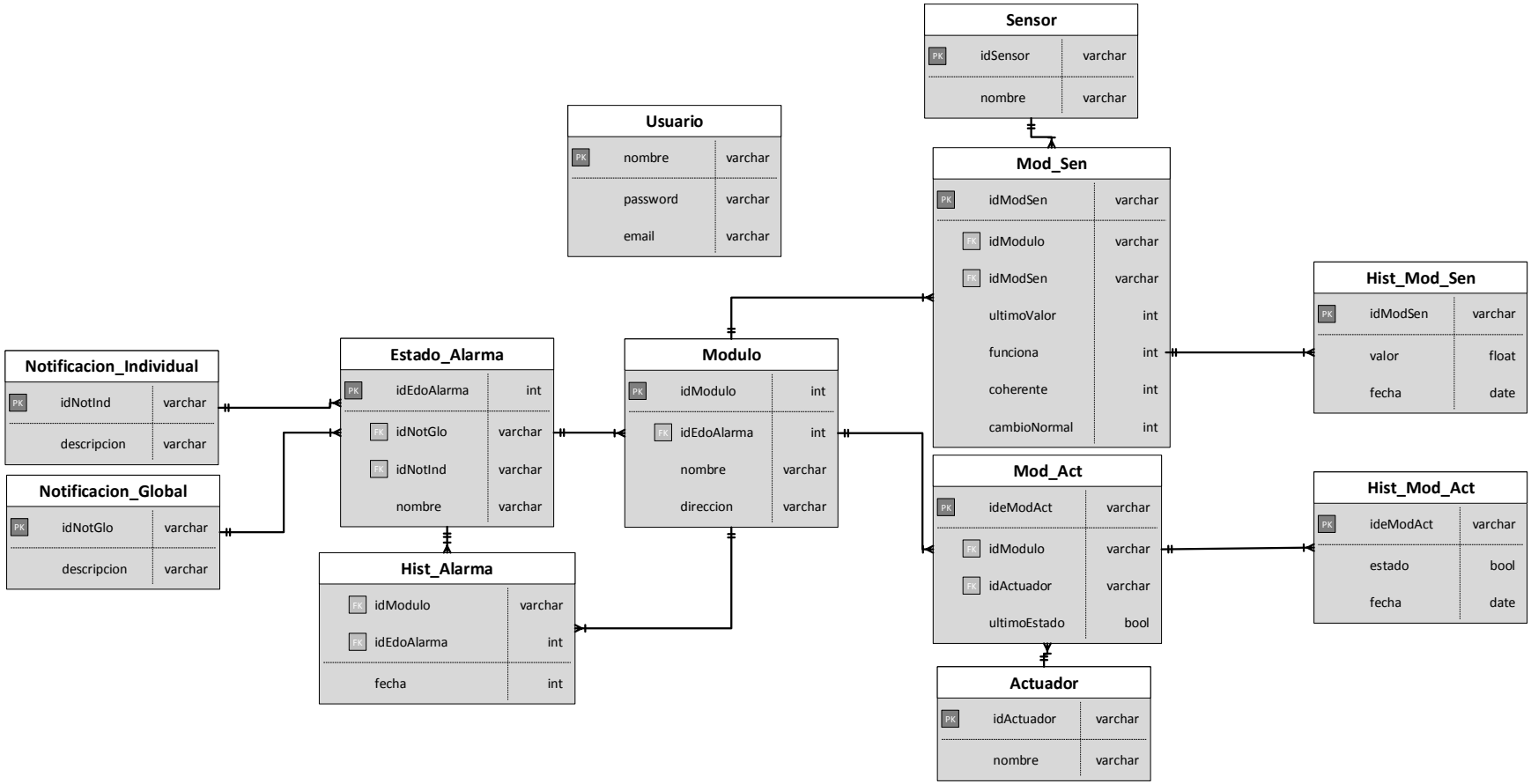


Diagrama 3. Modelo relacional de la base de datos normalizada (versión final)

4.1.2. Aplicación Móvil

El diseño de las pantallas de la aplicación para el dispositivo móvil se muestra en la Figura 16.

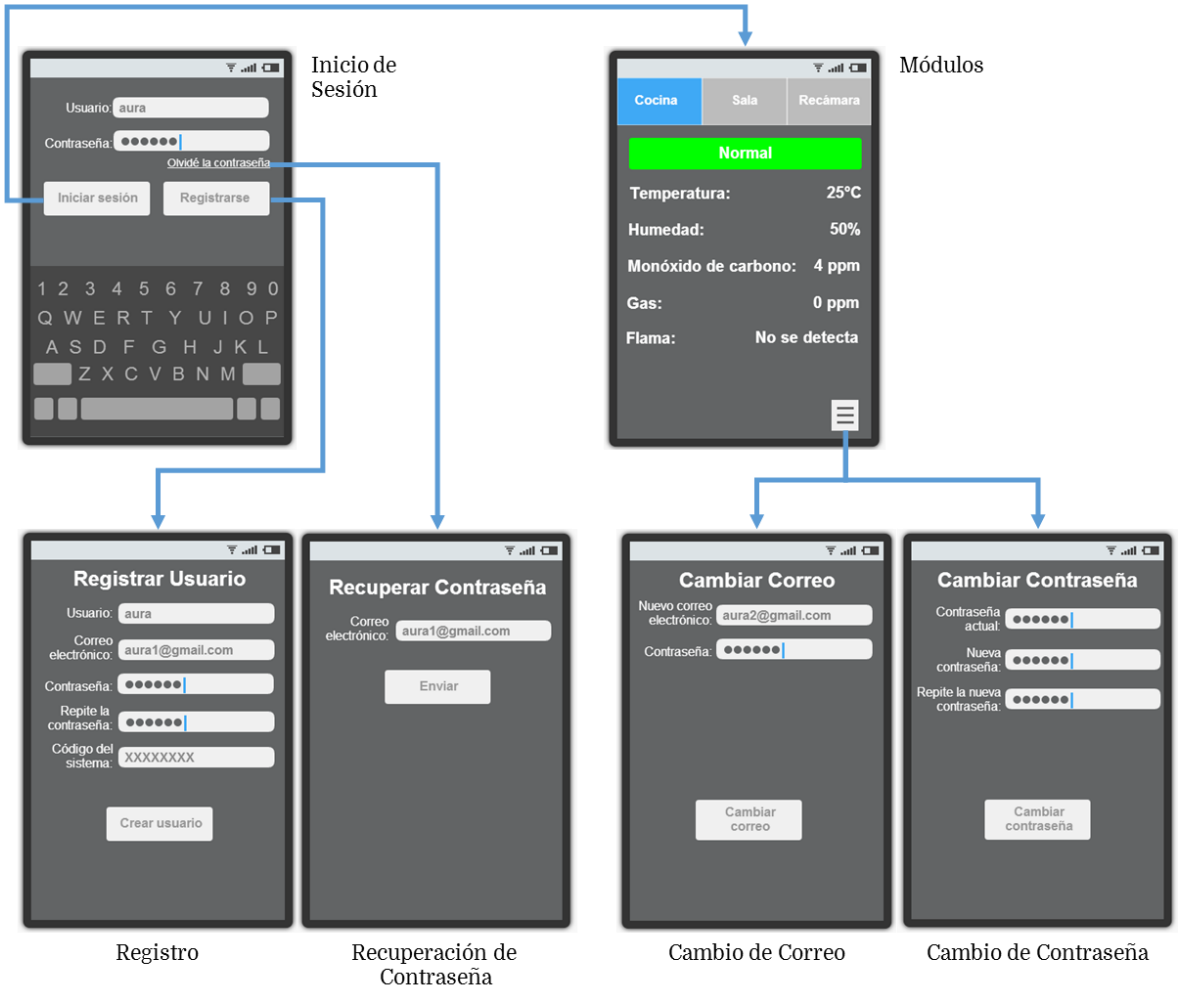


Figura 16. Pantallas de la aplicación

4.1.2.1. Casos de Uso

Los casos de uso que se describen en el Diagrama 4 muestran las principales funciones de la aplicación móvil.

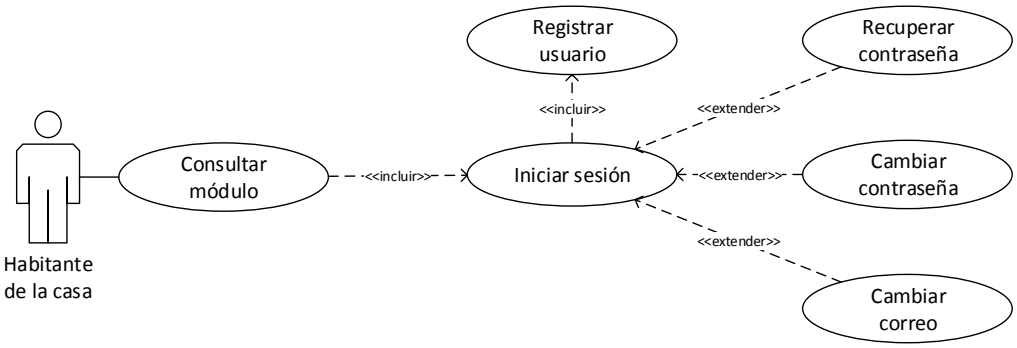


Diagrama 4. Casos de uso

4.1.2.1.1. Caso de Uso: Registrar Usuario [CU001]

Actor: Habitante (🧍)

Meta en contexto: Registrar un usuario para que pueda acceder al sistema.

Condiciones previas:

- El sistema se encuentra instalado y configurado.
- La aplicación *Domoduino* se encuentra instalada en el dispositivo móvil.
- El dispositivo móvil y el módulo central cuentan con conexión a Internet.

Disparador: Un habitante de la casa que no se ha registrado previamente decide consultar el estado del sistema.

Prioridad: Alta, se requiere para ingresar al sistema.

Disponible en: El primer incremento.

Frecuencia de uso: Poco frecuente.

Trayectoria Principal:

1. El 🧍 inicia la aplicación *Domoduino* en su dispositivo móvil (Trayectoria Alterna **CU001_A**).
2. La aplicación despliega la pantalla “Inicio de Sesión”.
3. El 🧍 presiona el botón “Registrarse”.
4. La aplicación despliega la pantalla “Registro”.
5. El 🧍 introduce su nombre de usuario válido.
6. El 🧍 introduce dos veces su contraseña válida.
7. El 🧍 introduce su dirección de correo electrónico.
8. El 🧍 introduce el código del sistema (ubicado en la parte posterior del módulo central).
9. El 🧍 presiona el botón “Crear Usuario”.
10. La aplicación valida el formato de los datos introducidos (véase Reglas del Negocio).
11. La aplicación envía los datos para ser validados y registrados en el módulo central (Trayectoria Alterna **CU001_B**).
12. La aplicación despliega el mensaje “Usuario Creado” (Trayectoria Alterna **CU001_C**).
13. La aplicación regresa a la pantalla “Inicio de Sesión”.

Trayectorias Alternas:

CU001_A: El dispositivo móvil pierde la conexión a Internet.

1. La aplicación despliega el mensaje “Revise su conexión a Internet”.
2. La aplicación se cierra.

CU001_B: El módulo central no responde.

1. La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde".
2. La aplicación se cierra.

CU001_C: El módulo central regresa un código de error

1. La aplicación verifica el código de error.
   a. La aplicación despliega el mensaje "El usuario o correo ya ha sido registrado anteriormente".
   b. La aplicación despliega el mensaje "Código de módulo inválido".
2. La aplicación borra los datos introducidos.

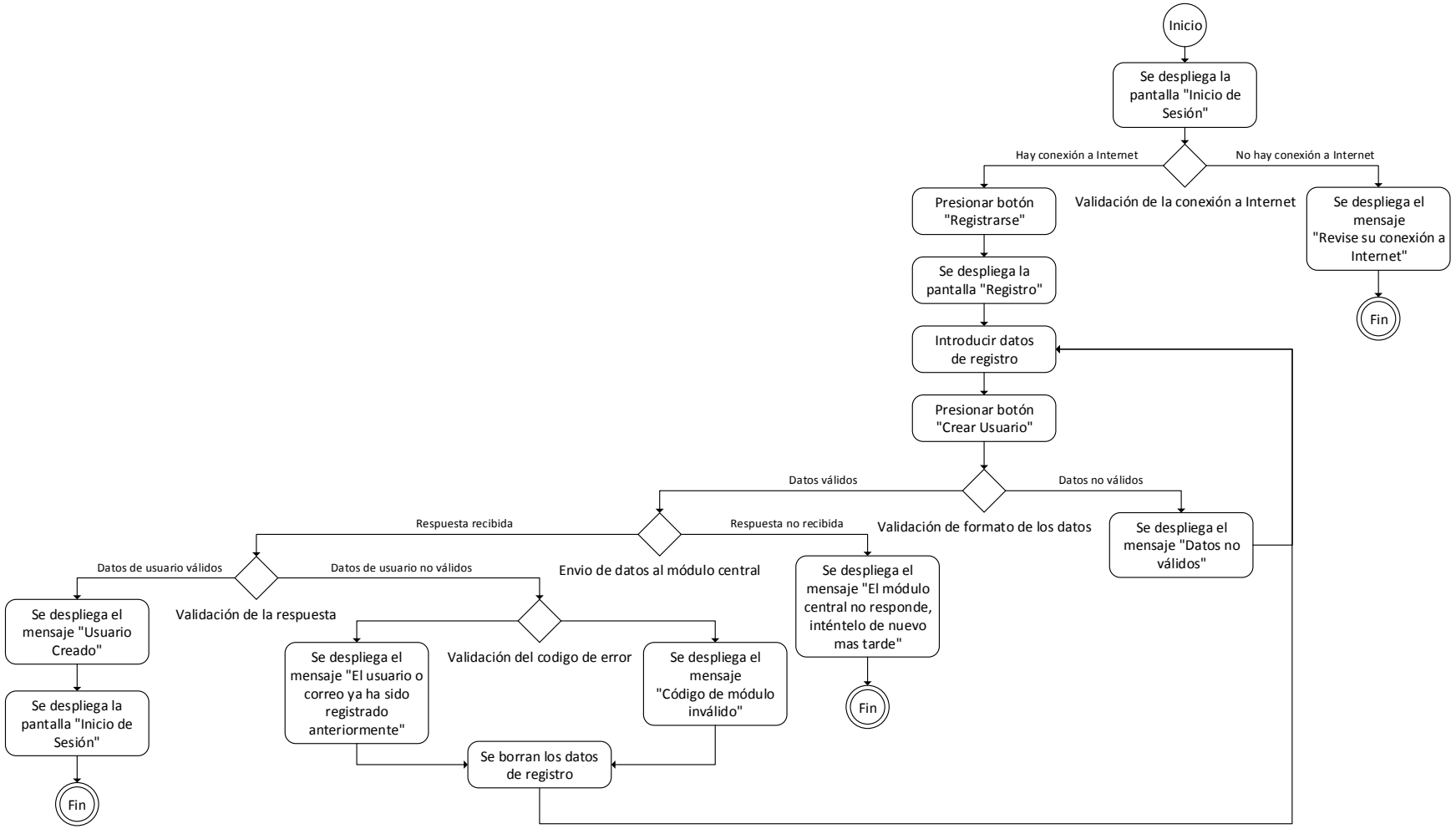


Diagrama 5. Diagrama de actividad para el caso de uso "Registrar Usuario"

4.1.2.1.2. CASO DE USO: INICIAR SESIÓN [CU002]

Actor: Habitante (🧍)

Meta en contexto: Autentificar a un usuario para que pueda acceder al sistema.

Condiciones previas:

- El sistema se encuentra instalado y configurado.
- La aplicación *Domoduino* se encuentra instalada en el dispositivo móvil.
- El dispositivo móvil y el módulo central cuentan con conexión a Internet.
- El 🧍 se encuentra registrado en el sistema (véase Caso de Uso "**CU001**").

Disparador: Un habitante de la casa registrado previamente decide consultar el estado del sistema.

Prioridad: Alta, se requiere para ingresar al sistema.

Disponible en: El primer incremento.

Frecuencia de uso: Frecuente.

Trayectoria Principal:

1. El 🧍 inicia la aplicación *Domoduino* en su dispositivo móvil (Trayectoria Alterna **CU001_A**).
2. La aplicación despliega la pantalla "Inicio de Sesión".
3. El 🧍 introduce su nombre de usuario y su contraseña.
4. La aplicación valida el formato de los datos introducidos (véase Reglas del Negocio).
5. El 🧍 presiona el botón "Iniciar sesión".
6. La aplicación envía los datos para ser validados en el módulo central (Trayectoria Alterna **CU001_B**).
7. La aplicación despliega el mensaje "Bienvenido"
8. El flujo del sistema continúa en el caso de uso "**CU003**".

Trayectorias Alternas:

CU002_A: El dispositivo móvil pierde la conexión a Internet.

1. La aplicación despliega el mensaje "Revise su conexión a Internet".
2. La aplicación se cierra.

CU002_B: El módulo central no responde.

1. La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde".
2. La aplicación se cierra.

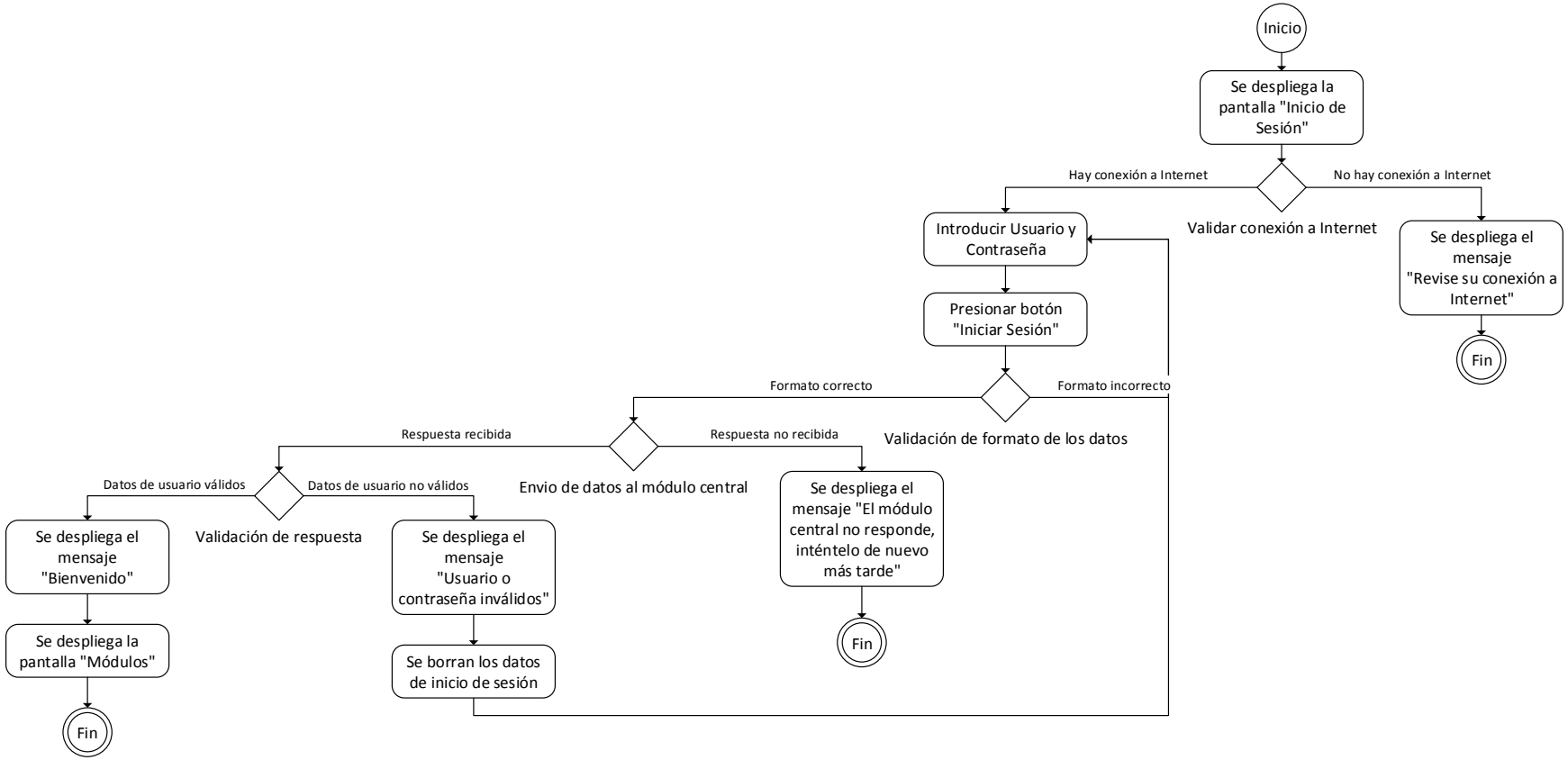


Diagrama 6. Diagrama de actividad para el caso de uso "Iniciar Sesión"

4.1.2.1.3. CONSULTAR MÓDULO [CU003]

Actor: Habitante ( )

Meta en contexto: Obtener información del estado de los sensores de cada módulo.

Condiciones previas:

- El usuario se encuentra autentificado (véase Caso de Uso "**CU002**").

Disparador: Un habitante de la casa quiere verificar el estado de los módulos y sus sensores instalados.

Prioridad: Alta, se requiere para visualizar el estado de los módulos y sus sensores.

Disponible en: El primer incremento.

Frecuencia de uso: Frecuente.

Trayectoria Principal:

1. El sistema se conecta al módulo central para obtener la información de los módulos con sensores (Trayectoria Alterna **CU003_A**).
2. La aplicación despliega la pantalla "Módulos" mostrando los módulos en forma de lista horizontal, su estado y los valores de sus respectivos sensores.
3. El verifica el estado de los módulos y los valores de sus sensores deslizando la pantalla hacia la izquierda.

Trayectorias Alternas:

CU003_A: El módulo central no responde.

1. La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde".
2. La aplicación se cierra.

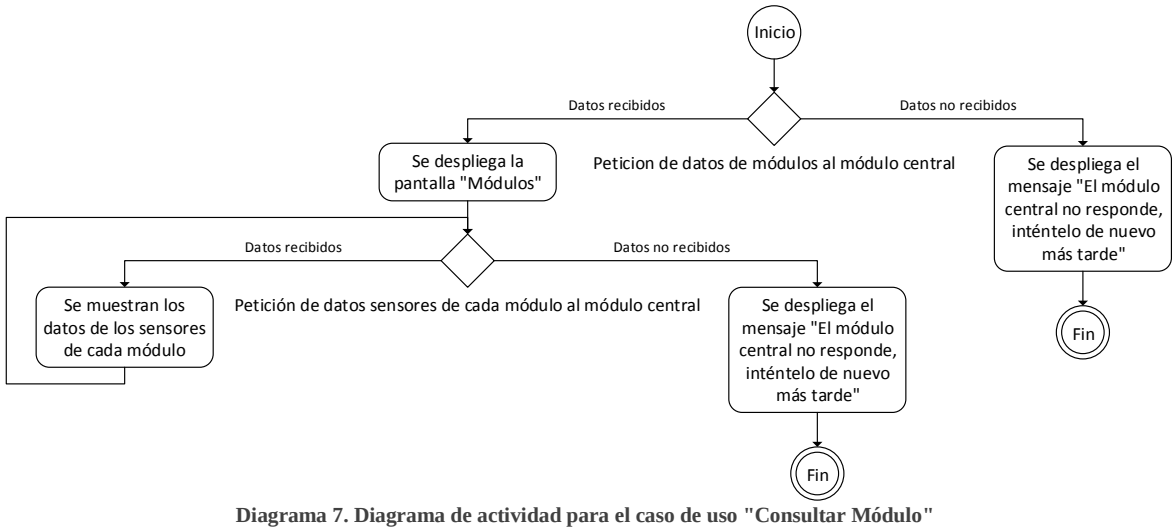


Diagrama 7. Diagrama de actividad para el caso de uso "Consultar Módulo"

4.1.2.1.4. CAMBIAR CORREO [CU004]

Actor: Habitante (🧍)

Meta en contexto: Cambiar el correo al que le llegarán las notificaciones al usuario.

Condiciones previas:

- El 🧍 se encuentra en la pantalla “Módulos” (véase Caso de Uso “**CU003**”).

Disparador: Un habitante de la casa desea cambiar el correo al que llegan las notificaciones.

Prioridad: Baja, es una situación poco frecuente.

Disponible en: El primer incremento.

Frecuencia de uso: Poco frecuente.

Trayectoria Principal:

1. El 🧍 presiona el botón de opciones del dispositivo móvil.
2. La aplicación despliega un menú de opciones.
3. El 🧍 selecciona la opción “Cambiar correo electrónico”.
4. La aplicación despliega la pantalla “Cambio de Correo”.
5. El 🧍 introduce el nuevo correo y su contraseña actual.
6. El 🧍 presiona el botón “Cambiar correo”.
7. La aplicación valida que el formato de los datos introducidos sea correcto (véase Reglas del Negocio).
8. La aplicación envía los datos al módulo central para ser actualizados (Trayectoria Alterna **CU004_A**).
9. La aplicación despliega el mensaje “Cambio de correo exitoso” (Trayectoria Alterna **CU004_B**).
10. La aplicación despliega la pantalla “Módulos”.

Trayectorias Alternas:

CU004_A: El módulo central no responde.

1. La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde".
2. La aplicación se cierra.

CU004_B: El módulo central regresa un código de error.

1. La aplicación verifica el código de error
   a. La aplicación despliega el mensaje "El correo ya ha sido registrado anteriormente".
   b. La aplicación despliega el mensaje "La contraseña es incorrecta".
2. La aplicación borra los datos introducidos.

