



FACULTAD DE INGENIERIA

Universidad de Buenos Aires

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Smart Public Buildings

Autor:

Ing. Lucas Fabricio Monzón Languasco

Director:

Dr. Ing. Emanuel Irrazabal (UNNE)

Jurados:

Mg. Ing. Leandro Lanzieri Rodriguez (FIUBA)

Mg. Ing. Ericson Joseph Estupiñan Pineda (FIUBA)

Esp. Ing. Rodrigo Tirapegui (FIUBA)

*Este trabajo fue realizado en la ciudad de Corrientes,
entre marzo de 2020 y diciembre de 2020.*

Resumen

En la presente memoria se aborda el diseño y desarrollo de un sistema de domótica para edificios públicos, que crea una red de nodos inteligentes con el objetivo de satisfacer las necesidades de automatización, monitoreo y eficiencia energética actuales. El trabajo contempla la selección de tecnologías de hardware y software, el diseño de la arquitectura de comunicación y la interfaz gráfica para su utilización. A lo largo del trabajo se aplicaron conceptos como programación multi-hilo y sincronización entre procesos, programación de microcontroladores en C, diseño de placas de circuito impreso y un sistema de control de versiones.

Agradecimientos

Esta sección es para agradecimientos personales y es totalmente **OPCIONAL**.

Índice general

Resumen	I
1. Introducción general	1
1.1. Domótica	1
1.1.1. Aspectos principales	1
1.1.2. Arquitectura	2
1.2. Estado del arte	3
1.3. Motivación	4
1.4. Objetivos y alcances	5
1.4.1. Objetivos	5
1.4.2. Alcance	5
2. Introducción específica	7
2.1. Funcionamiento general del sistema	7
2.2. Hardware y Firmware	7
2.3. Requerimientos	7
2.4. Planificación	7
3. Diseño e implementación	9
3.1. Solución adoptada	9
3.2. Arquitectura de funcionamiento	9
3.3. Nodos	10
3.3.1. Hardware	11
3.3.2. Firmware	14
3.4. Gateway	18
3.4.1. Hardware	18
3.4.2. Software y Firmware	20
3.5. Interfaz web	21
4. Ensayos y Resultados	25
4.1. Ensayos de comunicación	25
4.1.1. Banco de pruebas	25
4.1.2. Pruebas	26
Pruebas de conexión	26
Pruebas de Set y Get	26
Resultados	27
4.2. Ensayos de sensores y actuadores	27
4.2.1. Banco de pruebas	28
4.2.2. Pruebas	28
Pruebas de encendido y apagado de luces	28
Pruebas de encendido y apagado de aire acondicionado	29
Pruebas de sensores	30
Resultados	30

4.3. Ensayos de integración	31
4.3.1. Banco de pruebas	31
4.3.2. Pruebas	31
Pruebas de conexión con interfaz y aplicación	32
Pruebas de encendido y apagado de luz	33
Pruebas de encendido y apagado del aire acondicionado . .	34
Pruebas de sensor de temperatura y humedad	36
Pruebas de detección de movimiento	36
Resultados	37
5. Conclusiones	39
5.1. Trabajo realizado	39
5.2. Conocimientos aplicados	39
5.3. Trabajo futuro	40

Índice de figuras

1.1. Sistema de domótica genérico.	3
1.2. Sistema comercial de la marca iHaus.	4
1.3. Esquema del sistema planteado.	5
3.1. Diagrama de secuencia del sistema.	10
3.2. Circuito de fuente AC-DC.	11
3.3. Circuito de sensor de temperatura.	11
3.4. Conectores para el modulo Heltec LoRa v2.	12
3.5. Circuito de leds infrarrojos.	12
3.6. Circuito de relay.	13
3.7. Circuito de pulsadores.	13
3.8. Circuito de leds indicadores.	13
3.9. Render de placa de circuito impreso.	14
3.10. Placa de circuito impreso.	14
3.11. Arquitectura en capas del firmware del nodo.	15
3.12. Diagrama de flujo del nodo.	15
3.13. Raspberry Pi 3.	19
3.14. Módulo Heltec LoRa v2.	19
3.15. Cable USB.	19
3.16. Diagrama de flujo del transceptor.	20
3.17. Pestaña de información de la interfaz web.	22
3.18. Pestaña de configuración de la interfaz web.	22
3.19. Pestaña de reset de la interfaz web.	23
4.1. Banco de pruebas para el ensayo de comunicación.	26
4.2. Ensayo de conexión de un nuevo nodo.	26
4.3.	27
4.4.	27
4.5. Recepción de datos enviados por el nodo.	27
4.6. Banco de pruebas para el ensayo de sensores y actuadores.	28
4.7.	29
4.8.	29
4.9. Respuestas del nodo a los comandos utilizados.	30
4.10. Respuesta al cambio de temperatura y humedad en el nodo.	30
4.11. Pruebas del sensor de movimiento.	31
4.12. Banco de prueba para ensayos de integración.	32
4.13. Solapa de configuración.	32
4.14. Ventana de configuración.	33
4.15. Terminal de comandos muestra recepción del nodo.	33
4.16. Solapa de información.	33
4.17. Interfaz web luego de presionar el botón ON.	33
4.18. Envío y recepción de comandos después de presionar el botón ON.	34
4.19. Interfaz web luego de presionar el botón OFF.	34

4.20. Envío y recepción de comandos después de presionar el botón <i>OFF</i>	34
4.21. Selección de la marca del aire acondicionado en la interfaz de usuario.	35
4.22. Envío y recepción de comandos por la terminal.	35
4.23. Interfaz web luego de presionar el botón <i>ON</i>	35
4.24. Envío y recepción de comandos por la terminal.	35
4.25. Interfaz web luego de presionar el botón <i>OFF</i>	35
4.26. Envío y recepción de comandos por la terminal.	36
4.27. Recepción de datos, después del envío del comando <i>Get</i>	36
4.28. Interfaz web con los datos de temperatura y humedad	36
4.29. Interfaz web con el sistema en modo automático.	37
4.30. Interfaz web con ventana pop-up indicando una alerta de movimiento en el sector 3	37

Índice de tablas

1.1. Comparación	3
3.1. Solución adoptada	9
3.2. Estados de periféricos y variables del nodo	17
3.3. Estados de conexión	18
4.1. Banco de pruebas 1	25
4.2. Banco de pruebas 2	28
4.3. Banco de pruebas 3	31

Dedicado a... [OPCIONAL]

Capítulo 1

Introducción general

En este capítulo se realiza una introducción a la domótica para edificios públicos y monitoreo de oficinas. Asimismo, se explica la motivación, se mencionan algunos sistemas existentes en el mercado, y por último se explica el alcance y objetivos.

1.1. Domótica