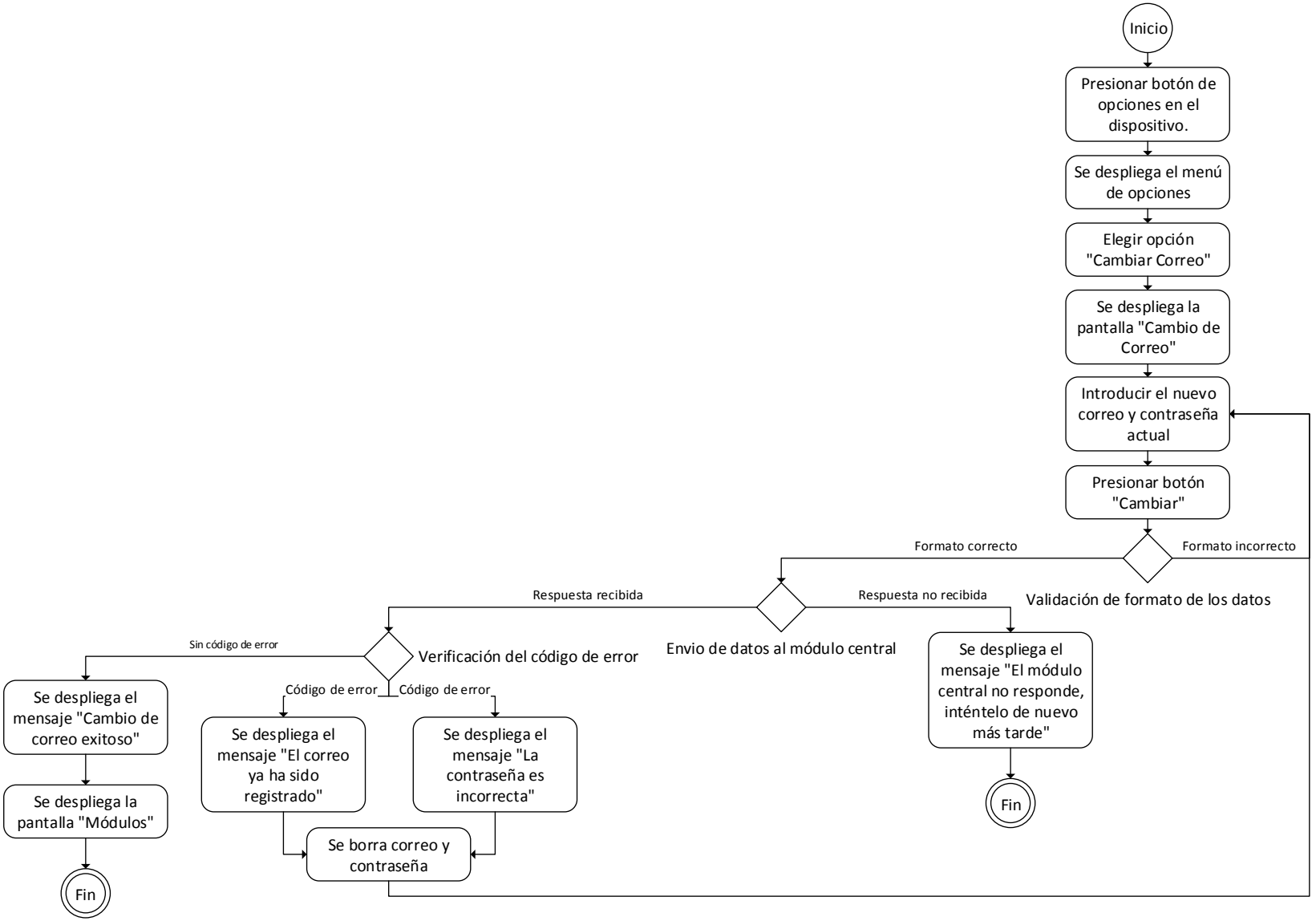


Diagrama 8. Diagrama de actividad para el caso de uso "Cambiar correo"

4.1.2.1.5. CAMBIAR CONTRASEÑA [CU005]

Actor: Habitante (🧍)

Meta en contexto: Cambiar la contraseña con la que el usuario inicia sesión.

Condiciones previas:

- El 🧍 se encuentra en la pantalla “Módulos” (véase Caso de Uso “**CU003**”).

Disparador: Un habitante de la casa desea cambiar su contraseña de inicio de sesión.

Prioridad: Media.

Disponible en: El primer incremento.

Frecuencia de uso: Poco frecuente.

Trayectoria Principal:

1. El 🧍 presiona el botón de opciones del dispositivo móvil.
2. La aplicación despliega un menú de opciones.
3. El 🧍 seleccionar la opción “Cambiar contraseña”.
4. La aplicación despliega la pantalla “Cambio de Contraseña”.
5. El 🧍 introduce su contraseña actual.
6. El 🧍 introduce su nueva contraseña 2 veces.
7. El 🧍 presiona el botón “Cambiar contraseña”.
8. La aplicación valida que el formato de los datos introducidos sea correcto (véase Reglas del Negocio).
9. La aplicación valida que las primeras dos contraseñas coincidan entre sí (Trayectoria Alterna **CU005_A**).
10. La aplicación envía los datos al módulo central para ser actualizados (Trayectoria Alterna **CU005_B**).
11. La aplicación despliega el mensaje “Cambio de contraseña exitoso” (Trayectoria Alterna **CU005_C**).
12. La aplicación despliega la pantalla “Módulos”.

Trayectorias Alternas:

CU005_A: Las contraseñas no son iguales.

1. La aplicación despliega el mensaje "Las contraseñas no son iguales, escríbalas nuevamente".
2. La aplicación borra los datos de los campos nueva contraseña.

CU005_B: El módulo central no responde.

1. La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde".
2. La aplicación se cierra.

CU003_C: El módulo central regresa un código de error.

1. La aplicación verifica el código de error
2. La aplicación despliega el mensaje "La contraseña actual es incorrecta".
3. La aplicación borra los datos introducidos.

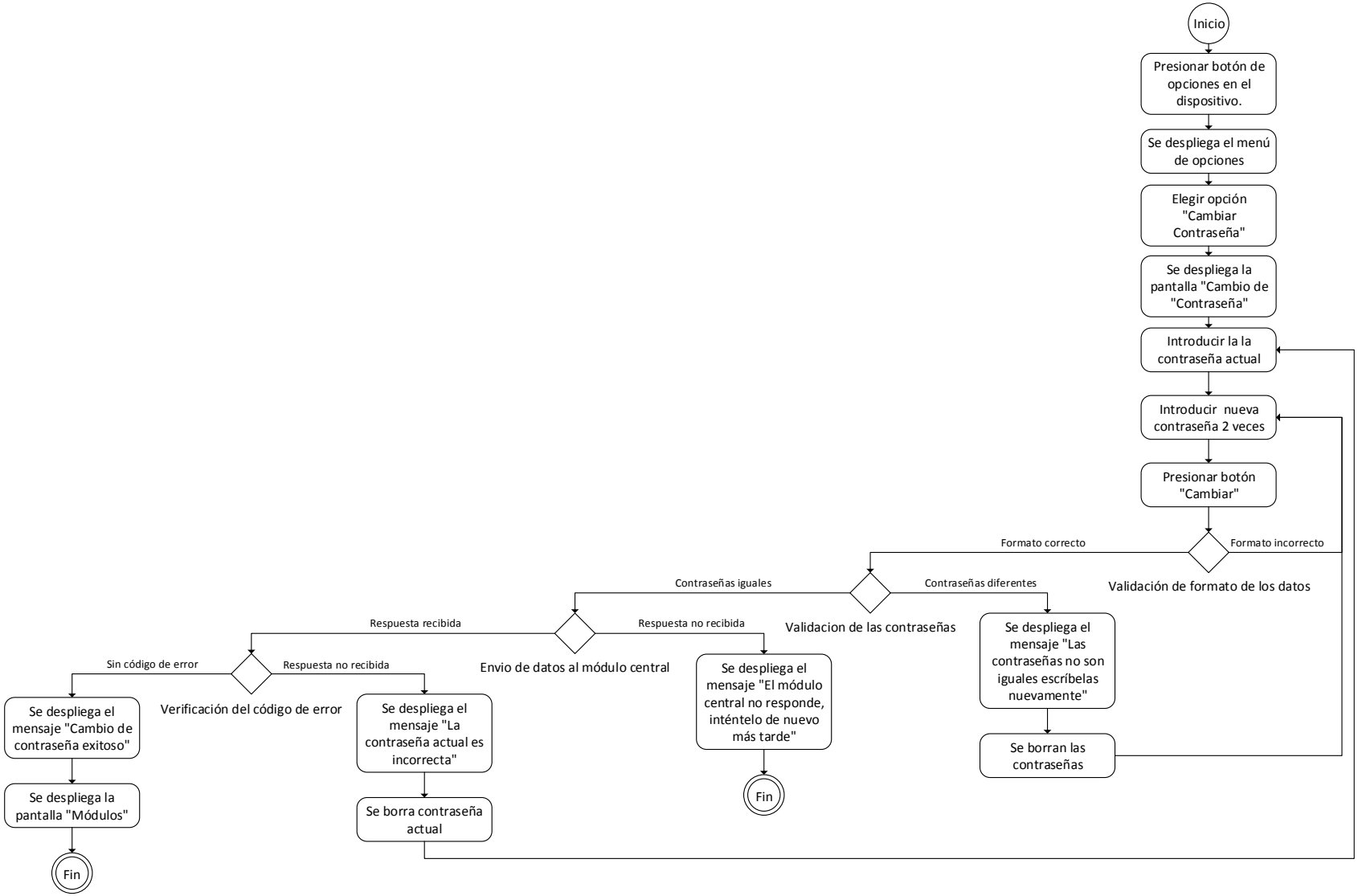


Diagrama 9. Diagrama de actividad para el caso de uso "Cambiar contraseña"

4.1.2.1.6. RECUPERAR CONTRASEÑA [CU006]

Actor: Habitante (🧍)

Meta en contexto: Obtener la contraseña del usuario.

Condiciones previas:

- El sistema se encuentra instalado y configurado.
- La aplicación *Domoduino* se encuentra instalada en el dispositivo móvil.
- El dispositivo móvil y el módulo central cuentan con conexión a Internet.
- El 🧍 se encuentra registrado en el sistema (véase Caso de Uso "**CU001**").

Disparador: Un habitante de la casa olvidó su contraseña y desea obtenerla.

Prioridad: Alta, se requiere para ingresar al sistema.

Disponible en: El primer incremento.

Frecuencia de uso: Poco frecuente.

Trayectoria Principal:

1. El 🧍 inicia la aplicación *Domoduino* en su dispositivo móvil (Trayectoria Alterna **CU006_A**).
2. La aplicación despliega la pantalla "Inicio de Sesión".
3. El usuario presiona el enlace "Olvidé mi contraseña".
4. La aplicación despliega la pantalla "Recuperación de Contraseña".
5. El usuario introduce su correo electrónico.
6. El 🧍 presiona el botón "Enviar".
7. La aplicación envía el correo al módulo central para validarlo y obtener la contraseña (Trayectoria Alterna **CU006_B** y **CU006_C**).
8. La aplicación despliega el mensaje "Se le ha enviado un correo electrónico con la contraseña".
9. La aplicación despliega la pantalla "Inicio de Sesión".

Trayectorias Alternas:

CU006_A: El dispositivo móvil pierde la conexión a Internet.

1. La aplicación despliega el mensaje "Revise su conexión a Internet".
2. La aplicación se cierra.

CU006_B: El módulo central no responde.

1. La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde".
2. La aplicación se cierra.

CU006_C: El correo no está registrado.

1. La aplicación despliega el mensaje "El correo proporcionado no existe, inténtalo nuevamente".
2. La aplicación borra los datos del correo.

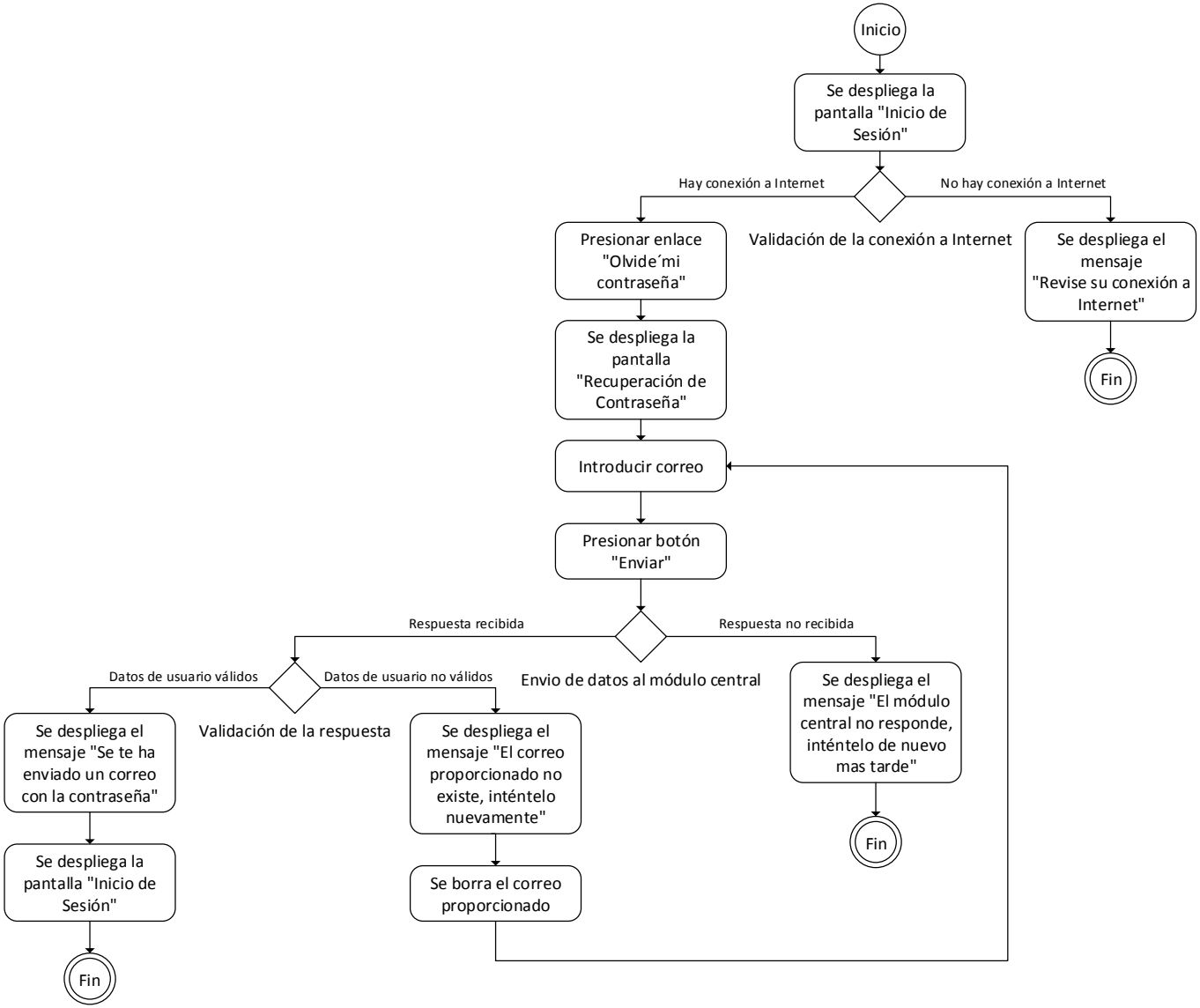


Diagrama 10. Diagrama de actividad para el caso de uso "Recuperar Contraseña"

A continuación se muestran los diagramas de secuencia que describen el funcionamiento del sistema.

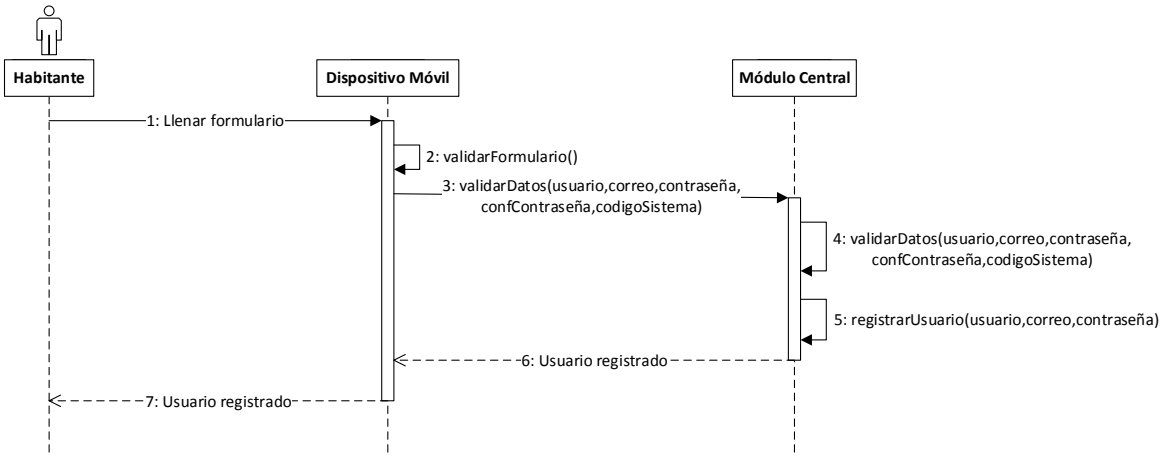


Diagrama 11. Diagrama de Secuencia para el registro de usuarios

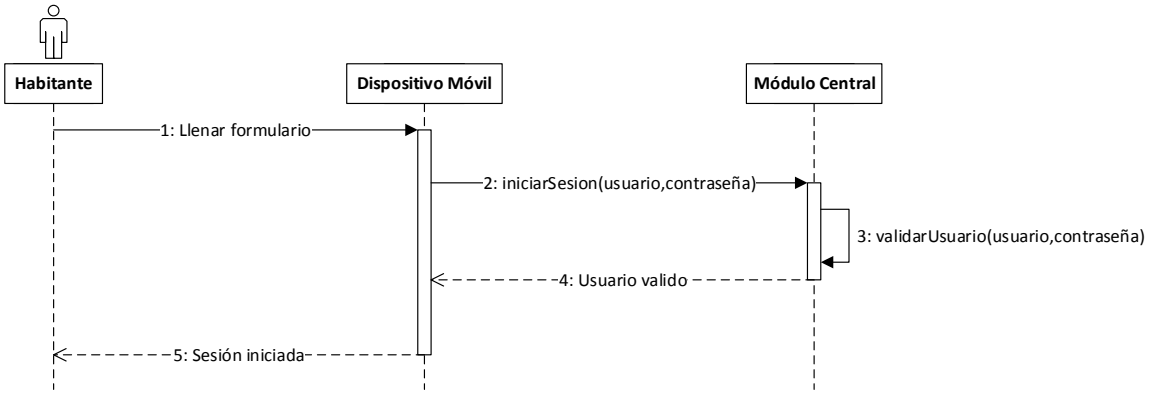


Diagrama 12. Diagrama de Secuencia para el inicio de sesión

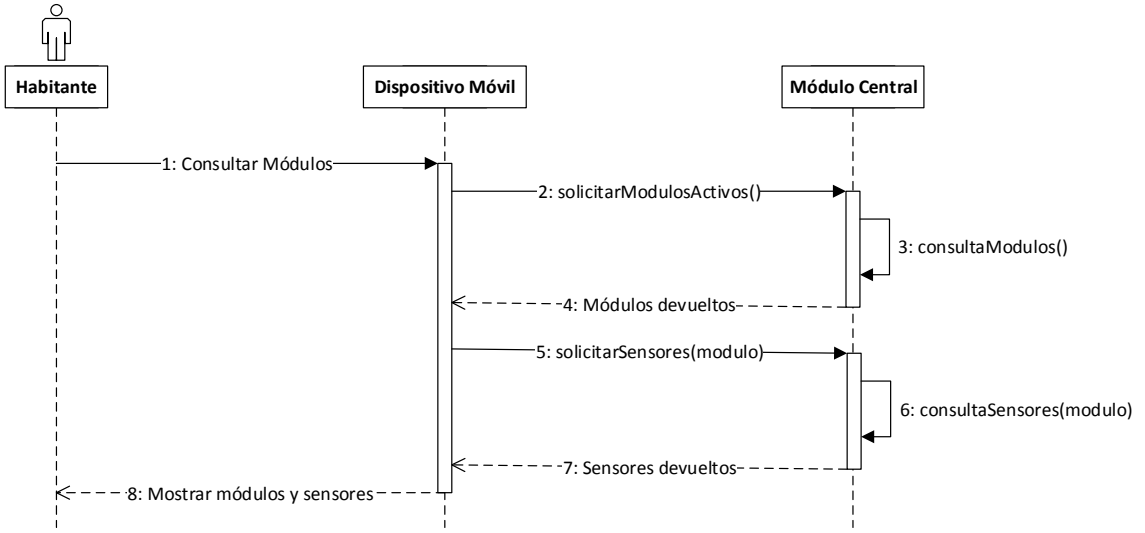


Diagrama 13. Diagrama de secuencia para consultar módulos

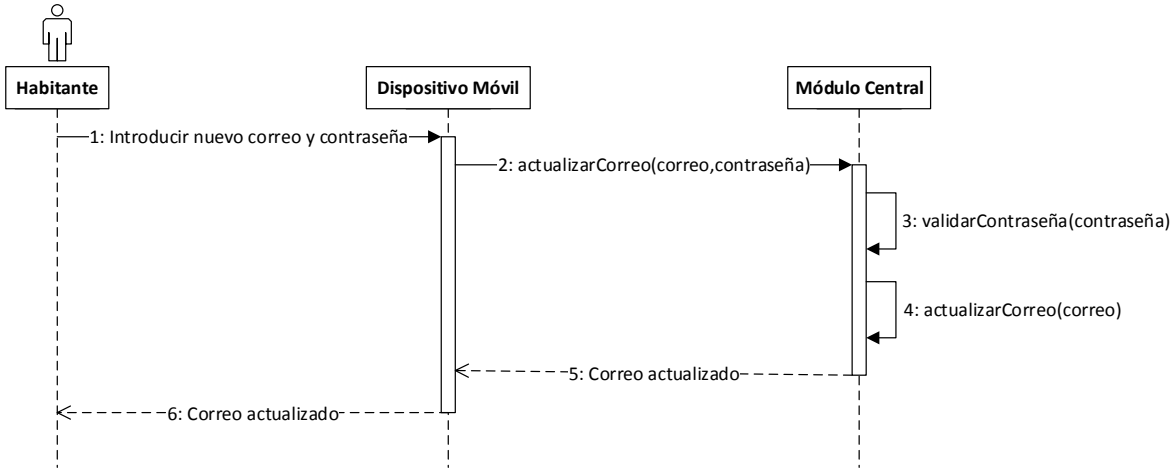


Diagrama 14. Diagrama de secuencia para cambiar correo

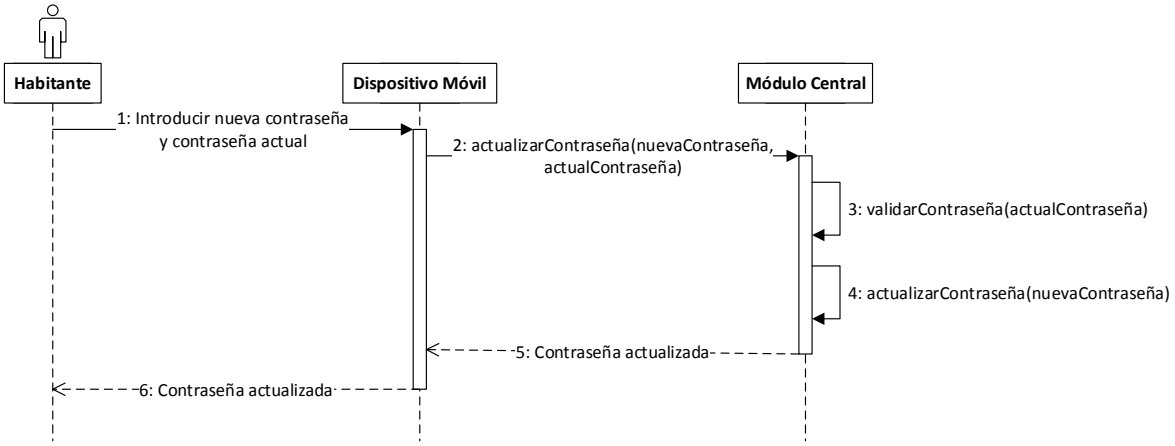


Diagrama 15. Diagrama de secuencia para cambiar contraseña

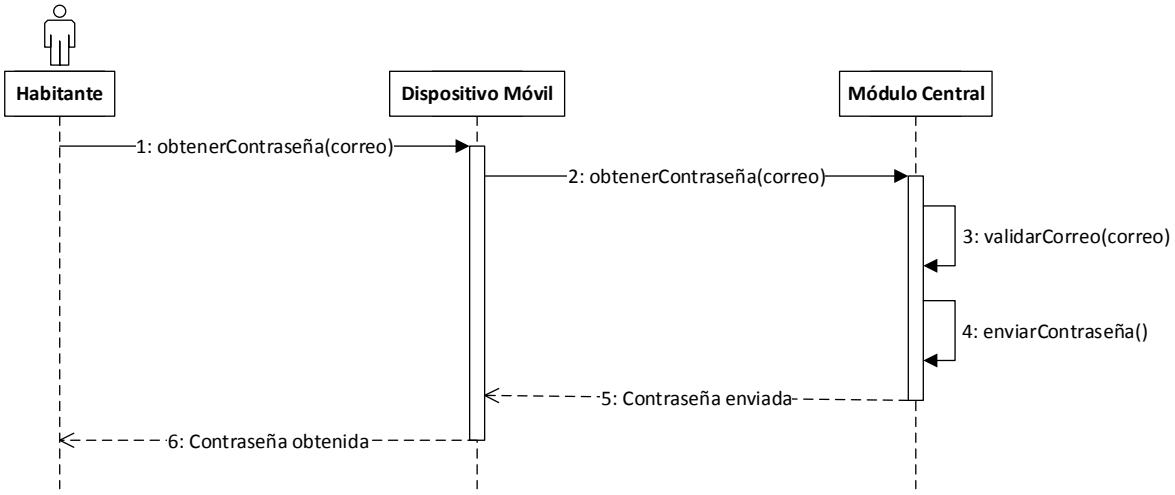


Diagrama 16. Diagrama de secuencia para recuperar contraseña

4.1.3. SERVICIOS WEB

Los servicios Web son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario [39].

Existen servicios Web de dos tipos: [40]

- SOAP (Simple Object Access Protocol) define un protocolo de comunicación estándar de especificación para el intercambio de mensajes basados en XML. SOAP utiliza diversos protocolos como HTTP y SMTP. El protocolo HTTP facilita que el modelo SOAP pase a través de firewalls y proxys sin hacer modificaciones al protocolo.
- REST (Representational State Transfer) describe una serie de principios de arquitectura bajo los cuales los datos pueden ser transmitidos sobre una superficie estandarizada (como HTTP). REST no contiene una capa adicional para mensajes y se enfoca en el diseño de reglas para crear servicios sin estado. Un cliente puede acceder a un recurso usando una única URI y la representación del recurso es retornada. Cuando se accede a recursos RESTful con el protocolo HTTP, la URL del recurso sirve como un recurso identificador y las operaciones estándar de HTTP (GET, PUT, DELETE, POST y HEAD) pueden ser realizadas sobre el objeto.

En este proyecto se utilizarán los servicios Web para realizar la comunicación entre un dispositivo móvil y un módulo central. Se decidió utilizar servicios Web RESTful debido a que Android no tiene soporte nativo para SOAP [41].

En el Diagrama 17 se muestran las clases presentes en los servicios Web.

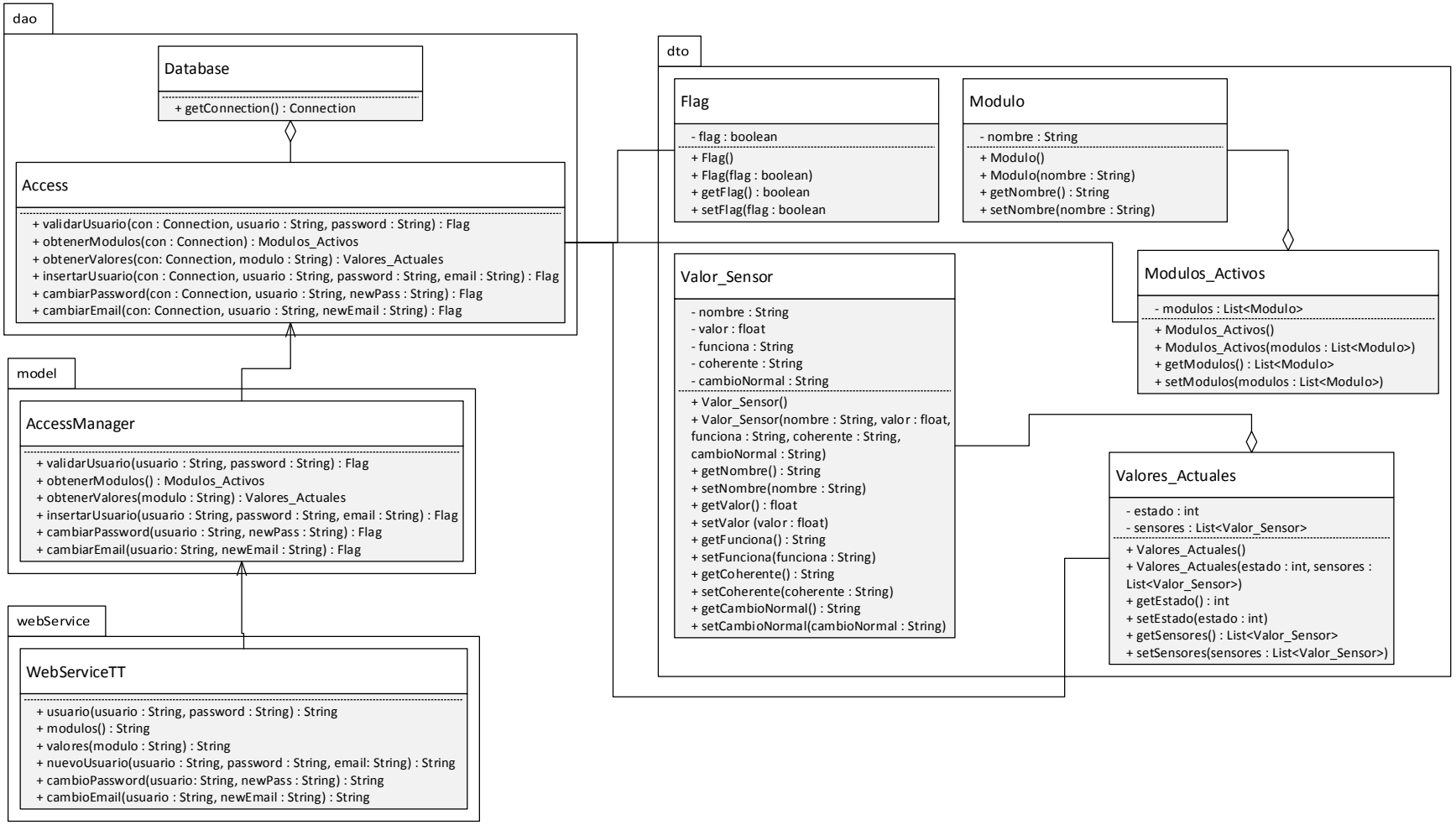


Diagrama 17. Diagrama de clases del servicio Web

4.2. INCREMENTO 2

En el segundo incremento del sistema se analizaron los métodos de calibración y obtención de valores para cada uno de los sensores y la forma de manipular un actuador para desarrollar la primera parte del código de Arduino.

4.2.1. CARACTERÍSTICAS DE LOS SENSORES

Parámetro	Valores	Unidad
Voltaje de operación	5	V
Resolución	0.1	%
Rango	0 <–> 99.9	%
Precisión	± 2	%
Tiempo de respuesta	< 5	seg

Tabla 15. Sensor de humedad (AM2302)

Parámetro	Valores	Unidad
Voltaje de operación	5	V
Resolución	0.1	°C
Rango	-40 <–> 80	°C
Precisión	± .5	°C
Tiempo de respuesta	< 10	seg

Tabla 16. Sensor de temperatura (AM2302)

Parámetro	Valores	Unidad
Voltaje de operación	5	V
Resolución	1	ppm
Rango	200 <–> 5000	ppm
Precisión	± 5	ppm
Tiempo de respuesta	< 1	seg

Tabla 17. Sensor de gas (MQ2)

Parámetro	Valores	Unidad
Voltaje de operación	5	V
Resolución	1	ppm
Rango	20 <–> 2000	ppm
Precisión	± 5	ppm
Tiempo de respuesta	< 1	seg

Tabla 18. Sensor de CO (MQ7)

Parámetro	Valores	Unidad
Voltaje de operación	5	V
Resolución	1	bit
Rango	0 <–> 1	bit
Precisión	± 1	bit
Tiempo de respuesta	< 1	seg

Tabla 19. Sensor de flama (Grove Flame Sensor)

4.2.2. Calibración y Obtención de Valores de Sensores

4.2.2.1. Sensor de Humedad y Temperatura

Este sensor no necesita una calibración, sin embargo al ser un dispositivo digital se tiene que analizar la trama enviada de éste para poder interpretar los valores. A continuación se explica qué se debe hacer para poder obtener la trama, cómo está constituida y cómo se interpreta.

Para realizar la petición de la trama se sigue el siguiente proceso:

1. Señal inicial: El microprocesador debe mandar una señal en alto por 250ms y luego en bajo por 20ms.
2. El sensor manda 3 señales: una en bajo, una en alto y al final una en bajo cada 80µs para indicarle al micro controlador que está listo para enviarle los datos.
3. El sensor envía la trama que contiene la información.

La trama está constituida por 5 bytes los cuales están distribuidos de la siguiente manera:

0000 0010	1001 0010	0000 0001	0000 1101	1010 0010
Parte alta Humedad	Parte baja Humedad	Parte alta Temperatura	Parte baja Temperatura	Suma de comprobación

Tabla 20. Formato de trama

Como se observa en la Tabla 20 primero llegan los datos (2 bytes) de la humedad empezando por la parte alta. Para obtener el valor de los bytes recibidos se debe aplicar la siguiente fórmula:

Humedad: 0000 0010 1001 0010 = 0x0292= 2×256 + 9×16 + 2 = 658
Humedad = 65.8%

Después se obtienen los datos de la temperatura (2 bytes). Se interpretan de la siguiente manera:

Temperatura. : 0000 0001 0000 1101 = 0x10D = 1×256 + 0×16 + 13 = 269
Temperatura = 26.9°C

En el caso de temperaturas bajo cero, si el primer bit es 1 se considera negativo el valor, pero se calcula de la misma manera que el caso anterior sustituyendo el 1 por 0 como se ve a continuación.

Dato recibido: 1 000 0000 0110 0101

Para calcular el valor se sustituye el 1 por 0.

Temperatura: 0000 0000 0110 0101 = 0x0065 = 6×16 + 5 = 101
Temperatura = -10.1°C

Para realizar la suma de comprobación se deben sumar los bytes de la humedad y la temperatura, como se muestra a continuación:

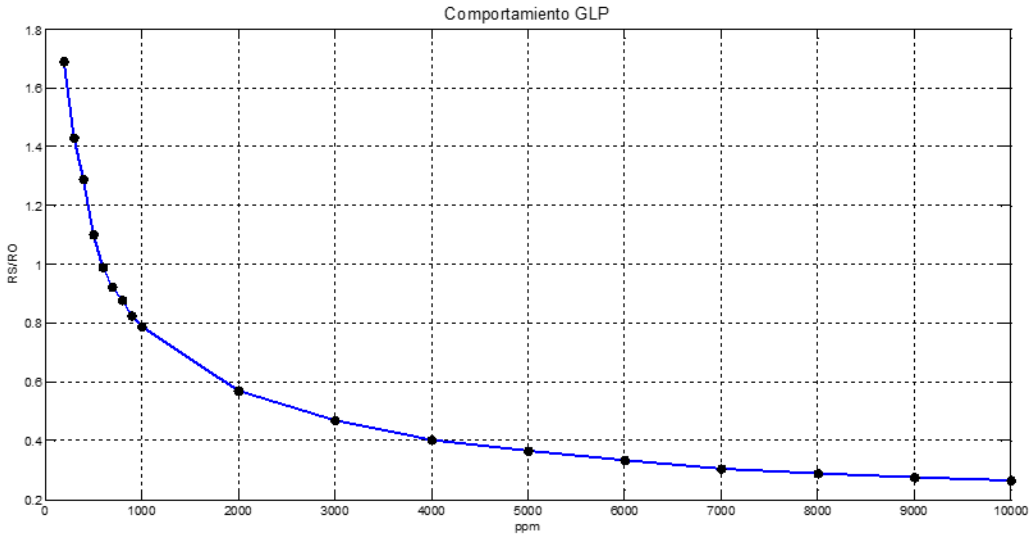
0000 0010 + 1001 0010 + 0000 0001 + 0000 1101 = 1010 0010

4.2.2.2. Sensor de Gas Licuado de Petróleo (GLP) y Monóxido de Carbono (CO)

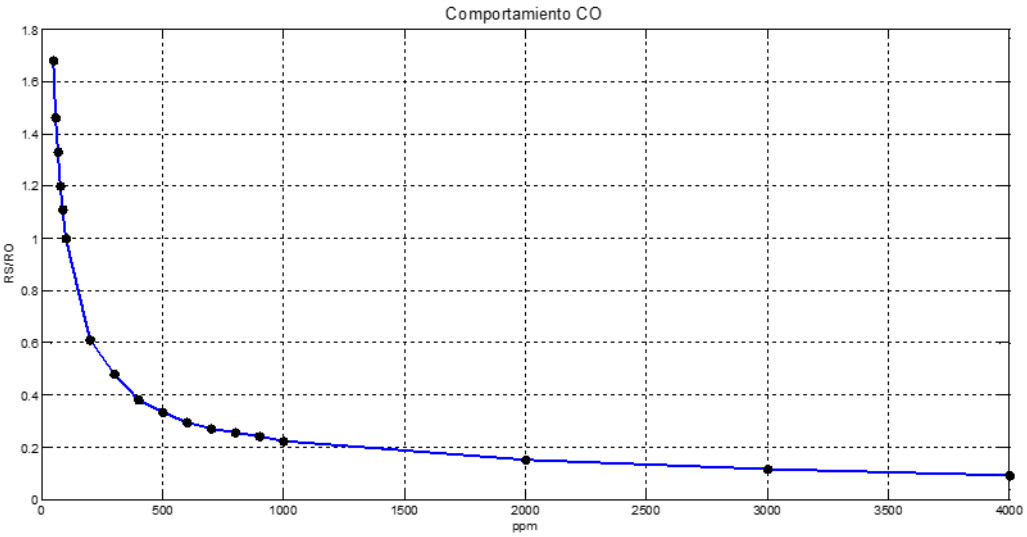
Estos dos sensores tienen procesos de calibración y obtención de datos parecidos así que se explicará un solo procedimiento aplicable a ambos sensores y se mostrarán los valores calculados.

Antes de proceder a la calibración, estos sensores se tienen que precalentar por 48 horas ininterrumpidas para que el sensor se adapte al ambiente donde se encuentra.

De acuerdo a la hoja de datos de los sensores, estos tienen un comportamiento no lineal como se muestra en las siguientes graficas:



Gráfica 1. Comportamiento del sensor de GLP



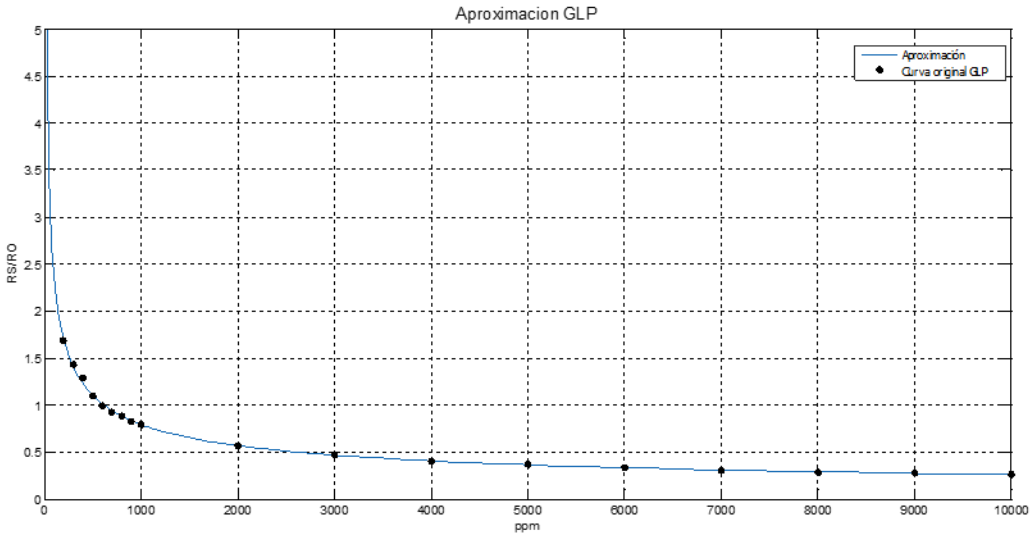
Gráfica 2. Comportamiento del sensor de CO

De estas graficas se deduce que hay que aplicar una función inversa para obtener los datos a partir de las partes por millón (ppm). Para obtener la función se tomaron muestras de las gráficas anteriores para realizar una aproximación con MATLAB quedando de la siguiente forma:

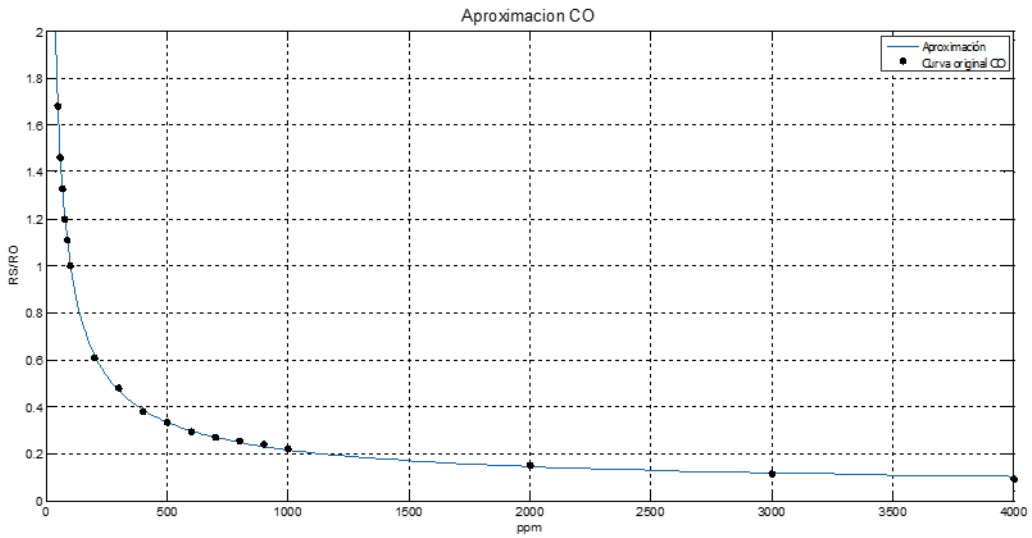
GLP RS/RO	GLP ppm	CO RS/RO	CO ppm
1.69	200	1.68	50
1.43	300	1.46	60
1.29	400	1.33	70
1.1	500	1.2	80
0.989	600	1.11	90
0.923	700	1	100
0.879	800	0.61	200
0.826	900	0.48	300
0.789	1000	0.38	400
0.569	2000	0.334	500
. 468	3000	0.293	600
0.403	4000	0.271	700
0.365	5000	0.255	800
0.333	6000	0.239	900
0.306	7000	0.221	1000
0.289	8000	0.151	2000
0.276	9000	0.115	3000
0.265	10000	0.09	4000

Tabla 21. Valores muestra

A continuación se muestran las gráficas de las aproximaciones realizadas con MATLAB.



Gráfica 3. Aproximación al comportamiento del sensor de GLP



Gráfica 4. Aproximación al comportamiento del sensor de CO

Como se observa en las gráficas se llegó a una muy buena aproximación donde la ecuación fue la siguiente:

$$y = ax^b + c \quad (1)$$

De la cual se obtuvieron los coeficientes a, b y c donde:

$$y = \frac{R_S}{R_O}$$

$$x = ppm$$

GLP	CO
$a = 21.52$	$a = 30.34$
$b = -0.4766$	$b = -0.7459$
$c = -0.007802$	$c = 0.04024$

Figura 17. Valores de a, b y c para cada sensor

Después de obtener la ecuación y los coeficientes de cada uno de los gases, se despeja "x" para poder obtener las ppm de los gases, quedando la formula como se muestra a continuación:

$$x = \left(\frac{y - c}{a}\right)^{\frac{1}{b}} \quad (2)$$

Sustituyendo los valores se tiene:

$$ppm = \left(\frac{\frac{R_S}{R_O} - c}{a}\right)^{\frac{1}{b}} \quad (3)$$

Para poder obtener $\frac{R_S}{R_O}$, la única incógnita de la cual depende nuestra ecuación se debe realizar el análisis del circuito.

En la Figura 18 se muestra cómo está conectado el sensor.

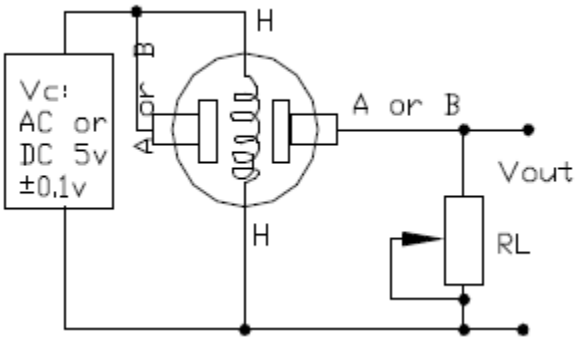


Figura 18. Circuito MQ2 y MQ7

Donde:
$R_L = Resistencia\ ajustable$
$R_S = Resistencia\ del\ sensor$
$Vo = Voltaje\ de\ salida$
$Vcc = Voltaje\ de\ entrada = 5V$

El circuito mostrado en la Figura 18 se puede representar como un divisor de voltaje sencillo, como el que se ilustra en la Figura 19.

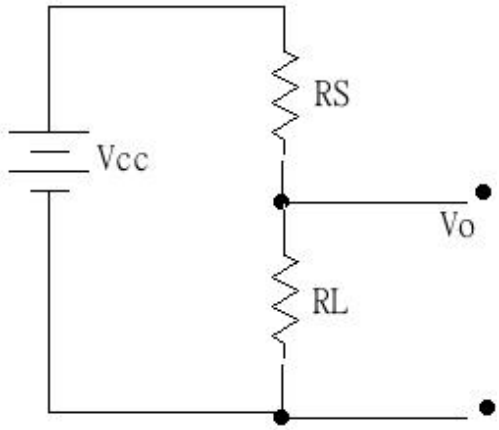


Figura 19. Divisor de voltaje

Por lo tanto se puede ocupar la ecuación que lo describe:

$$Vo = Vcc\left(\frac{R_L}{R_S + R_L}\right) \quad (4)$$

De acuerdo a la hoja de datos de los sensores nos recomienda que la resistencia de carga $R_L$de cada uno de los sensores GLP y CO deben ser 5kΩ y 10 kΩ respectivamente. La única incógnita es R_S, por lo tanto se tiene que despejar, quedando de la siguiente forma:

$$R_S = R_L\left(\frac{Vcc}{Vo} - 1\right) \quad (5)$$

Una vez que se sabe cómo obtener R_S, lo único que se tiene que hacer para obtener R_O es conocer la calidad del aire. Las hojas de datos de los sensores nos muestran los valores de $\frac{R_S}{R_O}$ en un ambiente libre de contaminantes mostrados en la Tabla 22.

GLP	CO
$\frac{R_S}{R_O} = 9.83$	$\frac{R_S}{R_O} = 11.64$

Tabla 22. Valores de resistencia en un ambiente libre de contaminantes

Para asegurar que el sensor se encuentra en un ambiente libre de contaminantes se consultaron las estadísticas de la CONAGUA las cuales señalan que las partes por millón de GLP y CO son:

GLP	CO
0 ppm	1.3 ppm

Tabla 23. Valores en un ambiente libre de contaminantes (ppm)

Con estos valores se puede determinar que la calidad del aire es buena y se pueden utilizar los valores de las hojas de datos. Por lo tanto solo se tiene que despejar R_O de las ecuaciones que aparecen en la Tabla 22. Los resultados se muestran la Tabla 24.

GLP	CO
$R_O = \frac{R_S}{9.83}$	$R_O = \frac{R_S}{11.64}$

Tabla 24. Valores de RO

Se sustituye R_S en la ecuación (5) quedando de la siguiente forma:

GLP	CO
$\boldsymbol{R_O = \frac{R_L\left(\frac{Vcc}{Vo} - 1\right)}{9.83}}$	$R_O = \frac{R_L\left(\frac{Vcc}{Vo} - 1\right)}{11.64}$

Tabla 25. Valores de RO en función de RL

Habiendo obtenido todos los valores necesarios para calibrar el sensor ya se pueden realizar las mediciones de cada uno de los gases. Para obtener las ppm de cada uno de los sensores solo se tiene que aplicar la formula (3), tomando en cuenta que la resistencia del sensor será la que variará, por lo tanto se tiene que sustituir la ecuación (5) en la ecuación (3) quedando de la siguiente manera:

$$ppm = \left(\frac{\frac{R_L\left(\frac{Vcc}{Vo} - 1\right)}{R_O} - c}{a}\right)^{\frac{1}{b}} \quad (6)$$

Los valores constantes para la ecuación son los siguientes:

GLP	CO
$Vcc = 5V$	$Vcc = 5V$
$Vo = Valor\ obtenido\ del\ circuito$	$Vo = Valor\ obtenido\ del\ circuito$
$R_L = 5k\Omega$	$R_L = 10k\Omega$
$R_O = Constante$	$R_O = Constante$
$a = 21.52$	$a = 30.34$
$b = -0.4766$	$b = -0.7459$
$c = -0.007802$	$c = 0.04024$

Tabla 26. Valores constantes para GLP y CO

Para obtener Vo desde la tarjeta Arduino, se realiza una lectura analógica que entrega un valor entre 0 y 1023, este valor se divide entre 1023 y luego se multiplica por 5.

$$Vo = \frac{lectura}{1023} * 5 \quad (7)$$

Sustituimos (7) en (6):

$$ppm = \left(\frac{R_L \left(\frac{Vcc}{\frac{lectura}{1023} * 5} - 1 \right)}{\frac{R_O}{a}} - c \right)^{\frac{1}{b}} \quad (8)$$

4.2.3. Manipulación de Actuadores

Como se ilustró en la Figura 15, dentro del sistema se cuenta con un módulo de actuadores que controla un sistema de protección contra incendios, el cual emula el comportamiento de los aspersores extintores de fuego y extractores de humo. Para activar un relevador únicamente se pone en alto el pin digital de la Arduino al que está conectado.

4.2.4. PROGRAMACIÓN EN ARDUINO

En este segmento se explica cuál es la estructura de un programa para Arduino para después explicar la codificación de cada una de las partes.

Un sketch (código de Arduino) tiene la siguiente estructura:

1. setup(): esta función es llamada una vez, cuando comienza el sketch. Este es un buen lugar donde se pueden realizar tareas de configuración como definir los pines a utilizar o inicializar bibliotecas.
2. loop(): esta función se llama una y otra vez y es el corazón de la mayoría de los sketches. Aquí que es donde se ejecuta todo el código.

En cada una de estas partes se pueden utilizar tanto variables como funciones ya sea que hayan sido declaradas dentro (internas) o fuera de ellas (globales).

El sketch que se desarrolló para este proyecto separa la codificación de cada sensor en bibliotecas para solamente ejecutar las funciones en el programa principal. En el Diagrama 18 se describe el funcionamiento general del programa; en este incremento solamente se profundizará en la calibración y obtención de los valores de los sensores y la manipulación de los actuadores.

Diagrama 18. Funcionamiento general del sistema

A continuación se muestran los diagramas de flujo correspondientes a la calibración (Diagrama 19), obtención de valores (Diagrama 20 y Diagrama 21), verificación de funcionamiento de los sensores (Diagrama 22), verificación de valores dentro del entorno (Diagrama 23), verificación de cambio coherente de los valores (Diagrama 24) y verificación de cambio respecto al valor anterior (Diagrama 25).

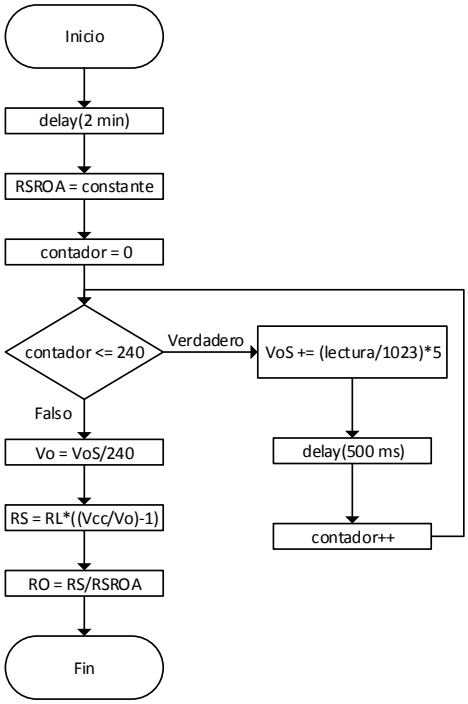


Diagrama 19. Calibración de sensores MQ2 y MQ7

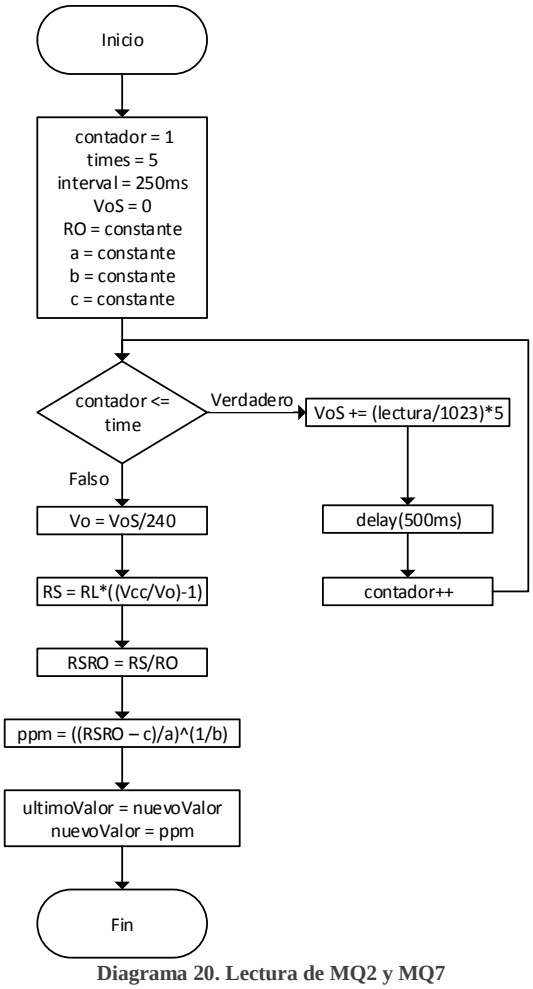


Diagrama 20. Lectura de MQ2 y MQ7

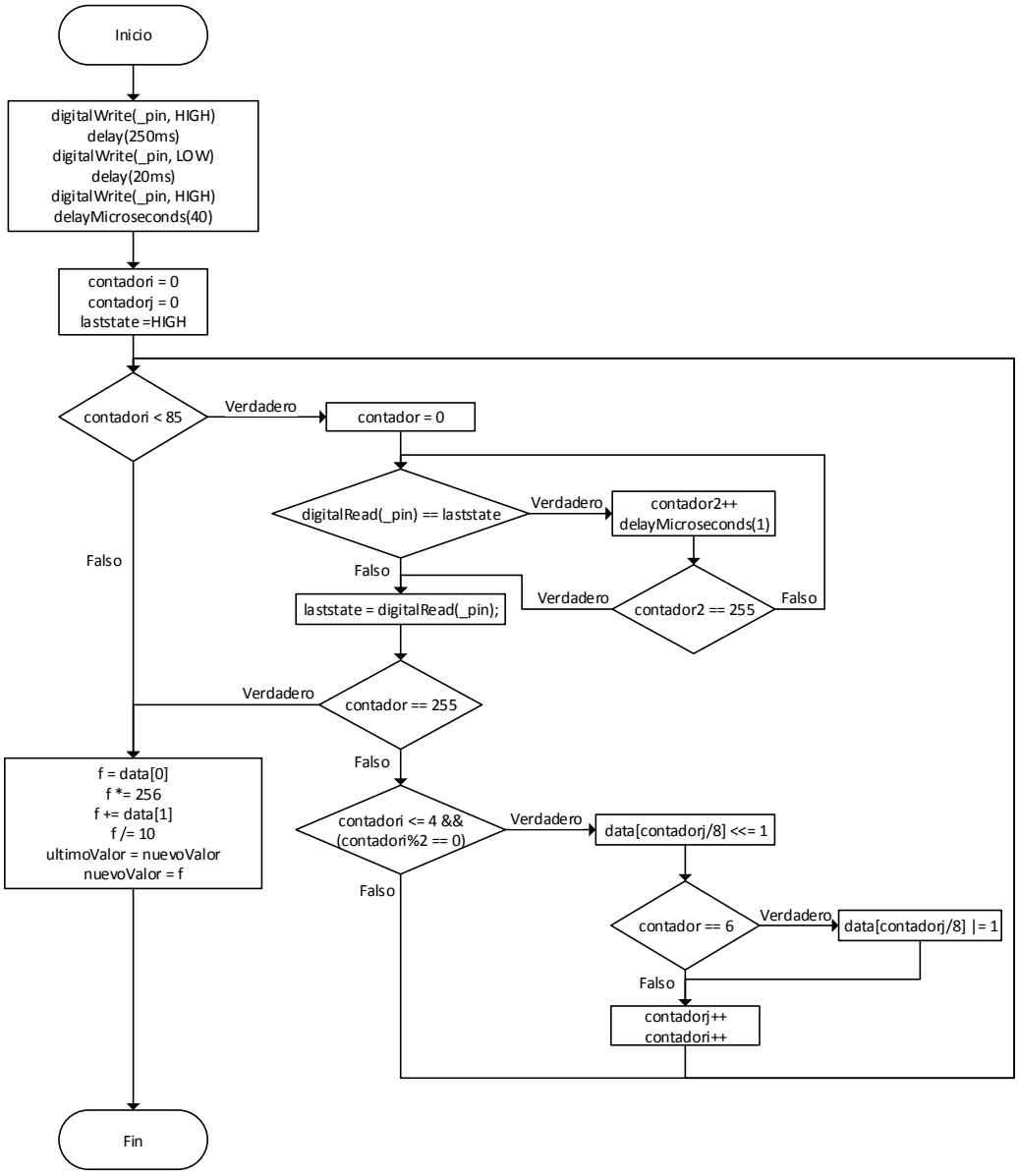


Diagrama 21. Lectura de AM2302

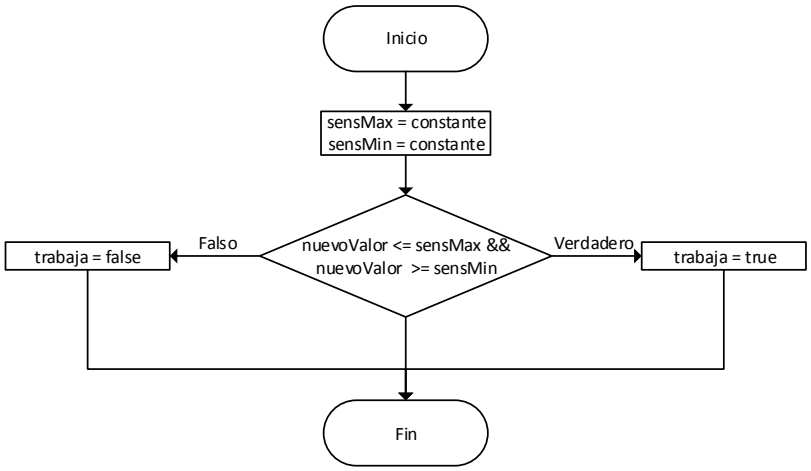


Diagrama 22. Verificación de funcionamiento de los sensores

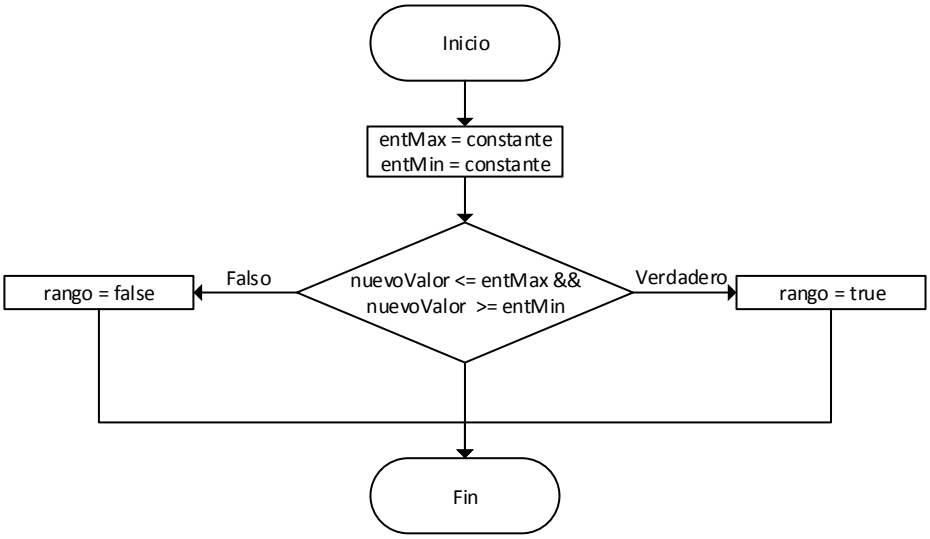


Diagrama 23. Verificación de valores dentro del entorno

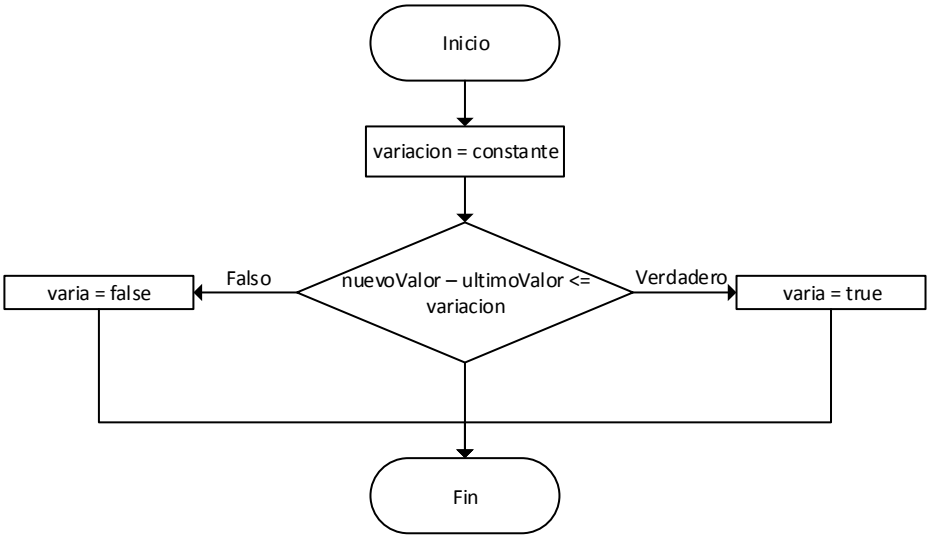


Diagrama 24. Verificación de cambio coherente de los valores

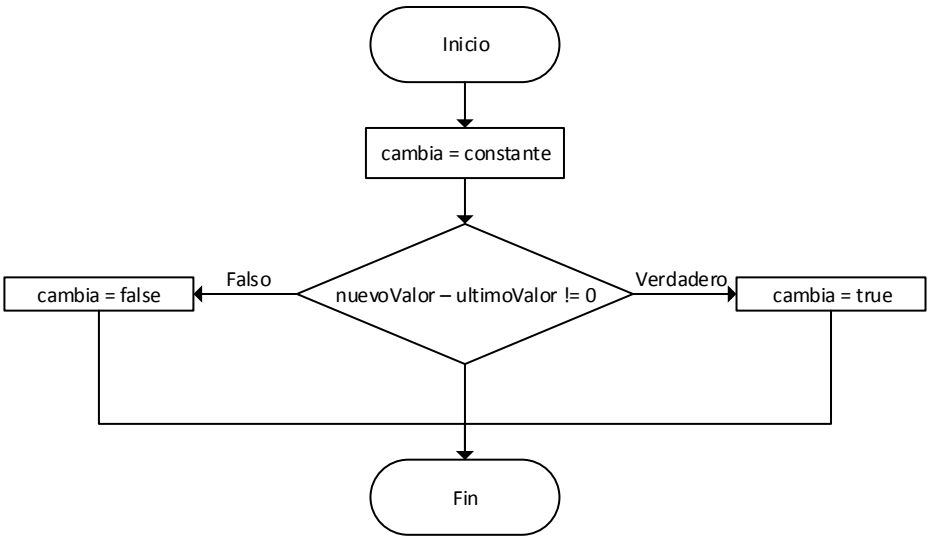


Diagrama 25. Verificación de cambio de valor respecto al valor anterior

Cabe destacar que los sensores adquiridos ya cuentan con etapa de caracterización, además de que el sensor de flama al ser digital no requiere de calibración y únicamente entrega valores 1 o 0 dependiendo de la presencia o ausencia de flama, respectivamente.

4.3. INCREMENTO 3

En el tercer incremento del sistema se desarrolló el algoritmo de detección de incendios basado en los detectores multi-criterio.

Antes de empezar a explicar la construcción del algoritmo se pretende dejar en claro por qué se usó esta metodología, habiendo probado la implementación del teorema de Bayes y el método Simplex.

4.3.1. TEOREMA DE BAYES

El teorema de Bayes en términos generales dice que se pueden vincular la probabilidad de A dado B con la probabilidad de B dado A, es decir, sabiendo la probabilidad de tener un dolor de cabeza dado que se tiene gripe se podría saber la probabilidad de tener gripe si se tiene un dolor de cabeza. Sabiendo esto se determinó que no se puede ocupar este teorema ya que está orientado a probabilidad y en este proyecto no se tienen dichas probabilidades y por lo tanto se tendrían que obtener por medio de experimentación y pruebas en un ambiente controlado.

4.3.2. MÉTODO SIMPLEX

Este algoritmo explicado de una forma general es un conjunto de métodos usados para resolver problemas en los cuales se busca el máximo de una función lineal sobre un conjunto de variables que satisfaga un conjunto de inecuaciones lineales. Este algoritmo tampoco se usó debido a que no solo se busca maximizar o minimizar una función hasta llegar al valor deseado, sino que cada valor obtenido es una parte importante para determinar diferentes soluciones.

4.3.3. DETECTOR MULTI-CRITERIO

Un detector multi-criterio es un dispositivo que contiene múltiples sensores que responden de manera separada a estímulos físicos tales como calor, humo o gases producto de la combustión, o emplea más de un sensor para medir el mismo estímulo. Este sensor es capaz de generar una sola señal de alarma a partir de los sensores empleados en el diseño tanto de manera independiente como en combinación. La señal de salida del sensor es evaluada matemáticamente para determinar cuándo una señal de alarma es garantizada. La evaluación puede ser ejecutada tanto en el detector como en la unidad de control. Este detector cuenta con una enumeración simple que establece su función primaria [42].

La detección multi-criterio: [42]

- Mejora el rendimiento de la detección ya que detecta flamas y fuegos que arden lentamente de igual manera.
- Mejora la inmunidad al ruido ya que detecta solamente el fuego.

La razón para utilizar este tipo de detectores es que generan una menor cantidad de falsas alarmas y tienen mayor probabilidad de detectar incendios de flamas lentas o que producen poco humo [43].

4.3.4. Algoritmo de Detección de Incendios

Para elaborar este algoritmo se empleó la metodología de decisión multi-criterio discreta, la cual consiste en elaborar una matriz que contiene los criterios y alternativas de decisión del problema.

Las alternativas son evaluaciones a las distintas características que presenta cada criterio, por lo tanto es necesario calificarlas de la manera más precisa posible. En este proyecto, la clasificación de escalas es cualitativa.

Conviene que los criterios cumplan las siguientes propiedades deseables:

- Exhaustividad: que no se haya olvidado criterio alguno que permita discriminar las alternativas.
- Coherencia: las preferencias globales del decisor deben ser coherentes con las preferencias según cada criterio.
- No redundancia: un conjunto de criterios, verificando las dos propiedades anteriores, es no redundante si la supresión de uno solo de los mismos implica que el subconjunto de los restantes viola alguna de tales propiedades.

Se realiza un análisis previo denominado preanálisis de dominación para seleccionar la mejor alternativa, pero dado que en este proyecto no existe una alternativa dominante, solo se realiza una ordenación de las alternativas para posteriormente realizar una ordenación final que incluya dentro de la matriz de decisión las alternativas para determinar si existe un incendio o riesgo de incendio.

La construcción del algoritmo para procesar las variables se compone principalmente de la matriz de decisión mostrada en la Tabla 27.

Alternativa 2	Alternativa 1	Tiempo	Flama	CO	GLP	Temperatura	Humedad
Normal	Normal	x	0	0	0	0	0
Normal	Normal	x	0	0	0	0	1
Normal	Normal	x	0	0	0	0	2
Normal	Exceso de temperatura	x	0	0	0	1	0
Normal	Exceso de temperatura	x	0	0	0	1	1
Normal	Exceso de temperatura	x	0	0	0	1	2
Normal	Exceso de GLP	x	0	0	1	0	0
Normal	Exceso de GLP	x	0	0	1	0	1
Normal	Exceso de GLP	x	0	0	1	0	2
Normal	Exceso CO	x	0	1	0	0	0
Normal	Exceso CO	x	0	1	0	0	1
Normal	Exceso CO	x	0	1	0	0	2
Normal	Normal	x	1	0	0	0	0
Normal	Normal	x	1	0	0	0	1

Normal	Normal	x	1	0	0	0	2
Riesgo de incendio	Exceso de GLP	x	0	0	1	1	0
Riesgo de incendio	Exceso de GLP	x	0	0	1	1	1
Riesgo de incendio	Exceso de GLP	x	0	0	1	1	2
Riesgo de incendio	Exceso de GLP	x	1	0	1	0	0
Riesgo de incendio	Exceso de GLP	x	1	0	1	0	1
Riesgo de incendio	Exceso de GLP	x	1	0	1	0	2
Posible incendio	Exceso de GLP	x	0	1	1	0	0
Posible incendio	Exceso de GLP	x	0	1	1	0	1
Posible incendio	Exceso de GLP	x	0	1	1	0	2
Posible incendio	Exceso de temperatura	x	1	0	0	1	0
Posible incendio	Exceso de temperatura	x	1	0	0	1	1
Posible incendio	Exceso de temperatura	x	1	0	0	1	2
Posible incendio	Exceso CO	x	1	1	0	0	0
Posible incendio	Exceso CO	x	1	1	0	0	1
Posible incendio	Exceso CO	x	1	1	0	0	2
Posible incendio	Exceso de GLP	x	1	1	1	0	0
Posible incendio	Exceso de GLP	x	1	1	1	0	1
Posible incendio	Exceso de GLP	x	1	1	1	0	2
Por determinar	Exceso CO	o	0	1	0	1	0
Por determinar	Exceso CO	o	0	1	0	1	1
Por determinar	Exceso CO	o	0	1	0	1	2
Por determinar	Exceso de GLP	o	0	1	1	1	0
Por determinar	Exceso de GLP	o	0	1	1	1	1
Por determinar	Exceso de GLP	o	0	1	1	1	2
Por determinar	Exceso de GLP	o	1	0	1	1	0
Por determinar	Exceso de GLP	o	1	0	1	1	1
Por determinar	Exceso de GLP	o	1	0	1	1	2
Por determinar	Exceso CO	o	1	1	0	1	0
Por determinar	Exceso CO	o	1	1	0	1	1
Por determinar	Exceso CO	o	1	1	0	1	2
Por determinar	Exceso de GLP	o	1	1	1	1	0
Por determinar	Exceso de GLP	o	1	1	1	1	1
Por determinar	Exceso de GLP	o	1	1	1	1	2
Incendio	CO Tóxico	x	0	2	0	0	0
Incendio	CO Tóxico	x	0	2	0	0	1
Incendio	CO Tóxico	x	0	2	0	0	2
Incendio	CO Tóxico	x	0	2	0	1	0
Incendio	CO Tóxico	x	0	2	0	1	1
Incendio	CO Tóxico	x	0	2	0	1	2
Incendio	CO Tóxico	x	0	2	1	0	0

Incendio	CO Tóxico	x	0	2	1	0	1
Incendio	CO Tóxico	x	0	2	1	0	2
Incendio	CO Tóxico	x	0	2	1	1	0
Incendio	CO Tóxico	x	0	2	1	1	1
Incendio	CO Tóxico	x	0	2	1	1	2
Incendio	CO Tóxico	x	1	2	0	0	0
Incendio	CO Tóxico	x	1	2	0	0	1
Incendio	CO Tóxico	x	1	2	0	0	2
Incendio	CO Tóxico	x	1	2	0	1	0
Incendio	CO Tóxico	x	1	2	0	1	1
Incendio	CO Tóxico	x	1	2	0	1	2
Incendio	CO Tóxico	x	1	2	1	0	0
Incendio	CO Tóxico	x	1	2	1	0	1
Incendio	CO Tóxico	x	1	2	1	0	2
Incendio	CO Tóxico	x	1	2	1	1	0
Incendio	CO Tóxico	x	1	2	1	1	1
Incendio	CO Tóxico	x	1	2	1	1	2

Tabla 27. Matriz de decisión

En la tabla anterior se muestran todas las posibles combinaciones basados en los siguientes criterios:

- Flama: Dado que este sensor solo detecta la presencia de flama solo se pueden tener 2 estados posibles.
  - 1: Flama detectada
  - 0: Flama no detectada
- CO: Este es el segundo sensor más importante ya que detecta uno de los principales productos de la combustión. Para este sensor se decidió tener 3 posibles estados que son:
  - 0: El sensor tiene mediciones normales (0ppm <-> 200ppm)
  - 1: El sensor tiene mediciones donde se necesita tomar precauciones (200ppm <-> 400ppm).
  - 2: El sensor tiene mediciones las cuales se consideran un incendio (más de 400ppm).
- GLP: Este sensor es muy importante ya que con él se puede saber si existe algún riesgo de incendio o alguna contaminación de este tipo de gas en el ambiente. Este sensor tiene solo 2 posibles estados:
  - 0: el sensor se encuentra en estado normal (0ppm <-> 1000ppm)
  - 1: el sensor detecta grandes cantidades de GLP (más de 1000ppm)
- Temperatura: La temperatura es un factor importante ya que esta variable se utiliza en el criterio del tiempo para saber si los cambios de temperatura son altos o bajos. Ya que la temperatura es una variable muy lenta solo se tienen 2 posibles estados.
  - 0: Significa que el sensor detecta temperatura normales (menos de 40°C)

    - 1: Significa que el sensor detecta una temperatura alta (más de 40°C), peligrosa para el ser humano.
- Humedad: esta variable es la menos utilizada pero aun así se toma en cuenta en conjunto con la temperatura para determinar fácilmente si existe un incendio o riesgo de uno. Para este sensor se decidió tener 3 posibles estados.
    - 0: La humedad en el ambiente es normal (más de 46%).
    - 1: La humedad en el ambiente es baja (20% <- >46%) pero se tiene que empezar a relacionar con la temperatura.
    - 2: La humedad es muy baja (0% <-> 20%) y posiblemente haya un riesgo de incendio.

Tomando en cuenta estas variables, se crearon las primeras alternativas que son:

- Normal: Todos los sensores se encuentran en un estado normal.
- Exceso de CO: El ambiente está contaminado de CO pero no es tóxico.
- Exceso de temperatura: El ambiente es muy caliente y existe un peligro para el usuario ya que puede provocar daños en la salud.
- Exceso de GLP: El ambiente está contaminado de este gas y es muy dañino para la salud.
- CO Tóxico: El ambiente es muy tóxico para la salud del ser humano.

Como se puede apreciar en este primer análisis solo se determinaron las posibles alternativas para las variables más importantes o que pueden causar un daño a la salud. Como cada una de estas alternativas es válida por separado lo único que se hace es una ordenación de menor a mayor del nivel de peligro que representan.

Después se procede a hacer un segundo análisis el cual ahora está dirigido a conjuntar todas las variables para saber si existe un incendio, riesgo de incendio o la incertidumbre de que haya uno. De igual manera que el análisis anterior se hará un ordenamiento de menor a mayor dependiendo del riesgo que represente la alternativa.

Las nuevas alternativas que se generaron son:

- Normal: No existe algún riesgo de incendio.
- Riesgo de incendio: Está determinado por una alta temperatura y niveles altos de concentración de GLP.
- Posible incendio: En esta alternativa existe la incertidumbre de que haya un incendio ya que los criterios no son suficientes para catalogarlo como un incendio o que haya riesgo de uno.
- Por determinar: Esta alternativa es muy similar a la anterior pero se puede llegar a determinar si es un incendio introduciendo un nuevo criterio el cual sería el cambio de la temperatura en un intervalo de tiempo.
- Incendio: Aquí ya se determinó que es un incendio.

Para finalizar solo se toman los casos diferentes de las combinaciones entre alternativas para después aplicar una o varias acciones a los actuadores, la matriz resultante se muestra en la Tabla 28.

Alternativa 2	Alternativa 1	Ventilador	Aspersor	Apagar Electricidad
Normal	Normal	0	0	0
Normal	Exceso de temperatura	1	0	0
Normal	Exceso de GLP	1	0	0
Riesgo de incendio	Exceso de GLP	1	0	0
Normal	Exceso CO	1	0	0
Por determinar	Exceso CO	?	?	?
Posible incendio	Exceso de GLP	1	0	0
Por determinar	Exceso de GLP	?	?	?
Incendio	CO Tóxico	0	1	1
Posible incendio	Exceso de temperatura	1	0	0
Posible incendio	Exceso CO	1	0	0

Tabla 28. Matriz con acciones

En la tabla anterior los signos de interrogación corresponden a incendio o a posible incendio según sea el caso que se determinó.

En el Diagrama 26 se muestra el algoritmo para la determinación de la Alternativa 1 y en el Diagrama 27 se muestra el algoritmo para la determinación de la Alternativa 2, con base en el resultado de la Alternativa 1.

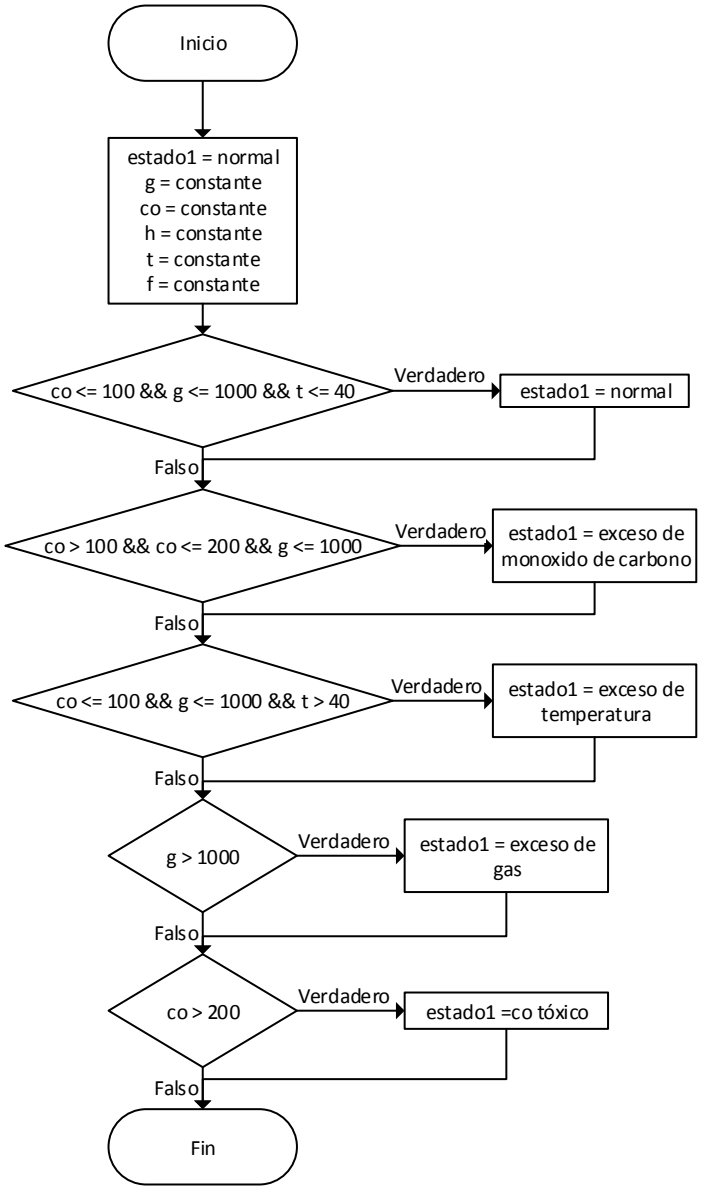


Diagrama 26. Algoritmo para la determinación de la Alternativa 1

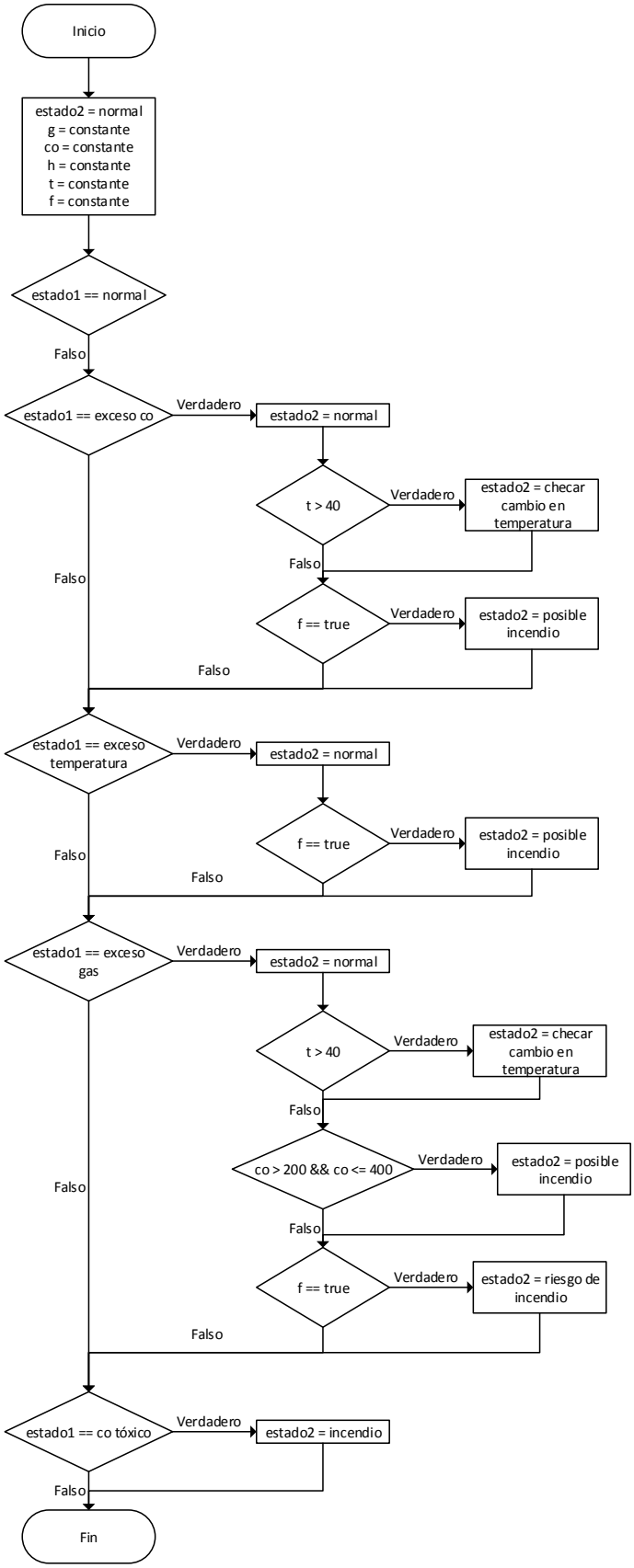


Diagrama 27. Algoritmo para la determinación de la Alternativa 2

4.4. INCREMENTO 4

En el cuarto incremento del sistema se desarrolló la parte de código de Arduino que permite la comunicación entre los módulos de sensores, de actuadores y el módulo central por medio de Xbee. También se desarrolló la aplicación que recibe los mensajes en el módulo central.

4.4.1. CONSTRUCCIÓN DE TRAMA PARA EL MÓDULO DE SENSORES

Para construir la trama lo primero que se analizó son los datos que van a ser enviados. En la Tabla 29 se muestran los datos a enviar y el espacio que ocupan en la trama.

Información	Tamaño
Numero de dispositivos que funcionan.	4 bits
Numero de dispositivos que no funcionan.	4 bits
Número del sensor	4 bits
Funcionamiento del sensor	1 bit
Funcionamiento dentro del entorno	1 bit
Variación coherente respecto al valor anterior	1 bit
Cambio respecto al valor anterior	1 bit
Valor de los sensores que si trabajan	4 bytes
Notificación de peligro individual	4 bits
Notificación de peligro global	4 bits

Tabla 29. Información de la trama

Al principio de la trama se ingresan los primeros dos datos (número de dispositivos que funcionan y no funcionan), presentes en el primer byte, a este byte se le llama Cabecera.

Después se tiene un byte de información con el número de sensor y los diferentes estados de funcionamiento, variación y cambio de los sensores, a este byte se le llama NumeroDatos

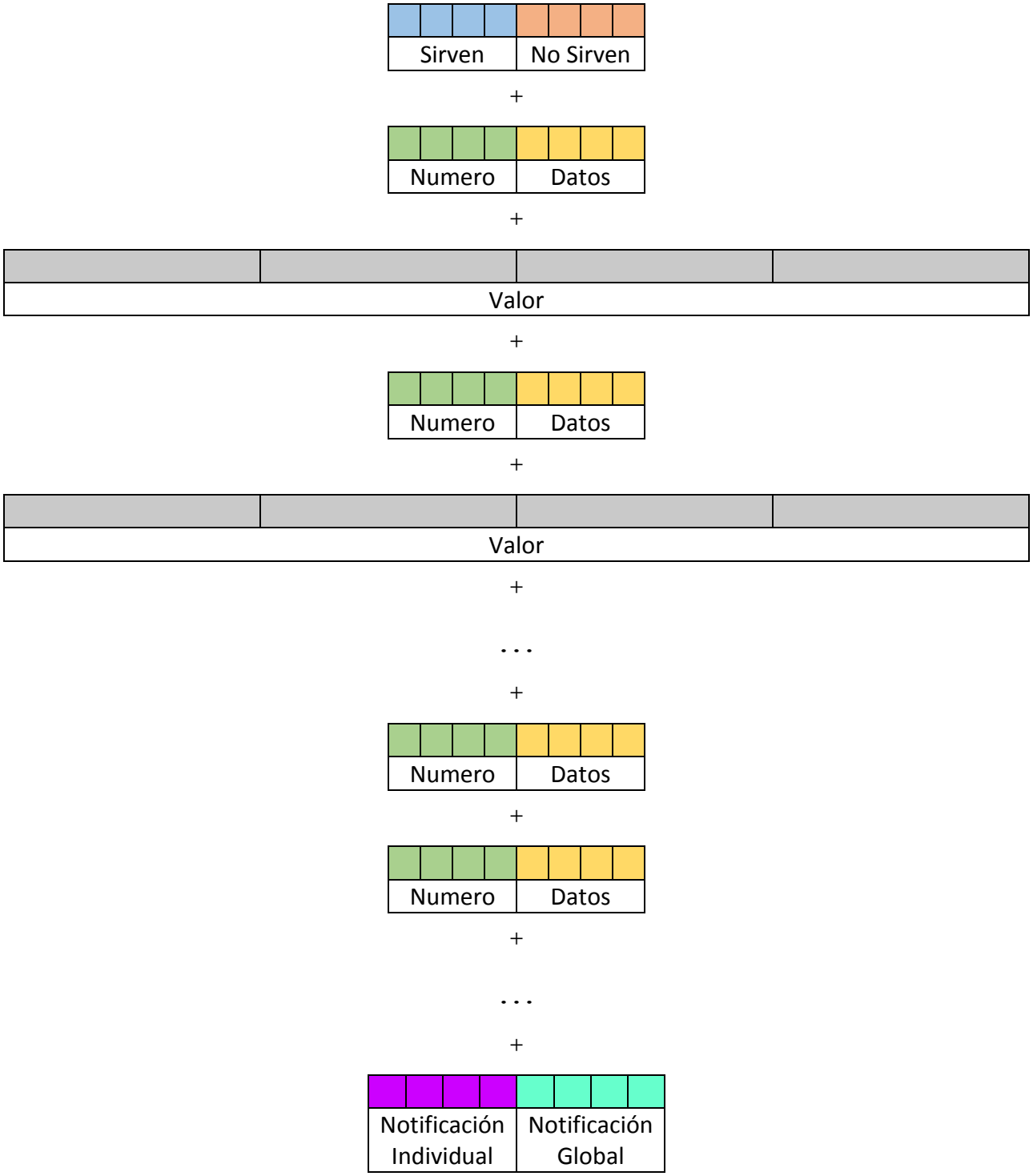
Posteriormente se tiene el valor de los sensores que sí trabajan que son cuatro bytes para poder representar números flotantes. En el caso de los sensores que no trabajan no es necesario enviar algún valor ya que es un valor erróneo. A estos cuatro bytes se les llama Valores.

Al finalizar la trama se tienen dos notificaciones en un byte, cada una de cuatro bits que indican el nivel de peligro individual y global del módulo en que se encuentran, a este byte se le llamara Notificaciones.

Después de definir cada uno de los componentes se procede a calcular el tamaño de la trama.

Cabecera = 1 byte
NumeroDatos = 1 byte
Valores = 4 bytes
Notificaciones = 1 byte
Tamaño = Cabecera + ((Valores + NumeroDatos) * Trabajan) + No Trabajan + Notificaciones

Para la construcción de la trama se introducen primero los sensores que sirven y luego los que no sirven para que el receptor sepa cómo leerla. A continuación se muestra la manera en que se agregan los datos a la trama.



El tamaño máximo de la trama es de 27 bytes si los 5 sensores funcionan, el tamaño mínimo seria de 7 bytes en el caso de que ningún sensor funcionara.

En el Diagrama 26 se muestra el proceso de construcción de la trama que se envía del módulo de sensores al módulo central.

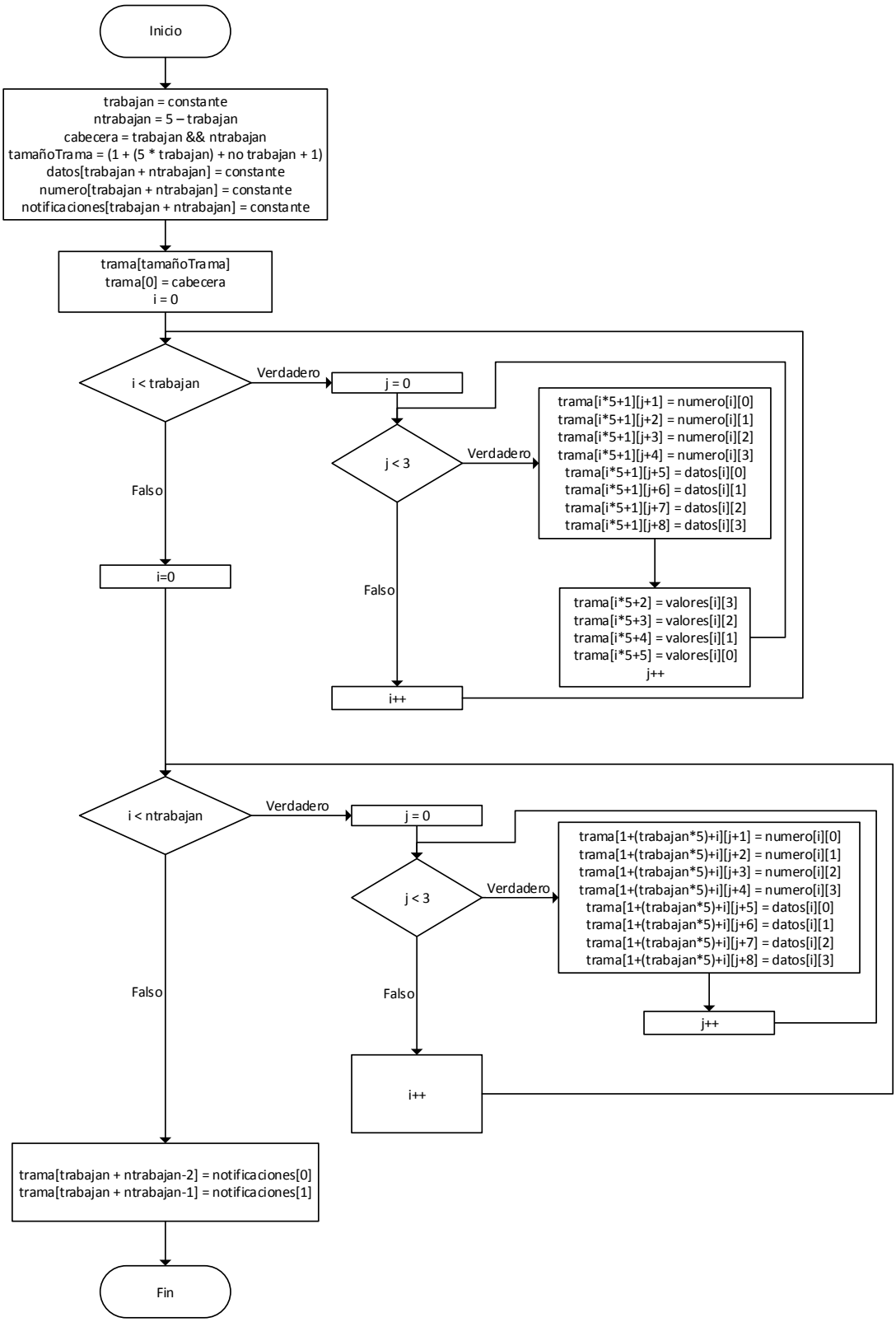


Diagrama 28. Construcción de la trama de información de sensores

4.4.2. Construcción de Trama para el Módulo de Actuadores

La construcción de esta trama es más sencilla ya que solo se envía el número del actuador y el estado deseado de este. La trama se muestra a continuación.

Como se observa en la imagen anterior el tamaño de la trama siempre es de un byte.

Esta misma trama es retransmitida por el módulo de actuadores hacia el modulo central para guardar el estado de los actuadores

En el Diagrama 29 se muestra el proceso de construcción de la trama que se envía del módulo de sensores al módulo de actuadores.

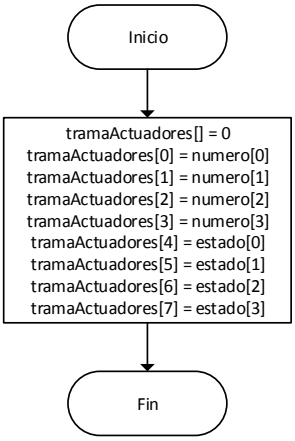


Diagrama 29. Construcción de la trama de información de actuadores

4.4.3. Envío de Datos

Para realizar el envío de las tramas se tiene sigue una serie de pasos para que se haga de manera correcta, aunque también existe la posibilidad de no enviar la trama.

1. Enviarle la trama al receptor.
2. Esperar a que responda el receptor en los siguientes 100ms. Si este no lo hace quiere decir que el receptor está apagado o fuera de alcance.
3. Verificar el estatus del mensaje recibido en el punto anterior. Si este estatus es igual a "ZB_TX_STATUS_RESPONSE" se procederá a leer el contenido del mensaje.
4. Verificar el contenido del mensaje. Si se recibió el mensaje "SUCCESS" quiere decir que la trama se envió correctamente y si no es posible que el receptor no la haya recibido correctamente.

En el Diagrama 30 se muestra el proceso de envío de la trama formada.

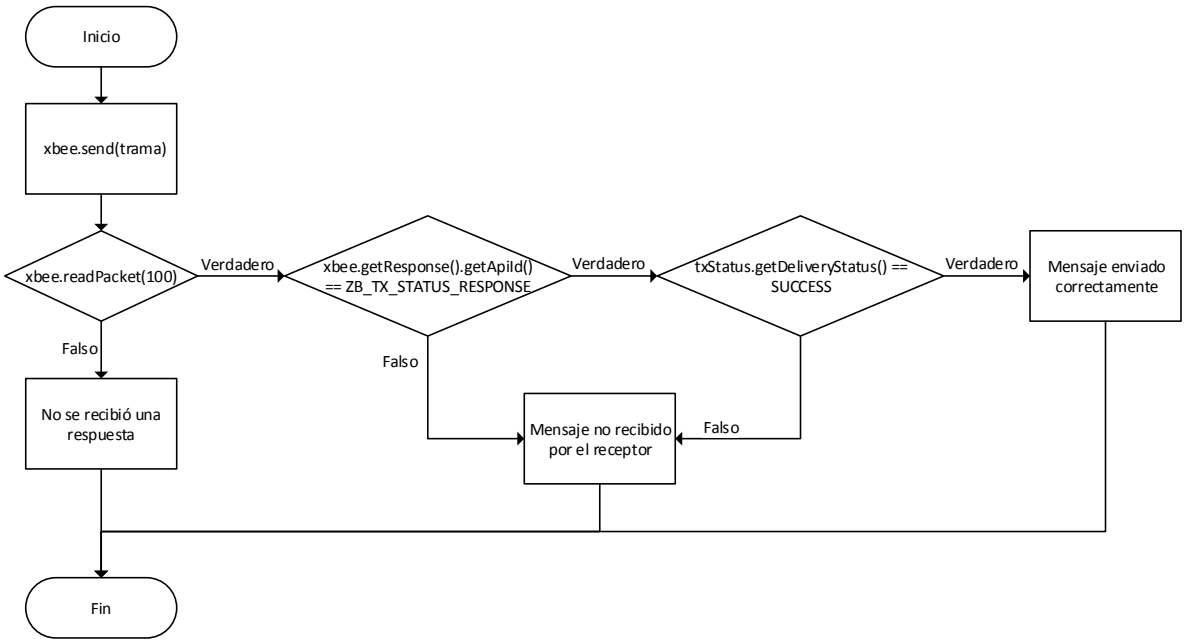


Diagrama 30. Envío de la trama

4.4.4. RECEPCIÓN DE DATOS

Para realizar la recepción de las tramas se sigue la siguiente serie de pasos:

1. Recibir el mensaje del emisor (la aplicación se encuentra escuchando).
2. Verificar el estatus del mensaje recibido. Si este estatus es igual a "ZNET_RX_RESPONSE" se procederá a leer el contenido del mensaje.
3. Verificar el contenido del mensaje. Si la trama es de tamaño de 1 byte significa que la envió el módulo de actuadores, por lo que se procede a cambiar el estado del actuador en la base de datos. Si la trama es de otro tamaño significa que la envió un módulo de sensores y se procede a guardar las nuevas mediciones en la base de datos.

En el Diagrama 31 se muestra el diagrama de clases de la aplicación que recibe los mensajes de Xbee. Para el desarrollo de esta aplicación se utilizó la API de Xbee para Java.

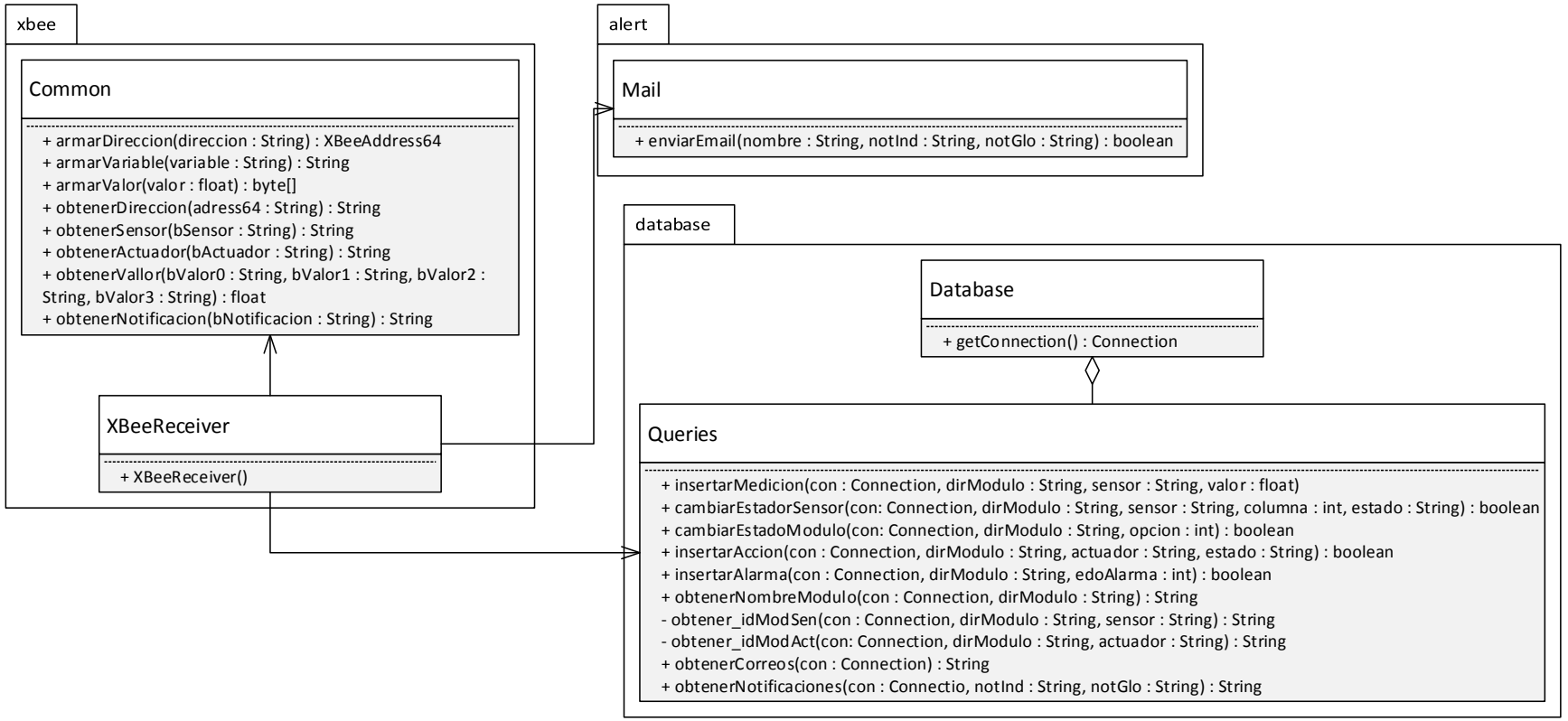


Diagrama 31. Diagrama de clases de la aplicación que recibe los mensajes Xbee en el módulo central

4.5. INCREMENTO 5

En el quinto incremento se desarrolló la funcionalidad que envía las notificaciones al usuario cuando el sistema se encuentra en estado de alerta. Se decidió que las notificaciones fueran por correo electrónico ya que para enviar SMS se necesita instalar un módulo extra y un chip de alguna compañía telefónica.

4.5.1. ENVÍO DE CORREO ELECTRÓNICO

Para enviar un correo electrónico es necesario utilizar el protocolo SMTP (Simple Mail Transfer Protocol) que es un protocolo de red para el intercambio de correo electrónico entre computadoras u otros dispositivos. Fue definido en el RFC 2821 y su funcionamiento se da en línea.

En un principio se pensó en crear una aplicación nueva para el envío de correo electrónico, pero debido a que este proceso debe realizarse inmediatamente después de que se recibe una notificación de peligro se decidió solamente agregar la funcionalidad a la aplicación que recibe los mensajes Xbee, como se observa en el Diagrama 31. Para el desarrollo de esta aplicación se utilizó la API de correo electrónico (JavaMail) para Java.

En la Figura 20 se muestra un correo generado por el sistema.

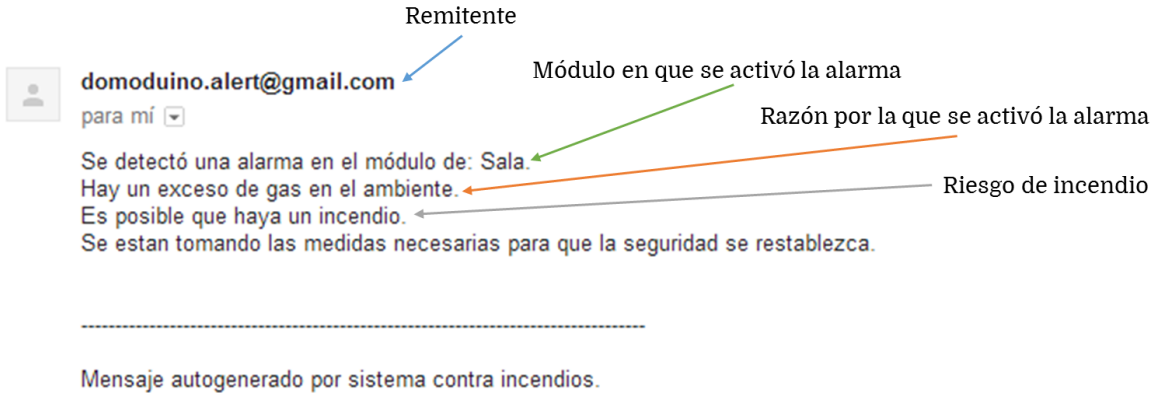


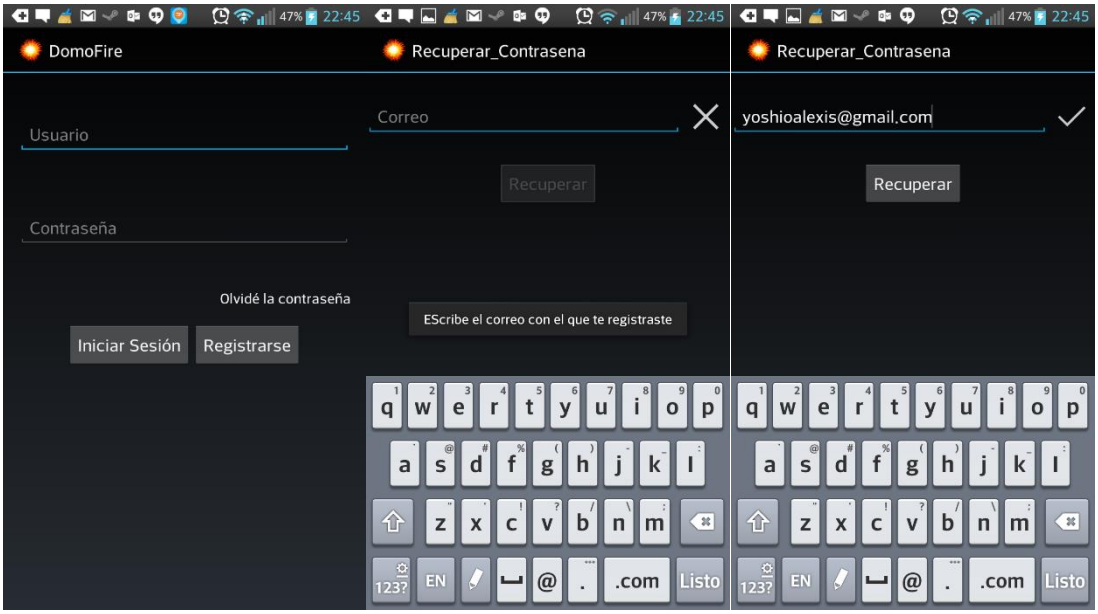
Figura 20. Correo de prueba

CAPÍTULO V: IMPLEMENTACIÓN DEL SISTEMA

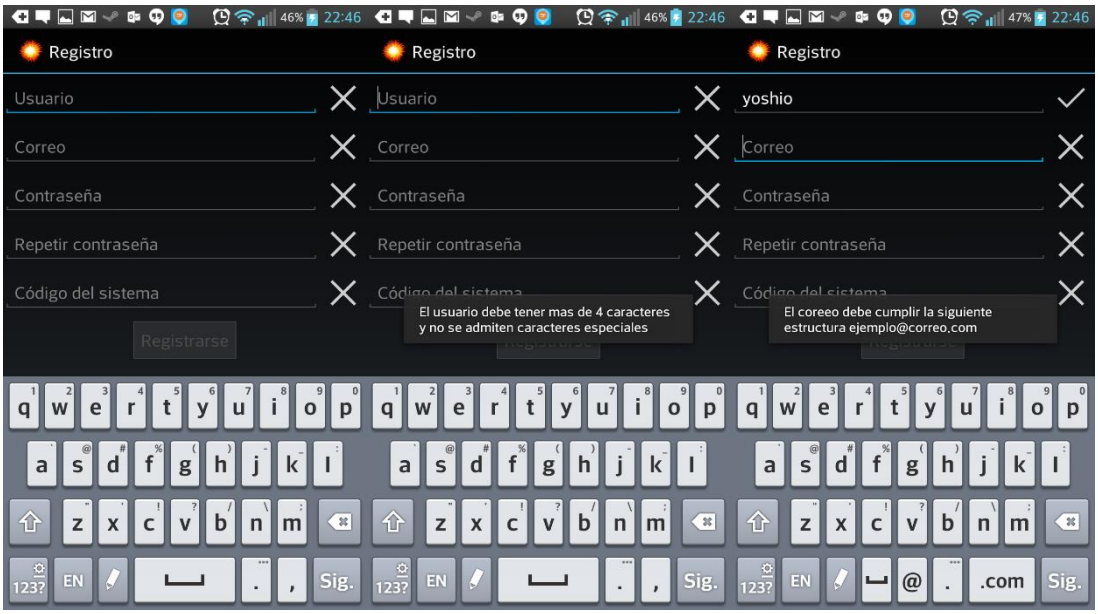
5.1. Aplicación Móvil

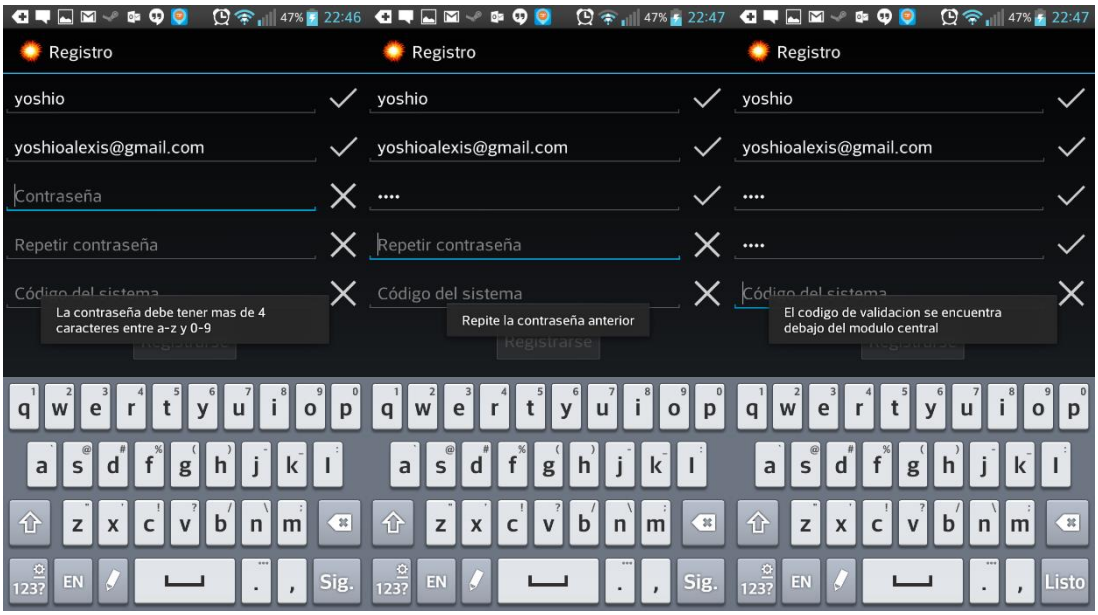
A continuación se muestran las pantallas en funcionamiento de la aplicación móvil.

5.1.1. Recuperación de Contraseña

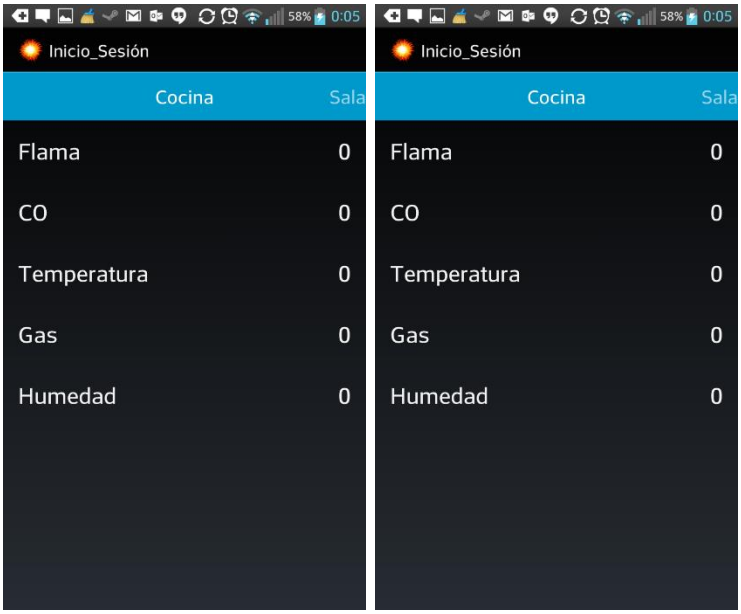


5.1.2. Registro

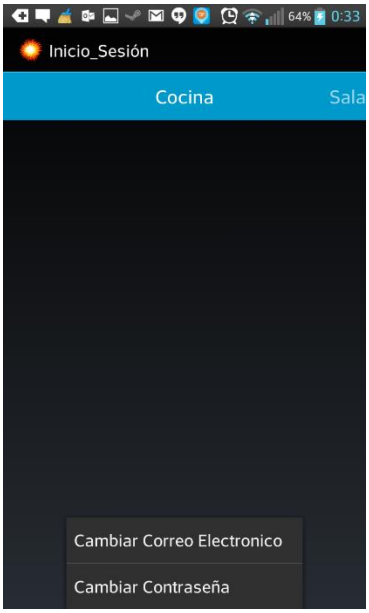


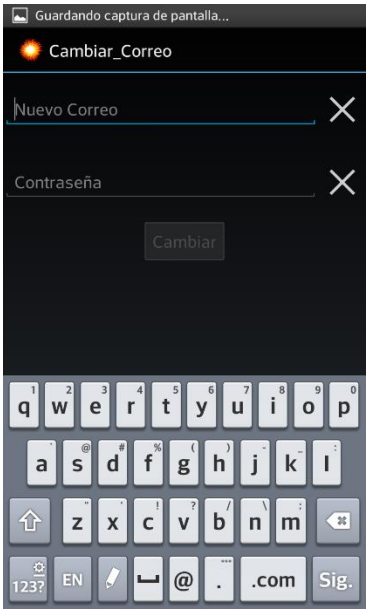

Registro
yoshio
yoshioalexis@gmail.com
Contraseña
Repetir contraseña
Código del sistema
La contraseña debe tener mas de 4 caracteres entre a-z y 0-9
Repite la contraseña anterior
El codigo de validacion se encuentra debajo del modulo central

5.1.3. MÓDULOS

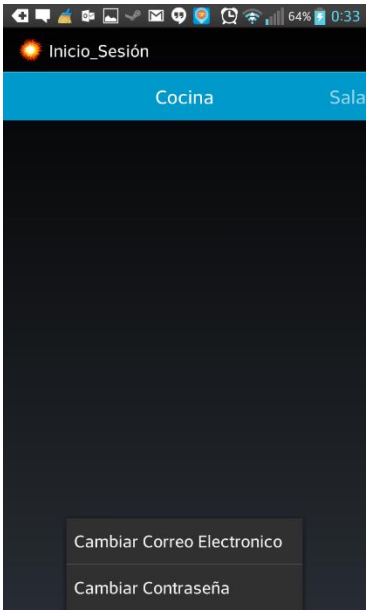

Inicio_Sesión
Cocina
Sala
Flama 0
CO 0
Temperatura 0
Gas 0
Humedad 0

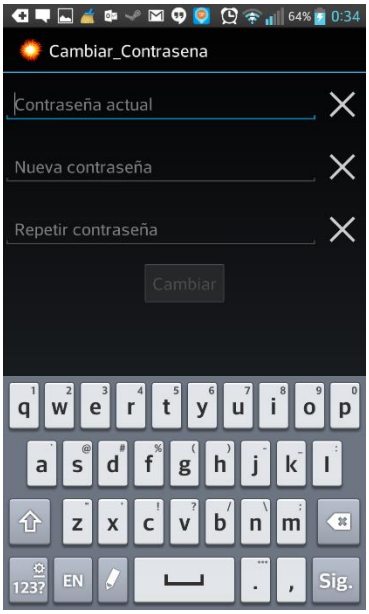
5.1.4. CAMBIO DE CORREO





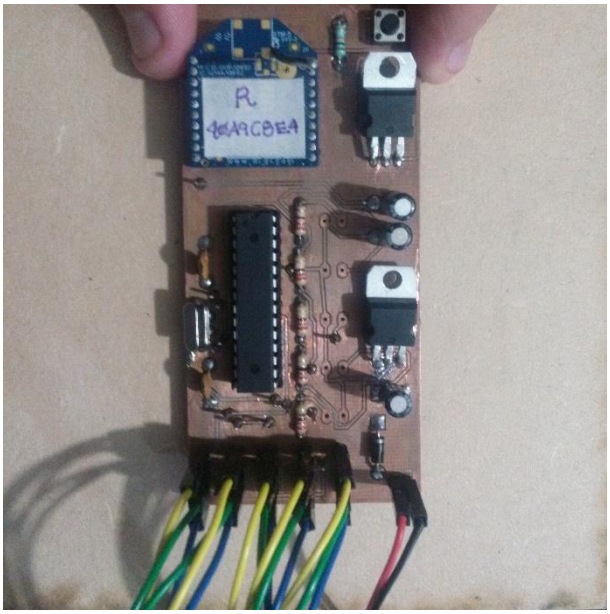
5.1.5. CAMBIO DE CONTRASEÑA





5.2. Modelo a Escala de una Casa Habitación

5.3. Módulo de Sensores



CAPÍTULO VI: PRUEBAS DEL SISTEMA

6.1. PRUEBAS BASADAS EN LOS CASOS DE USO

6.1.1. REGISTRAR USUARIO

En la Tabla 30 se muestra la prueba paso a paso de Caso de Uso “Registrar Usuario”.

Trayectoria	Paso	Respuesta	Resultado
Principal	1. El 🧍 inicia la aplicación *Domoduino* en su dispositivo móvil.	La aplicación despliega la pantalla “Inicio de Sesión”.	✓
	2. El 🧍 presiona el botón “Registrarse”.	La aplicación despliega la pantalla “Registro”.	✓
	3. El 🧍 introduce sus datos (nombre de usuario, contraseña dos veces, dirección de correo electrónico, código del sistema) y presiona el botón “Crear Usuario”.	La aplicación valida el formato de los datos introducidos y envía los datos para ser validados y registrados en el módulo central. La aplicación despliega el mensaje “Usuario Creado” y regresa a la pantalla “Inicio de Sesión”	✓
Alterna A: El dispositivo móvil pierde la conexión a Internet		El dispositivo móvil despliega el mensaje “Revise su conexión a Internet” y la aplicación se cierra.	✓
Alterna B: El módulo central no responde.		La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde" y se cierra.	✓
Alterna C: El módulo central regresa un código de error.	1. La aplicación verifica el código de error.	a. La aplicación despliega el mensaje "El usuario o correo ya ha sido registrado anteriormente".	✓
		b. La aplicación despliega el mensaje "Código de módulo inválido".	✓
	2. La aplicación borra los datos introducidos.		✓

Tabla 30. Pruebas del Caso de Uso "Registrar Usuario"

6.1.2. INICIAR SESIÓN

En la Tabla 31 se muestra la prueba paso a paso de Caso de Uso “Iniciar Sesión”.

Trayectoria	Paso	Respuesta	Resultado
Principal	1. El 🧍 inicia la aplicación *Domoduino* en su dispositivo móvil.	La aplicación despliega la pantalla “Inicio de Sesión”.	✓
	2. El 🧍 introduce su nombre de usuario y su contraseña	La aplicación valida el formato de los datos introducidos.	✓
	3. El 🧍 presiona el botón “Iniciar sesión”	La aplicación envía los datos para ser validados en el módulo central y despliega el mensaje Bienvenido si la información era correcta.	✓
Alterna A: El dispositivo móvil pierde la conexión a Internet		El dispositivo móvil despliega el mensaje “Revise su conexión a Internet” y la aplicación se cierra.	✓
Alterna B: El módulo central no responde.		La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde" y se cierra.	✓

Tabla 31. Pruebas del Caso de Uso "Iniciar Sesión"

6.1.3. CONSULTAR MÓDULO

En la Tabla 32 se muestra la prueba paso a paso de Caso de Uso “Consultar Modulo”.

Trayectoria	Paso	Respuesta	Resultado
Principal	1. El sistema se conecta al módulo central para obtener la información de los módulos con sensores y despliega la pantalla “Módulos” mostrando los módulos en forma de lista horizontal, su estado y los valores de sus respectivos sensores.	El verifica el estado de los módulos y los valores de sus sensores deslizando la pantalla hacia la izquierda.	✓
Alterna A: El módulo central no responde.		La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde" y se cierra.	✓

Tabla 32. Pruebas del Caso de Uso "Consultar Módulo"

6.1.4. CAMBIAR CORREO

En la Tabla 33 se muestra la prueba paso a paso de Caso de Uso “Cambiar Correo”.

Trayectoria	Paso	Respuesta	Resultado
Principal	1. El 🧍 presiona el botón de opciones del dispositivo móvil	La aplicación despliega un menú de opciones.	✓
.	2. El 🧍 selecciona la opción “Cambiar correo electrónico”	La aplicación despliega la pantalla “Cambio de Correo”.	✓
	3. El 🧍 introduce el nuevo correo y su contraseña actual y presiona el botón “Cambiar correo”.	La aplicación valida que el formato de los datos introducidos sea correcto y envía los datos al módulo central para ser actualizados. Una vez son actualizados, la aplicación despliega el mensaje “Cambio de correo exitoso” y despliega la pantalla “Módulos”.	✓
Alterna A: El módulo central no responde.		La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde" y se cierra.	✓
Alterna B: El módulo central regresa un código de error.	1. La aplicación verifica el código de error	a. La aplicación despliega el mensaje "El correo ya ha sido registrado anteriormente".	✓
		b. La aplicación despliega el mensaje "La contraseña es incorrecta".	✓
	2. La aplicación borra los datos introducidos		✓

Tabla 33. Pruebas del Caso de Uso "Cambiar Correo"

6.1.5. CAMBIAR CONTRASEÑA

En la Tabla 34 se muestra la prueba paso a paso de Caso de Uso “Cambiar Contraseña”.

Trayectoria	Paso	Respuesta	Resultado
Principal	1. El 🧍 presiona el botón de opciones del dispositivo móvil.	La aplicación despliega un menú de opciones.	✓
.	2. El 🧍 selecciona la opción “Cambiar contraseña”	La aplicación despliega la pantalla “Cambio de Contraseña”.	✓
	3. El 🧍 introduce su contraseña actual, la nueva contraseña 2 veces y presiona el botón “Cambio de contraseña”.	La aplicación valida que el formato de los datos introducidos sea correcto y envía los datos al módulo central para ser actualizados. Una vez son actualizados, la aplicación despliega el mensaje “Cambio de correo exitoso” y despliega la pantalla “Módulos”.	✓
Alterna A: Las contraseñas no son iguales.		La aplicación despliega el mensaje "Las contraseñas no son iguales, escríbalas nuevamente" y borra los datos de los campos de nueva contraseña.	✓
Alterna B: El módulo central no responde.		La aplicación despliega el mensaje "El módulo central no responde, inténtelo de nuevo más tarde" y se cierra.	✓
Alterna C: El módulo central regresa un código de error.	1. La aplicación verifica el código de error	La aplicación despliega el mensaje "La contraseña actual es incorrecta" y borra los datos introducidos.	✓

Tabla 34. Pruebas del Caso de Uso "Cambiar Contraseña"

6.1.6. RECUPERAR CONTRASEÑA

En la Tabla 35 se muestra la prueba paso a paso de Caso de Uso “Recuperar Contraseña”.

Trayectoria	Paso	Respuesta	Resultado
Principal	1. El 🧍 inicia la aplicación *Domoduino* en su dispositivo móvil.	La aplicación despliega la pantalla “Inicio de Sesión”.	✓
.	2. El 🧍 presiona el enlace “Olvidé mi contraseña”	La aplicación despliega la pantalla “Recuperación de Contraseña”	✓
	3. El 🧍 introduce su correo electrónico y presiona el botón enviar.	La aplicación envía el correo al módulo central para validarlo y obtener la contraseña. Una vez enviado despliega el mensaje “Se le ha enviado un correo electrónico con la contraseña” y regresa a la pantalla “Inicio de Sesión”.	✓
Alterna A: Las contraseñas no son iguales.		La aplicación despliega el mensaje "Las contraseñas no son iguales, escríbalas nuevamente" y borra los datos de los campos de nueva contraseña.	✓
Alterna A: El dispositivo móvil pierde la conexión a Internet		El dispositivo móvil despliega el mensaje “Revise su conexión a Internet” y la aplicación se cierra.	✓
Alterna C: El correo no se encuentra registrado		La aplicación despliega el mensaje "El correo proporcionado no existe, inténtalo nuevamente" y borra el campo de correo.	✓

Tabla 35. Pruebas del Caso de Uso "Recuperar Contraseña"

CAPÍTULO VII: CONCLUSIONES

Al concluir el desarrollo del sistema exitosamente se logró implementar un sistema de prevención de detección de incendios en un modelo a escala de una casa habitación. El sistema detecta fugas de gas (que se consideran un factor de riesgo para que se desarrolle un incendio), aumento de temperatura y presencia de flama junto con presencia de monóxido de carbono (que se consideran los componentes más importantes de un incendio) y contaminación por monóxido de carbono (que es un riesgo importante para la salud).

Otros aspectos importantes que surgieron durante la realización de este proyecto son:

- Modelar el diseño de un circuito: Esto debido a que los módulos de Arduino requieren un mayor espacio y un mayor cableado para poder conectar todos los componentes.
- Construcción de un modelo a escala de una habitación: esto se realizó ya que no se pueden generar incendios reales en una casa para probar la funcionalidad del sistema, por lo que se desarrolló un modelo a escala de una habitación que funge como un entorno controlado.
- Investigación de un material no inflamable para la realización del modelo a escala.

Las dificultades que fueron superadas en el desarrollo del sistema fueron:

- Al intentar implementar el sistema en una red cerrada donde no se tiene acceso a él o los routers. La consulta de información no se puede realizar ya que estos tienen que configurarse de tal manera que cuando se trate de conectar al módulo central mediante un puerto en específico este tiene que estar configurado para re direccionarlo al puerto del módulo central.
- En la construcción del modelo no se consideró un sistema de desagüe para expulsar el agua proveniente de los aspersores. Por lo tanto se tuvo que realizar el secado manualmente.
- No se realizaron pruebas de incendio con gas debido a la volatilidad del mismo.
- El tiempo de desarrollo supero los parámetros estimados en el análisis de riesgos.

Se considera que los objetivos del proyecto se cumplieron exitosamente a pesar de las dificultades, pero el proyecto sigue abierto a nuevas posibilidades.

CAPÍTULO VIII: TRABAJO A FUTURO

Como trabajo a futuro se considera lo siguiente:

- Implementar otro tipo de sensores y actuadores.
- Robustecer el algoritmo para detección y prevención de incendios.
- Implementar una fuente de alimentación recargable.
- Realizar estadísticas de la base de datos ya incorporada al sistema.
- Implementar diferentes tipos de notificaciones como mensajes de texto, llamadas a servicios de emergencia.
- Desarrollo de la aplicación para otros sistemas operativos móviles principalmente iOS, Windows Phone.
- Ampliar el enfoque de desarrollo para poder implementar otro tipo de sistemas de seguridad.

REFERENCIAS

REFERENCIAS

[1] Asesores en Emergencias y Desastres S. de R.L. de C.V., «Manual Contra Incendio,» [En línea]. Available: http://www.camafu.org.mx/index.php/ManejoEmergencias/articles/manual-contra-incendios.html. [Último acceso: 9 Abril 2014].

[2] Ninja Blocks, «NINJA SPHERE: Next Generation Control of Your Environment,» [En línea]. Available: http://www.kickstarter.com/projects/ninja/ninja-sphere-next-generation-control-of-your-envir?ref=live. [Último acceso: 22 Abril 2014].

[3] DSI Detección y Supresión Inteligente, «CHEETAH Xi - Sistema de Comunicación Digital, Peer to Peer y Bi-Direccional,» [En línea]. Available: http://www.fike.com/documents/firesupp/firesdsys/chxi/promo/brochures/B9109%20SPA%20Cheetah%20Xi.pdf. [Último acceso: 22 Abril 2014].

[4] Libelium Comunicaciones Distribuidas S.L., «Waspmote,» [En línea]. Available: http://www.libelium.com/products/waspmote/. [Último acceso: 22 Abril 2014].

[5] G. Baddewithana, G. Godigamuwa, P. Gauder, D. Hapuarachchi, U. Dampage y R. Wijesiriwardana, «Smart and automated fire and power monitoring system,» de *Industrial and Information Systems (ICIIS), 2013 8th IEEE International Conference*, Peradeniya, 2013.

[6] A. Cellatoglu y K. Balasubramanian, «Simple schemes of remote alerting by fire and intruder detection: A proposal favoring home security,» de *Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference*, Penang, 2011.

[7] K. Chang Lee y H.-H. Lee, «Network-based fire-detection system via controller area network for smart home automation,» *Consumer Electronics, IEEE Transactions*, vol. 50, nº 4, pp. 1093-2000, 2004.

[8] S. L. Rose-Pehrsson, R. E. Shaffer, S. J. Hart, F. W. Williams, D. T. Gottuk, B. D. Strehlen y S. A. Hill, «Multi-criteria fire detection systems using a probabilistic neural network,» *Sensors and Actuators*, vol. B, nº 69, pp. 325-335, 2000.

[9] Academia del Heroico Cuerpo de Bomberos, México D.F., «Folleto de Información sobre el Fuego y los Incendios,» [En línea]. Available:

http://www.bomberos.df.gob.mx/work/sites/hcb/resources/LocalContent/182/1/Elfuegoylosincendios.pdf. [Último acceso: 2014 Abril 2014].

[10] Dirección General de Protección Civil y Emergencias de Madrid, «Incendios,» [En línea]. Available: http://www.proteccioncivil.org/catalogo/carpeta02/carpeta24/vademecum12/vdm010.htm#1001. [Último acceso: 9 Abril 2014].

[11] Heroico Cuerpo de Bomberos, «Siniestros relacionados con el fuego,» Coordinación General de Modernización Administrativa - Distrito Federal, [En línea]. Available: http://www.bomberos.df.gob.mx/wb/hcb/siniestros_relacionados_con_fuego_1. [Último acceso: 9 Abril 2014].

[12] National Fire Protection Association, «Symptoms of CO poisoning,» 5 Enero 2009. [En línea]. Available: http://www.nfpa.org/safety-information/for-consumers/fire-and-safety-equipment/carbon-monoxide/symptoms-of-co-poisoning. [Último acceso: 26 Abril 2014].

[13] PEMEX Gas y Petroquímica Básica, «Hoja de datos de seguridad para sustancias químicas - Gas licuado del petróleo,» Febrero 2007. [En línea]. Available: http://www.gas.pemex.com.mx/NR/rdonlyres/D3D851A9-FDE6-4F68-8FD1-3CC6E50163E4/0/HojaSeguridadGasLP_v2007.pdf. [Último acceso: 26 Abril 2014].

[14] Instituto Nacional de Seguridad e Higiene en el Trabajo, Gobierno de España, «Incendios, Riesgos Generales,» Gobierno de España, Ministerio de Empleo y Seguridad Social, [En línea]. Available: http://www.insht.es/InshtWeb/Contenidos/Documentacion/TextosOnline/EnciclopediaOIT/tomo2/41.pdf. [Último acceso: 17 Abril 2014].

[15] Nora, «Revisión del sistema de protección contra incendios,» 19 Junio 2012. [En línea]. Available: http://unajaponesaenjapon.com/wp-content/uploads/2012/06/detector-de-incendios.jpg. [Último acceso: 22 Abril 2014].

[16] R. Pallas Areny, Adquisición y Distribución de Señales, Barcelona: Marcomo S.A. , 1993.

[17] R. Pallas Areny, Sensores y Acondicionadores de Señal, Barcelona: Marcombo S.A., 2003.

[18] J. Tomás Gironés, El Gran Libro de Android, México: Alfaomega Grupo Editor, 2012.

[19] F. Briano, «Android: La nueva plataforma de desarrollos móviles,» 15 Noviembre 2007. [En línea]. Available: http://picandocodigo.net/2007/android-la-nueva-plataforma-de-desarrollos-moviles/. [Último acceso: 22 Abril 2014].

[20] A. Mehta, «The Sweet and Tasty Android Versions History: Cupcake to KitKat,» 14 Febrero 2014. [En línea]. Available: http://techtoward.com/2014/02/14/sweet-tasty-android-versions-history-cupcake-kitkat/. [Último acceso: 22 Abril 2014].

[21] Social Compare, «Comparison of ARM based boards suitable for derivative product development,» [En línea]. Available: http://socialcompare.com/es/comparison/arm-boards. [Último acceso: 19 Abril 2014].

[22] Sunxi, «Cubieboard,» [En línea]. Available: http://linux-sunxi.org/Cubieboard. [Último acceso: 19 Abril 2014].

[23] Arduino, «What is Arduino?,» [En línea]. Available: http://www.arduino.cc/es/. [Último acceso: Marzo 2013].

[24] J. A. Carballar Falcón, Wi-Fi, Instalación, Seguridad y Aplicaciones, México: Alfaomega Grupo Editor, 2007.

[25] R. Faludi, Building Wireless Sensor Networks, Sebastopol: O'Reilly Media, Inc, 2011.

[26] JMN, «Comenzando con ZigBee,» 14 Febrero 2012. [En línea]. Available: http://webdelcire.com/wordpress/archives/1714. [Último acceso: 22 Abril 2014].

[27] MetAs & Metrólogos Asociados, «Sensores de humedad, tipos y aplicaciones,» Mayo 2008. [En línea]. Available: http://www.metas.com.mx/guiametas/la-guia-metas-08-05-sensores-de-humedad.pdf. [Último acceso: 22 Abril 2014].

[28] «Sensor de temperatura,» [En línea]. Available: http://medirtemperatura.com/sensor-temperatura.php. [Último acceso: 22 Abril 2014].

[29] Universidad Técnica Federico Santa María - Departamento de Electrónica, «Automatización Industrial: Sensores de Humedad,» 1 Junio 2001. [En línea]. Available: http://www2.elo.utfsm.cl/~elo372/complemento2.pdf. [Último acceso: 22 Abril 2014].

[30] Y. Pantoja, «Sensores de gases,» 30 Octubre 2012. [En línea]. Available: http://www.slideshare.net/Dabyus/sensores-de-gases. [Último acceso: 22 Abril 2014].

[31] Instituo Nacional de Seguridad e Higiene en el Trabajo, Ministerio de Trabajo y Asuntos Sociales España, «NTP 215: Detectores de humos,» 1988. [En línea]. Available: http://www.insht.es/InshtWeb/Contenidos/Documentacion/FichasTecnicas/NTP/Ficheros/201a300/ntp_215.pdf. [Último acceso: 22 Abril 2014].

[32] G. Bayne, «¿Cómo funciona un sensor de llama?,» [En línea]. Available: http://www.ehowenespanol.com/funciona-sensor-llama-como_146146/. [Último acceso: 22 Abril 2014].

[33] Aosong Electronics Co., «Digital-output relative humidity & temperature sensor / module,» [En línea]. Available: http://www.adafruit.com/datasheets/DHT22.pdf. [Último acceso: 22 Abril 2014].

[34] Henan Hanwei Electronics Co., «MQ-2 Semiconductor Sensor for Combustible Gas,» [En línea]. Available: http://www.pololu.com/file/0J309/MQ2.pdf. [Último acceso: 22 Abril 2014].

[35] Henan Hanwei Electronics Co., «MQ-7 Semiconductor Sensor for Carbon Monoxide,» [En línea]. Available: http://www.pololu.com/file/0J313/MQ7.pdf. [Último acceso: 22 Abril 2014].

[36] Seeed Studio, «Grove - Flame Sensor,» [En línea]. Available: http://www.seeedstudio.com/depot/Grove-Flame-Sensor-p-1450.html. [Último acceso: 22 Abril 2014].

[37] I. Sommerville, Ingeniería del Software, Madrid: Pearson Educación, 2005.

[38] F. L. Osorio Rivera, Bases de datos relacionales, Teoría y práctica, Medellín: Instituto Tecnológico Metropolitano, 2008.

[39] W3C España, «Guía Breve de Servicios Web,» [En línea]. Available: http://w3c.es/Divulgacion/GuiasBreves/ServiciosWeb. [Último acceso: 22 Abril 2014].

[40] S. Dhingra, «REST vs. SOAP: How to choose the best Web service,» SeachSOA, 8 Abril 2013. [En línea]. Available: http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service. [Último acceso: 22 Abril 2014].

[41] T. Biske, «REST or SOAP: Which offers the most benefits for mobile applications,» SearchSOA, 23 Abril 2013. [En línea]. Available: http://searchsoa.techtarget.com/answer/REST-or-SOAP-Which-offers-the-most-benefits-for-mobile-applications. [Último acceso: 22 Abril 2014].

[42] S. Lang, «Multi-criteria Fire Detection,» Febrero 2011. [En línea]. Available: http://osfm.fire.ca.gov/firelifesafety/pdf/Smoke%20Alarm%20Task%20Force/Smoke%20Alarm%20Presentations/Multi-criteria%20Fire%20Detection.pdf. [Último acceso: 2014 Abril 2014].

[43] J. Y. Wong, «Multi-Criteria Fire Alarm (MCFA),» 5 Marzo 2012. [En línea]. Available: http://www.nfpa.org/~/media/Files/Research/Research%20Foundation/foundation%20proceedings/2012%20SUPDET/6Wong%20presentation.ashx. [Último acceso: 22 Abril 2014].

ANEXOS

BIBLIOTECA PARA EL MANEJO DEL SENSOR MQ2

```
/*
  MQ2.h - Biblioteca para obtener datos del sensor de Humo, Gas LPG Y CO.
  Creado por Yoshio Alexis Martinez Lopez.
*/

#ifndef MQ2_H
#define MQ2_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
#include "Math.h"
#define _S 1
#define _LPG 2
#define _CO 3

class MQ2 {
      public:
            MQ2(int pin, float R_L = 5, float RSROA = 9.83f, int
calibrationTime = 1);
            void begin(void);
            void calibrate(void);
            float getSmoke(void);
            float getLPG(void);
            float getCO(void);
            void coherence(int);
            bool works(int);
            bool inRange(int);
            bool vary(int);
            bool change(int);



      private:
            int _pin;
            //const int _S,_LPG,_CO;
            long      _calibrationTime,      _heatTime,      _sampleTimes,
_sampleInterval, _times, _interval;
            float _S_a, _S_b, _S_c;//Constantes a,b,c de la ecuacion de
Humo
            float _LPG_a, _LPG_b, _LPG_c;//Constantes a,b,c de la ecuacion
LPG
            float _CO_a, _CO_b, _CO_c;//Constantes a,b,c de la ecuacion de
CO
            float *_a, *_b, *_c;
            float _LS, _LLPG, _LCO;//Penultimos valores tomados
            float _NS, _NLPG, _NCO;//Ultimos valores tomados
```

```
            float *_N, *_L;

            float _sensMinS, _sensMinLPG, _sensMinCO;//Minimos del sensor
            float _sensMaxS, _sensMaxLPG, _sensMaxCO;//Maximos del sensor
            float _envMinS, _envMinLPG, _envMinCO;//Minimos del entorno
            float _envMaxS, _envMaxLPG, _envMaxCO;//Maximos del entorno
            float *_sensMin, *_sensMax, *_envMin, *_envMax;
            float  _varS,  _varLPG,  _varCO;//Variaciones  maximas  de  los
sensores
            float *_var;

            bool _workS, _workLPG, _workCO;
            bool _rangeS, _rangeLPG, _rangeCO;
            bool _varyS, _varyLPG, _varyCO;
            bool _changeS, _changeLPG, _changeCO;
            bool *_work, *_range, *_vary, *_change;

            float _lastRead;
            float _RL, _RS, _RO, _RSROA, _RSRO, _PPM, _VO, _VCC;
            void calibrationTimes(int);
            float calcVO(int, int);//Calcula el voltaje de entrada
            float calcRO(float);//Calcula Ro
            float  calcRS(float);//Calcula  la  resistencia  variable  del
sensor
            float calcRSRO(float, float);//Calcula la division entre RS y
RO
            float calcPPM(float, int);//Calcula las partes por millon de
un tipo de gas
            void selectVariable(int);//Selecciona la variable tratar


};
#endif

/*
  MQ2.cpp - Biblioteca para obtener datos del sensor de Humo, Gas LPG y CO.
  Creado por Yoshio Alexis Martinez Lopez.
*/

#include "MQ2.h"

float VO,RS,RO,RL,RSRO,PPM;

MQ2::MQ2(int pin, float R_L, float RSROA, int calibrationTime) {
      _pin = pin;
       // _S = 1;
       // _LPG = 2;
       // _CO = 3;
      _RL = R_L;
      _RSROA = RSROA;
```

```
	_VCC = 5;
	_S_a = 18.73f;
	_S_b = -0.2855f;
	_S_c = -.7243f;
	_LPG_a = 21.52f;
	_LPG_b = -0.4766f;
	_LPG_c = -0.007802f;
	_CO_a = 0;
	_CO_b = 0;
	_CO_c = 0;


	_sensMinS = 0;
	_sensMinLPG = 0;
	_sensMinCO = 0;
	_sensMaxS = 10000;
	_sensMaxLPG = 10000;
	_sensMaxCO = 10000;
	_envMinS = 0;
	_envMinLPG = 0;
	_envMinCO = 0;
	_envMaxS = 2;
	_envMaxLPG = 2;
	_envMaxCO = 1.3;

	_varS = 10000;
	_varLPG = 10000;
	_varCO = 10000;


	calibrationTimes(calibrationTime);
}

void MQ2::begin(void) {
	// Inicializar los pines a utilizar!
	pinMode(_pin, INPUT);
	//_lastRead = getSmoke();
}

void MQ2::calibrationTimes(int calibrationTime){
	_calibrationTime = calibrationTime * 60000;
	_heatTime = _calibrationTime/2;
	_sampleInterval = 500;
	_sampleTimes = _heatTime/_sampleInterval;
	_times = 5;
	_interval = 50;
}

void MQ2::calibrate(void) {
	//Serial.println("calibrando");
```

```
	//Serial.println(_heatTime);
	delay(_heatTime);
	_VO = calcVO(_sampleTimes,_sampleInterval);
	_RO = calcRO(_VO);
	_NS = getSmoke();
	_NLPG = getLPG();
	_NCO = getCO();
}

float MQ2::calcVO (int times,int interval){
	float VOS = 0;
	for(int i = 0;i < times; i++){
		VOS += (float)analogRead(_pin)/1023*_VCC;
		delay(interval);
	}
	VO = VOS/times;
	return VO;
}

float MQ2::calcRSRO(float R_S, float R_O){
	RSRO = R_S/R_O;
	return RSRO;
}

float MQ2::calcRO(float V_O){
	_RS = calcRS(V_O);
	RO = _RS/_RSROA;
	return RO;
}

float MQ2::calcRS(float V_O){
	RS = _RL*((_VCC/V_O)-1);
	return RS;
}

float MQ2::calcPPM(float RS_RO,int type){
	switch(type){
	case _S:
		_a = &_S_a;
		_b = &_S_b;
		_c = &_S_c;
		break;
	case _LPG:
		_a = &_LPG_a;
		_b = &_LPG_b;
		_c = &_LPG_c;
		break;
	case _CO:
		_a = &_CO_a;
		_b = &_CO_b;
```

```
            _c = &_CO_c;
            break;
      }
      PPM = pow(((RS_RO - *_c) / *_a),(1 / *_b));
      return PPM;
}

float MQ2::getSmoke(void){
      _VO = calcVO(_times,_interval);
      _RS = calcRS(_VO);
      _RSRO = calcRSRO(_RS,_RO);
      _PPM = calcPPM(_RSRO,_S);
      _LS = _NS;
      _NS = float(int(_PPM));
      return _NS;
}

float MQ2::getLPG(void){
      _VO = calcVO(_times,_interval);
      _RS = calcRS(_VO);
      _RSRO = calcRSRO(_RS,_RO);
      _PPM = calcPPM(_RSRO,_LPG);
      _LLPG = _NLPG;
      _NLPG = float(int(_PPM));
      return _NLPG;
}

float MQ2::getCO(void){
      _VO = calcVO(_times,_interval);
      _RS = calcRS(_VO);
      _RSRO = calcRSRO(_RS,_RO);
      _PPM = calcPPM(_RSRO,_CO);
      _LCO = _NCO;
      _NCO = float(int(_PPM));
      return _NCO;
}

void MQ2::selectVariable(int type){
      switch(type){
      case _S:
            _sensMin = &_sensMinS;
            _sensMax = &_sensMaxS;
            _envMin = &_envMinS;
            _envMax = &_envMaxS;
            _var = &_varS;
            _N = &_NS;
            _L = &_LS;
            _work = &_workS;
            _range = &_rangeS;
            _vary = &_varyS;
```

```
            _change = &_changeS;
            break;
      case _LPG:
            _sensMin = &_sensMinLPG;
            _sensMax = &_sensMaxLPG;
            _envMin = &_envMinLPG;
            _envMax = &_envMaxLPG;
            _var = &_varLPG;
            _N = &_NLPG;
            _L = &_LLPG;
            _work = &_workLPG;
            _range = &_rangeLPG;
            _vary = &_varyLPG;
            _change = &_changeLPG;
            break;
      case _CO:
            _sensMin = &_sensMinCO;
            _sensMax = &_sensMaxCO;
            _envMin = &_envMinCO;
            _envMax = &_envMaxCO;
            _var = &_varCO;
            _N = &_NCO;
            _L = &_LCO;
            _work = &_workCO;
            _range = &_rangeCO;
            _vary = &_varyCO;
            _change = &_changeCO;
            break;
      }
}
bool MQ2::works(int type){
      selectVariable(type);
      if(*_N <= *_sensMax && *_N >= *_sensMin){
            *_work = true;
      }
      else{
            *_work = false;
      }
      return *_work;
}

bool MQ2::inRange(int type){
      selectVariable(type);
      if(*_N <= *_envMax && *_N >= *_envMin){
            *_range = true;
      }
      else{
            *_range = false;
      }
      return *_range;
```

```
}

bool MQ2::vary(int type){
	selectVariable(type);
	if(abs(*_N - *_L) <= *_var){
		*_vary = true;
	}
	else{
		*_vary = false;
	}
	return *_vary;
}

bool MQ2::change(int type){
	selectVariable(type);
	if((*_N - *_L) != 0.0){
		*_change = true;
	}
	else{
		*_change = false;
	}
	return *_change;
}

void MQ2::coherence(int type){
	selectVariable(type);
	if(*_N <= *_sensMax && *_N >= *_sensMin){
		*_work = true;
		if(*_N <= *_envMax && *_N >= *_envMin){
			*_range = true;
			if(abs(*_N - *_L) <= *_var){
				*_vary = true;
			}
			else{
				*_vary = false;
			}
		}
		else{
			*_range = false;
			if(abs(*_N - *_L) <= *_var){
				*_vary = true;
			}
			else{
				*_vary = false;
			}
		}
	}
	else{
		*_work = false;
		if(abs(*_N - *_L) <= *_var){
```

```
                  *_vary = true;
            }
            else{
                  *_vary = false;
            }
      }
}
```

BIBLIOTECA PARA EL MANEJO DEL SENSOR MQ7

```
/*
  MQ7.h - Biblioteca para obtener datos del sensor de CH4, Gas LPG Y CO.
  Creado por Yoshio Alexis Martinez Lopez.
*/

#ifndef MQ7_H
#define MQ7_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
#include "Math.h"
#define _CH4 1
#define _LPG 2
#define _CO 3

class MQ7 {
      public:
            MQ7(int pin, float R_L = 10, float RSROA = 11.64f, int
calibrationTime = 1);
            void begin(void);
            void calibrate(void);
            float getCH4(void);
            float getLPG(void);
            float getCO(void);
            void coherence(int);
            bool works(int);
            bool inRange(int);
            bool vary(int);
            bool change(int);



      private:
            int _pin;
            //const int _S,_LPG,_CO;
            int      _calibrationTime,      _heatTime,      _sampleTimes,
_sampleInterval, _times, _interval;
```

```
            float _CH4_a, _CH4_b, _CH4_c;//Constantes a,b,c de la ecuacion
de CH4
            float _LPG_a, _LPG_b, _LPG_c;//Constantes a,b,c de la ecuacion
de LPG
            float _CO_a, _CO_b, _CO_c;//Constantes a,b,c de la ecuacion de
CO
            float *_a, *_b, *_c;
            float _LCH4, _LLPG, _LCO;//Penultimos valores tomados
            float _NCH4, _NLPG, _NCO;//Ultimos valores tomados
            float *_N, *_L;

            float  _sensMinCH4,  _sensMinLPG,  _sensMinCO;//Minimos  del
sensor
            float  _sensMaxCH4,  _sensMaxLPG,  _sensMaxCO;//Maximos  del
sensor
            float _envMinCH4, _envMinLPG, _envMinCO;//Minimos del entorno
            float _envMaxCH4, _envMaxLPG, _envMaxCO;//Maximos del entorno
            float *_sensMin, *_sensMax, *_envMin, *_envMax;
            float _varCH4, _varLPG, _varCO;//Variaciones maximas de los
sensores
            float *_var;

            bool _workCH4, _workLPG, _workCO;
            bool _rangeCH4, _rangeLPG, _rangeCO;
            bool _varyCH4, _varyLPG, _varyCO;
            bool _changeCH4, _changeLPG, _changeCO;
            bool *_work, *_range, *_vary, *_change;

            float _lastRead;
            float _RL, _RS, _RO, _RSROA, _RSRO, _PPM, _VO, _VCC;
            void calibrationTimes(int);
            float calcVO(int, int);//Calcula el voltaje de entrada
            float calcRO(float);//Calcula Ro
            float  calcRS(float);//Calcula  la  resistencia  variable  del
sensor
            float calcRSRO(float, float);//Calcula la division entre RS y
RO
            float calcPPM(float, int);//Calcula las partes por millon de
un tipo de gas
            void selectVariable(int);//Selecciona la variable tratar


};
#endif

/*
  MQ7.cpp - Biblioteca para obtener datos del sensor de CH4, Gas LPG Y CO.
  Creado por Yoshio Alexis Martinez Lopez.
*/
```

```
#include "MQ7.h"

float MQ7VO,MQ7RS,MQ7RO,MQ7RL,MQ7RSRO,MQ7PPM;

MQ7::MQ7(int pin, float R_L, float RSROA, int calibrationTime) {
	_pin = pin;
	// _CH4 = 1;
	// _LPG = 2;
	// _CO = 3;
	_RL = R_L;
	_RSROA = RSROA;
	_VCC = 5;
	_CH4_a = 0.0f;
	_CH4_b = 0.0f;
	_CH4_c = 0.0f;
	_LPG_a = 0.0f;
	_LPG_b = 0.0f;
	_LPG_c = 0.0f;
	_CO_a = 30.34f;
	_CO_b = -0.7459f;
	_CO_c = 0.04024f;

	_sensMinCH4 = 50;
	_sensMinLPG = 50;
	_sensMinCO = 0;
	_sensMaxCH4 = 4000;
	_sensMaxLPG = 4000;
	_sensMaxCO = 4000;
	_envMinCH4 = 0;
	_envMinLPG = 0;
	_envMinCO = 0;
	_envMaxCH4 = 2;
	_envMaxLPG = 2;
	_envMaxCO = 1.3;
	_varyCH4 = 4000;
	_varyLPG = 4000;
	_varyCO = 4000;

	_varCH4 = 4000;
	_varLPG = 4000;
	_varCO = 4000;


	calibrationTimes(calibrationTime);
}

void MQ7::begin(void) {
	// Inicializar los pines a utilizar!
	pinMode(_pin, INPUT);
	//_lastRead = getSmoke();
```

```
}

void MQ7::calibrationTimes(int calibrationTime){
      _calibrationTime = calibrationTime * 1000;
      _heatTime = _calibrationTime/2;
      _sampleInterval = 500;
      _sampleTimes = _heatTime/_sampleInterval;
      _times = 5;
      _interval = 50;
}

void MQ7::calibrate(void) {
      delay(_heatTime);
      _VO = calcVO(_sampleTimes,_sampleInterval);
      _RO = calcRO(_VO);
      _NCH4 = getCH4();
      _NLPG = getLPG();
      _NCO = getCO();
}

float MQ7::calcVO (int times,int interval){
      float VOS = 0;
      for(int i=0;i < times; i++){
            VOS += (float)analogRead(_pin)/1023*_VCC;
            delay(interval);
      }
      MQ7VO = VOS/times;
      return MQ7VO;
}

float MQ7::calcRSRO(float R_S, float R_O){
      MQ7RSRO = R_S/R_O;
      return MQ7RSRO;
}

float MQ7::calcRO(float V_O){
      _RS = calcRS(V_O);
      MQ7RO = _RS/_RSROA;
      return MQ7RO;
}

float MQ7::calcRS(float V_O){
      MQ7RS = _RL*((_VCC/V_O)-1);
      return MQ7RS;
}

float MQ7::calcPPM(float RS_RO,int type){
      switch(type){
      case _CH4:
            _a = &_CH4_a;
```

```
            _b = &_CH4_b;
            _c = &_CH4_c;
            break;
      case _LPG:
            _a = &_LPG_a;
            _b = &_LPG_b;
            _c = &_LPG_c;
            break;
      case _CO:
            _a = &_CO_a;
            _b = &_CO_b;
            _c = &_CO_c;
            break;
      }
      MQ7PPM = pow(((RS_RO - *_c) / *_a),(1 / *_b));
      return MQ7PPM;
}

float MQ7::getCH4(void){
      _VO = calcVO(_times,_interval);
      _RS = calcRS(_VO);
      _RSRO = calcRSRO(_RS,_RO);
      _PPM = calcPPM(_RSRO,_CH4);
      _LCH4 = _NCH4;
      _NCH4 = float(int(_PPM));
      return _NCH4;
}

float MQ7::getLPG(void){
      _VO = calcVO(_times,_interval);
      _RS = calcRS(_VO);
      _RSRO = calcRSRO(_RS,_RO);
      _PPM = calcPPM(_RSRO,_LPG);
      _LLPG = _NLPG;
      _NLPG = float(int(_PPM));
      return _NLPG;
}

float MQ7::getCO(void){
      _VO = calcVO(_times,_interval);
      _RS = calcRS(_VO);
      _RSRO = calcRSRO(_RS,_RO);
      _PPM = calcPPM(_RSRO,_CO);
      _LCO = _NCO;
      _NCO = float(int(_PPM));
      return _NCO;
}

void MQ7::selectVariable(int type){
      switch(type){
```

```
        case _CH4:
              _sensMin = &_sensMinCH4;
              _sensMax = &_sensMaxCH4;
              _envMin = &_envMinCH4;
              _envMax = &_envMaxCH4;
              _var = &_varCH4;
              _N = &_NCH4;
              _L = &_LCH4;
              _work = &_workCH4;
              _range = &_rangeCH4;
              _vary = &_varyCH4;
              _change = &_changeCH4;
              break;
        case _LPG:
              _sensMin = &_sensMinLPG;
              _sensMax = &_sensMaxLPG;
              _envMin = &_envMinLPG;
              _envMax = &_envMaxLPG;
              _var = &_varLPG;
              _N = &_NLPG;
              _L = &_LLPG;
              _work = &_workLPG;
              _range = &_rangeLPG;
              _vary = &_varyLPG;
              _change = &_changeLPG;
              break;
        case _CO:
              _sensMin = &_sensMinCO;
              _sensMax = &_sensMaxCO;
              _envMin = &_envMinCO;
              _envMax = &_envMaxCO;
              _var = &_varCO;
              _N = &_NCO;
              _L = &_LCO;
              _work = &_workCO;
              _range = &_rangeCO;
              _vary = &_varyCO;
              _change = &_changeCO;
              break;
        }
}
bool MQ7::works(int type){
        selectVariable(type);
        if(*_N <= *_sensMax && *_N >= *_sensMin){
              *_work = true;
        }
        else{
              *_work = false;
        }
        return *_work;
```

```
}

bool MQ7::inRange(int type){
	selectVariable(type);
	if(*_N <= *_envMax && *_N >= *_envMin){
		*_range = true;
	}
	else{
		*_range = false;
	}
	return *_range;
}

bool MQ7::vary(int type){
	selectVariable(type);
	if(abs(*_N - *_L) <= *_var){
		*_vary = true;
	}
	else{
		*_vary = false;
	}
	return *_vary;
}

bool MQ7::change(int type){
	selectVariable(type);
	if((*_N - *_L) != 0.0){
		*_change = true;
	}
	else{
		*_change = false;
	}
	return *_change;
}

void MQ7::coherence(int type){
	selectVariable(type);
	if(*_N <= *_sensMax && *_N >= *_sensMin){
		*_work = true;
		if(*_N <= *_envMax && *_N >= *_envMin){
			*_range = true;
			if(abs(*_N - *_L) <= *_var){
				*_vary = true;
			}
			else{
				*_vary = false;
			}
		}
		else{
			*_range = false;
```

```
                  if(abs(*_N - *_L) <= *_var){
                        *_vary = true;
                  }
                  else{
                        *_vary = false;
                  }
            }
      }
      else{
            *_work = false;
            if(abs(*_N - *_L) <= *_var){
                  *_vary = true;
            }
            else{
                  *_vary = false;
            }
      }
}
```

BIBLIOTECA PARA EL MANEJO DEL SENSOR DHT22

```
/*
  DHT22.h - Biblioteca para manipular el sensor de temperatura y humedad.
  Creado por Yoshio Alexis Martinez Lopez.
*/

#ifndef DHT22_H
#define DHT22_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
// how many timing transitions we need to keep track of. 2 * number bits +
extra
#define MAXTIMINGS 85
//#define DHT22 22
#define _T 4
#define _H 5

class DHT22 {
      public:
            DHT22(uint8_t pin, uint8_t count=6);
            void begin(void);
            void calibrate(void);
            float getTemperature(void);
            float getHumidity(void);
            void coherence(int);
            bool works(int);
            bool inRange(int);
```

```
            bool vary(int);
            bool change(int);

      private:
            uint8_t data[6];
            uint8_t _pin, _count;
            boolean read(void);
            unsigned long _lastreadtime;
            boolean firstreading;

            double _LT, _LH;//Penultimos valores tomados
            double _NT, _NH;//Ultimos valores tomados
            double *_N, *_L;

            float _sensMinT, _sensMinH;//Minimos del sensor
            float _sensMaxT, _sensMaxH;//Maximos del sensor
            float _envMinT, _envMinH;//Minimos del entorno
            float _envMaxT, _envMaxH;//Maximos del entorno
            float *_sensMin, *_sensMax, *_envMin, *_envMax;
            float _varT, _varH;//Variaciones maximas de los sensores
            float *_var;

            bool _workT, _workH;
            bool _rangeT, _rangeH;
            bool _varyT, _varyH;
            bool _changeT, _changeH;
            bool *_work, *_range, *_vary, *_change;

            void selectVariable(int);//Selecciona la variable tratar
};
#endif

/*
  DHT22.cpp - Biblioteca para manipular el sensor de temperatura y humedad.
  Creado por Yoshio Alexis Martinez Lopez.
*/

#include "DHT22.h"

DHT22::DHT22(uint8_t pin, uint8_t count) {
      _pin = pin;
      _count = count;
      firstreading = true;

      _sensMinT = -40;
      _sensMinH = 0;
      _sensMaxT = 200;
      _sensMaxH = 100;
      _envMinT = 0;
      _envMinH = 0;
```

```
	_envMaxT = 40;
	_envMaxH = 65;
	_varT = 1.0;
	_varH = 100.0;

}

void DHT22::begin(void) {
	// Inicializar los pines a utilizar!
	pinMode(_pin, INPUT);
	digitalWrite(_pin, HIGH);
	_lastreadtime = 0;
}

void DHT22::calibrate(void) {
	_NT = getTemperature();
	_NH = getHumidity();
}

float DHT22::getTemperature(void) {
	float f;
	if (read()) {
		f = data[2] & 0x7F;
		f *= 256;
		f += data[3];
		f /= 10;
		if (data[2] & 0x80){
			f *= -1;
		}
		_LT = _NT;
		_NT = float(int(f));
		return _NT;
	}
	Serial.print("Lectura de temperatura fallida");
	return NAN;
}

float DHT22::getHumidity(void) {
	float f;
	if (read()) {
		f = data[0];
		f *= 256;
		f += data[1];
		f /= 10;
		_LH = _NH;
		_NH = float(int(f));
		return _NH;
	}
	Serial.print("Lectura de humedad fallida");
	return NAN;
```

```
}

boolean DHT22::read(void) {
      uint8_t laststate = HIGH;
      uint8_t counter = 0;
      uint8_t j = 0, i;
      unsigned long currenttime;

      // Poner el pin en HIGH por 250 milisegundos
      digitalWrite(_pin, HIGH);
      delay(250);

      currenttime = millis();
      if (currenttime < _lastreadtime) {
            // Hubo un vuelco
            _lastreadtime = 0;
      }

      if (!firstreading && ((currenttime - _lastreadtime) < 2000)) {
            return true; // Regresa la ultima lectura correcta
      }

      firstreading = false;
      _lastreadtime = millis();
      data[0] = data[1] = data[2] = data[3] = data[4] = 0;

      // Ponemos el pin en LOW por 20 milisegundos
      pinMode(_pin, OUTPUT);
      digitalWrite(_pin, LOW);
      delay(20);
      cli();//Deshabilito interrupciones globales
      digitalWrite(_pin, HIGH);
      delayMicroseconds(40);
      pinMode(_pin, INPUT);

      // Leer por lapsos de tiempo
      for ( i=0; i< MAXTIMINGS; i++) {
            counter = 0;
            while (digitalRead(_pin) == laststate) {
                  counter++;
                  delayMicroseconds(1);
                  if (counter == 255)     break;
            }
            laststate = digitalRead(_pin);

            if (counter == 255) break;

            // Ignorar las primeras 3 transiciones
            if ((i >= 4) && (i%2 == 0)) {
                  // Correr cada bit en el arreglo de bytes
```

```
                  data[j/8] <<= 1;
                  if (counter > _count)
                        data[j/8] |= 1;
                  j++;
            }

      }
      sei();//Habilito interrupciones globales

      // Checamos que se hallan leido los 40 bits y que la suma de
comprobacion coincide
      if ((j >= 40) &&
            (data[4] == ((data[0] + data[1] + data[2] + data[3]) & 0xFF))
) {
            return true;
      }
      return false;
}

void DHT22::selectVariable(int type){
      switch(type){
      case _T:
            _sensMin = &_sensMinT;
            _sensMax = &_sensMaxT;
            _envMin = &_envMinT;
            _envMax = &_envMaxT;
            _var = &_varT;
            _N = &_NT;
            _L = &_LT;
            _work = &_workT;
            _range = &_rangeT;
            _vary = &_varyT;
            _change = &_changeT;
            break;
      case _H:
            _sensMin = &_sensMinH;
            _sensMax = &_sensMaxH;
            _envMin = &_envMinH;
            _envMax = &_envMaxH;
            _var = &_varH;
            _N = &_NH;
            _L = &_LH;
            _work = &_workH;
            _range = &_rangeH;
            _vary = &_varyH;
            _change = &_changeH;
            break;
      }
}
bool DHT22::works(int type){
```

```
	selectVariable(type);
	if(*_N <= *_sensMax && *_N >= *_sensMin){
		*_work = true;
	}
	else{
		*_work = false;
	}
	return *_work;
}

bool DHT22::inRange(int type){
	selectVariable(type);
	if(*_N <= *_envMax && *_N >= *_envMin){
		*_range = true;
	}
	else{
		*_range = false;
	}
	return *_range;
}

bool DHT22::vary(int type){
	selectVariable(type);
	if((*_N - *_L) >= *_var){
		*_vary = true;
	}
	else{
		*_vary = false;
	}
	return *_vary;
}

bool DHT22::change(int type){
	selectVariable(type);
	if((*_N - *_L) != 0.0){
		*_change = true;
	}
	else{
		*_change = false;
	}
	return *_change;
}

void DHT22::coherence(int type){
	selectVariable(type);
	if(*_N <= *_sensMax && *_N >= *_sensMin){
		*_work = true;
		if(*_N <= *_envMax && *_N >= *_envMin){
			*_range = true;
			if(abs(*_N - *_L) <= *_var){
```

```
                        *_vary = true;
                  }
                  else{
                        *_vary = false;
                  }
            }
            else{
                  *_range = false;
                  if(abs(*_N - *_L) <= *_var){
                        *_vary = true;
                  }
                  else{
                        *_vary = false;
                  }
            }
      }
      else{
            *_work = false;
            if(abs(*_N - *_L) <= *_var){
                  *_vary = true;
            }
            else{
                  *_vary = false;
            }
      }
}
```

BIBLIOTECA PARA EL MANEJO DEL SENSOR DE FLAMA

```
/*
  FLAME.h - Biblioteca para obtener datos del sensor de Flama.
  Creado por Yoshio Alexis Martinez Lopez.
*/

#ifndef FLAME_H
#define FLAME_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

class FLAME {
      public:
            FLAME(int pin);
            void begin(void);
            void calibrate(void);
            float getFlame(void);
            bool change(void);
```

```
      private:
            bool isFlameDetected();
            int _pin;
            float _NF,_LF;
            bool _changeF;
};
#endif

/*
  FLAME.cpp - Biblioteca para obtener datos del sensor de Flama.
  Creado por Yoshio Alexis Martinez Lopez.
*/

#include "FLAME.h"

FLAME::FLAME(int pin) {
      _pin = pin;
}

void FLAME::begin(void) {
      // Inicializar los pines a utilizar!
      pinMode(_pin, INPUT);
}

void FLAME::calibrate(void) {
      _NF = getFlame();
}

float FLAME::getFlame(void){
      if(isFlameDetected()){
            _LF = _NF;
            _NF = 1.0f;
            return _NF;
      }
      else{
            _LF = _NF;
            _NF = 0.0f;
            return _NF;
      }
}

bool FLAME::change(){
      if(_NF != _LF){
            _changeF = true;
      }
      else{
            _changeF = false;
      }
      return _changeF;
}
```

```
bool FLAME::isFlameDetected(){
      if(digitalRead(_pin)){
            return false;
      }
      else {
            return true;
      }
}
```

BIBLIOTECA PARA EL ALGORITMO DE DETECCIÓN

```
/*
  DANGERS.h - Biblioteca para procesar los datos de los sensores y obtner
codigos de peligro o advertencia
  Creado por Yoshio Alexis Martinez Lopez.
  Released into the public domain.
*/

#ifndef DANGERS_H
#define DANGERS_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
#include "Math.h"

class DANGERS {
      public:
            DANGERS(int pin, int pinrele);
            int checkDanger1(float,float,float,float,float);
            int checkDanger2(bool);

      private:
            int _pin,_pinrele;
            float _g, _co, _h, _t, _f;
            int checarTiempo(bool);

};
#endif

/*
  DANGERS.cpp - Biblioteca para procesar los datos de los sensores y obtner
codigos de peligro o advertencia
  Creado por Yoshio Alexis Martinez Lopez.
  Released into the public domain.
*/
```

```
#include "DANGERS.h"

int estado1 = 1;
int estado2 = 1;

DANGERS::DANGERS(int pin, int pinrele) {
      _pin = pin;
      _pinrele = pinrele;
}

int DANGERS::checkDanger1(float g,float co,float t,float h,float f){
      _g = g;
      _co = co;
      _h = h;
      _t = t;
      _f = f;
////////////////////// NORMAL ////////////////////
      if(_co <= 100 && _g <= 1000 && _t <= 40){
        estado1 = 1;
      }
////////////////////// EXCESO CO ////////////////////
      if(_co > 100 && _co <= 200 && _g <= 1000){
        estado1 = 2;
      }
////////////////////// EXCESO TEMP ////////////////////
      if(_co <= 100 && _g <= 1000 && _t > 40){
        estado1 = 3;
      }
////////////////////// EXCESO GAS ////////////////////
      if(_g > 1000){
        estado1 = 4;
      }
////////////////////// INCENDIO ////////////////////
      if(_co > 200){
        estado1 = 5;
      }
      return estado1;
}
int DANGERS::checkDanger2(bool var){
////////////////////// NORMAL ////////////////////
      if( estado1 == 1){
            //digitalWrite(_pinrele, LOW);
            estado2 = 1;
      }
////////////////////// EXCESO CO ////////////////////
      if(estado1 == 2){
            //digitalWrite(_pinrele, LOW);
            estado2 = 1;
            if(_t > 40){
                  estado2 = checarTiempo(var);
```

```
            }
            else if(_f == 1.0){
                  estado2 = 3;
            }
      }
////////////////////// EXCESO TEMP ////////////////////
      if(estado1 == 3){
            //digitalWrite(_pinrele, LOW);
            estado2 = 1;
            if(_f == 1.0){bitClear
                  estado2 = 3;
            }
      }
////////////////////// EXCESO GAS ////////////////////
      if(estado1 == 4){
            //digitalWrite(_pinrele, LOW);
            estado2 = 1;
            if(_t > 40){
                  estado2 = checarTiempo(var);
            }
            else if(_co > 200 && _co <= 400){
                  estado2 = 3;
            }
            else if(_f == 1.0){
                  estado2 = 2;
            }
      }
////////////////////// INCENDIO ////////////////////
      if(estado1 == 5){
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, HIGH);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, LOW);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, HIGH);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, LOW);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, HIGH);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, LOW);
            //digitalWrite(_pinrele, HIGH);
      }
      return estado2;
}
```

```
////////////////////// checarTiempo ////////////////////
int DANGERS::checarTiempo(bool var){
      if(var){
            return 3;
      }
      else {
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, HIGH);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, LOW);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, HIGH);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, LOW);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, HIGH);
            delay(200);
            //digitalWrite(_pinrele, HIGH);
            digitalWrite(_pin, LOW);
            //digitalWrite(_pinrele, HIGH);
            return 4;
      }
}
```

BIBLIOTECA PARA EL MANEJO DE TRAMAS EN XBEE

```
/*
  FRAMES.h - Biblioteca para procesar las tramas de XBee
  Creado por Yoshio Alexis Martinez Lopez.
  Released into the public domain.
*/

#ifndef FRAMES_H
#define FRAMES_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
#include "Math.h"

class FRAMES {
      public:
            FRAMES(int pin);
            int totalWork(bool*,int);
```

```
            int sizeFrame();
            uint8_t*
buildFrameServer(int*,int,float*,int,bool*,int,bool*,int,bool*,int,bool*,
int,int*,int);

      private:
            int _pin;
            int _totalWork,_totalNoWork,_sizeFrame;
            bool _wg, _wco, _wt, _wh, _wf;
            byte _header;
            int checarTiempo(bool);

};
#endif

/*
  FRAMES.cpp - Biblioteca para procesar las tramas de XBee
  Creado por Yoshio Alexis Martinez Lopez.
*/

#include "FRAMES.h"

FRAMES::FRAMES(int pin) {
      _pin = pin;
      _header = 0;
}

int FRAMES::totalWork(bool *work,int swork){
      _wg = *(work);
      _wco = *(work+1);
      _wt = *(work+2);
      _wh = *(work+3);
      _wf = *(work+4);
      _totalWork = int(_wg) + int(_wco) + int(_wt) + int(_wh) + int(_wf);
      return _totalWork;
}

int FRAMES::sizeFrame(){
      _totalNoWork = (5 - _totalWork);
      _sizeFrame = (1 + (5 * _totalWork) + _totalNoWork + 1);
      return _sizeFrame;
}
uint8_t*   FRAMES::buildFrameServer(int   *id,int   sid,float   *value,int
svalue,bool   *work,int   swork,bool   *range,int   srange,bool   *vary,int
svary,bool *change,int schange,int *danger,int sdanger){
      _header = 0;
////////// Trabajan //////////
      switch(_totalWork){
            case 0:
                  bitClear(_header, 0);
```

```
            bitClear(_header, 1);
            bitClear(_header, 2);
            bitClear(_header, 3);
            break;
      case 1:
            bitClear(_header, 0);
            bitClear(_header, 1);
            bitClear(_header, 2);
            bitSet(_header, 3);

            break;
      case 2:
            bitClear(_header, 0);
            bitClear(_header, 1);
            bitSet(_header, 2);
            bitClear(_header, 3);
            break;
      case 3:
            bitClear(_header, 0);
            bitClear(_header, 1);
            bitSet(_header, 2);
            bitSet(_header, 3);
            break;
      case 4:
            bitClear(_header, 0);
            bitSet(_header, 1);
            bitClear(_header, 2);
            bitClear(_header, 3);
            break;
      case 5:
            bitClear(_header, 0);
            bitSet(_header, 1);
            bitClear(_header, 2);
            bitSet(_header, 3);
            break;
      case 6:
            bitClear(_header, 0);
            bitSet(_header, 1);
            bitSet(_header, 2);
            bitClear(_header, 3);
        break;
      case 7:
            bitClear(_header, 0);
            bitSet(_header, 1);
            bitSet(_header, 2);
            bitSet(_header, 3);
            break;
      default:
            break;
}
```

```
////////// No Trabajan //////////
      switch(_totalNoWork){
            case 0:
                  bitClear(_header, 4);
                  bitClear(_header, 5);
                  bitClear(_header, 6);
                  bitClear(_header, 7);
                  break;
            case 1:
                  bitClear(_header, 4);
                  bitClear(_header, 5);
                  bitClear(_header, 6);
                  bitSet(_header, 7);

                  break;
            case 2:
                  bitClear(_header, 4);
                  bitClear(_header, 5);
                  bitSet(_header, 6);
                  bitClear(_header, 7);
                  break;
            case 3:
                  bitClear(_header, 4);
                  bitClear(_header, 5);
                  bitSet(_header, 6);
                  bitSet(_header, 7);
                  break;
            case 4:
                  bitClear(_header, 4);
                  bitSet(_header, 5);
                  bitClear(_header, 6);
                  bitClear(_header, 7);
                  break;
            case 5:
                  bitClear(_header, 4);
                  bitSet(_header, 5);
                  bitClear(_header, 6);
                  bitSet(_header, 7);
                  break;
            case 6:
                  bitClear(_header, 4);
                  bitSet(_header, 5);
                  bitSet(_header, 6);
                  bitClear(_header, 7);
              break;
            case 7:
                  bitClear(_header, 4);
                  bitSet(_header, 5);
                  bitSet(_header, 6);
                  bitSet(_header, 7);
```

```
                  break;
            default:
                  break;
      }
////////// Tamano trama //////////
      uint8_t frame[_sizeFrame];
      frame[0] = _header;
      byte numNot = 0;
      byte peligros = 0;
      //bool works[] = {wg, wco, wt, wh, wf};
      //int ids[idg,idco,idt,idh,idf];

////////// Datos de los que si trabajan //////////
      int idW[_totalWork];
      float valueW[_totalWork];
      bool workW[_totalWork];
      bool rangeW[_totalWork];
      bool varyW[_totalWork];
      bool changeW[_totalWork];
////////// Datos de los que no trabajan //////////
      int idNW[_totalNoWork];
      float valueNW[_totalNoWork];
      bool workNW[_totalNoWork];
      bool rangeNW[_totalNoWork];
      bool varyNW[_totalNoWork];
      bool changeNW[_totalNoWork];


      int total = _totalWork + _totalNoWork;
      int countw = 0;
      int countnw = 0;
      for(int i = 0; i < total; i++){
            if(*(work + i) == true){
                  idW[countw] = *(id + i);
                  valueW[countw] = *(value + i);
                  workW[countw] = *(work + i);
                  rangeW[countw] = *(range + i);
                  varyW[countw] = *(vary + i);
                  changeW[countw] = *(change + i);
                  countw = countw + 1;
            }
            else{
                  idNW[countnw] = *(id + i);
                  valueNW[countnw] = *(value + i);
                  workNW[countnw] = *(work + i);
                  rangeNW[countnw] = *(range + i);
                  varyNW[countnw] = *(vary + i);
                  changeNW[countnw] = *(change + i);
                  countnw = countnw + 1;
            }
```

```
      }
////////// Primero los que trabajan //////////
      for(int i = 0; i < _totalWork; i++){
            numNot = 0;

            switch(idW[i]){
                  case 0:
                        bitClear(numNot, 0);
                        bitClear(numNot, 1);
                        bitClear(numNot, 2);
                        bitClear(numNot, 3);
                        break;
                  case 1:
                        bitClear(numNot, 0);
                        bitClear(numNot, 1);
                        bitClear(numNot, 2);
                        bitSet(numNot, 3);

                        break;
                  case 2:
                        bitClear(numNot, 0);
                        bitClear(numNot, 1);
                        bitSet(numNot, 2);
                        bitClear(numNot, 3);
                        break;
                  case 3:
                        bitClear(numNot, 0);
                        bitClear(numNot, 1);
                        bitSet(numNot, 2);
                        bitSet(numNot, 3);
                        break;
                  case 4:
                        bitClear(numNot, 0);
                        bitSet(numNot, 1);
                        bitClear(numNot, 2);
                        bitClear(numNot, 3);
                        break;
                  case 5:
                        bitClear(numNot, 0);
                        bitSet(numNot, 1);
                        bitClear(numNot, 2);
                        bitSet(numNot, 3);
                        break;
                  case 6:
                        bitClear(numNot, 0);
                        bitSet(numNot, 1);
                        bitSet(numNot, 2);
                        bitClear(numNot, 3);
                    break;
                  case 7:
```

```
                        bitClear(numNot, 0);
                        bitSet(numNot, 1);
                        bitSet(numNot, 2);
                        bitSet(numNot, 3);
                        break;
                  default:
                        break;
            }
            if(workW[i]){
                  bitSet(numNot, 4);
            }
            else{
                  bitClear(numNot, 4);
            }

            if(rangeW[i]){
                  bitSet(numNot, 5);
            }
            else{
                  bitClear(numNot, 5);
            }

            if(varyW[i]){
                  bitSet(numNot, 6);
            }
            else{
                  bitClear(numNot, 6);
            }
            if(changeW[i]){
                  bitSet(numNot, 7);
            }
            else{
                  bitClear(numNot, 7);
            }

            byte *bt = (byte *) &valueW[i];
            Serial.println(valueW[i]);
        frame[(i*5)+1] = numNot;
        frame[(i*5)+2] = bt[3];
        frame[(i*5)+3] = bt[2];
        frame[(i*5)+4] = bt[1];
        frame[(i*5)+5] = bt[0];
      }
////////// Ahora los que no trabajan //////////
      for(int i = 0; i < _totalNoWork; i++){
            numNot = 0;

            switch(idNW[i]){
                  case 0:
                        bitClear(numNot, 0);
```

```
            bitClear(numNot, 1);
            bitClear(numNot, 2);
            bitClear(numNot, 3);
            break;
      case 1:
            bitClear(numNot, 0);
            bitClear(numNot, 1);
            bitClear(numNot, 2);
            bitSet(numNot, 3);

            break;
      case 2:
            bitClear(numNot, 0);
            bitClear(numNot, 1);
            bitSet(numNot, 2);
            bitClear(numNot, 3);
            break;
      case 3:
            bitClear(numNot, 0);
            bitClear(numNot, 1);
            bitSet(numNot, 2);
            bitSet(numNot, 3);
            break;
      case 4:
            bitClear(numNot, 0);
            bitSet(numNot, 1);
            bitClear(numNot, 2);
            bitClear(numNot, 3);
            break;
      case 5:
            bitClear(numNot, 0);
            bitSet(numNot, 1);
            bitClear(numNot, 2);
            bitSet(numNot, 3);
            break;
      case 6:
            bitClear(numNot, 0);
            bitSet(numNot, 1);
            bitSet(numNot, 2);
            bitClear(numNot, 3);
        break;
      case 7:
            bitClear(numNot, 0);
            bitSet(numNot, 1);
            bitSet(numNot, 2);
            bitSet(numNot, 3);
            break;
      default:
            break;
}
```

```
            if(workNW[i]){
                  bitSet(numNot, 4);
            }
            else{
                  bitClear(numNot, 4);
            }

            if(rangeNW[i]){
                  bitSet(numNot, 5);
            }
            else{
                  bitClear(numNot, 5);
            }

            if(varyNW[i]){
                  bitSet(numNot, 6);
            }
            else{
                  bitClear(numNot, 6);
            }
            if(changeNW[i]){
                  bitSet(numNot, 7);
            }
            else{
                  bitClear(numNot, 7);
            }

            byte *bt = (byte *) &valueNW[i];
        frame[1 + (_totalWork*5)+i] = numNot;
      }
////////// Peligro1 //////////
      switch(*(danger)){
            case 0:
                  bitClear(peligros, 0);
                  bitClear(peligros, 1);
                  bitClear(peligros, 2);
                  bitClear(peligros, 3);
                  break;
            case 1:
                  bitClear(peligros, 0);
                  bitClear(peligros, 1);
                  bitClear(peligros, 2);
                  bitSet(peligros, 3);

                  break;
            case 2:
                  bitClear(peligros, 0);
                  bitClear(peligros, 1);
                  bitSet(peligros, 2);
                  bitClear(peligros, 3);
```

```
                  break;
            case 3:
                  bitClear(peligros, 0);
                  bitClear(peligros, 1);
                  bitSet(peligros, 2);
                  bitSet(peligros, 3);
                  break;
            case 4:
                  bitClear(peligros, 0);
                  bitSet(peligros, 1);
                  bitClear(peligros, 2);
                  bitClear(peligros, 3);
                  break;
            case 5:
                  bitClear(peligros, 0);
                  bitSet(peligros, 1);
                  bitClear(peligros, 2);
                  bitSet(peligros, 3);
                  break;
            case 6:
                  bitClear(peligros, 0);
                  bitSet(peligros, 1);
                  bitSet(peligros, 2);
                  bitClear(peligros, 3);
              break;
            case 7:
                  bitClear(peligros, 0);
                  bitSet(peligros, 1);
                  bitSet(peligros, 2);
                  bitSet(peligros, 3);
                  break;
            default:
                  break;
      }
////////// Peligro2 //////////

      switch(*(danger+1)){
            case 0:
                  bitClear(peligros, 4);
                  bitClear(peligros, 5);
                  bitClear(peligros, 6);
                  bitClear(peligros, 7);
                  break;
            case 1:
                  bitClear(peligros, 4);
                  bitClear(peligros, 5);
                  bitClear(peligros, 6);
                  bitSet(peligros, 7);

                  break;
```

```
            case 2:
                  bitClear(peligros, 4);
                  bitClear(peligros, 5);
                  bitSet(peligros, 6);
                  bitClear(peligros, 7);
                  break;
            case 3:
                  bitClear(peligros, 4);
                  bitClear(peligros, 5);
                  bitSet(peligros, 6);
                  bitSet(peligros, 7);
                  break;
            case 4:
                  bitClear(peligros, 4);
                  bitSet(peligros, 5);
                  bitClear(peligros, 6);
                  bitClear(peligros, 7);
                  break;
            case 5:
                  bitClear(peligros, 4);
                  bitSet(peligros, 5);
                  bitClear(peligros, 6);
                  bitSet(peligros, 7);
                  break;
            case 6:
                  bitClear(peligros, 4);
                  bitSet(peligros, 5);
                  bitSet(peligros, 6);
                  bitClear(peligros, 7);
              break;
            case 7:
                  bitClear(peligros, 4);
                  bitSet(peligros, 5);
                  bitSet(peligros, 6);
                  bitSet(peligros, 7);
                  break;
            default:
                  break;
      }
      frame[_sizeFrame-1] = peligros;

      return frame;


}
```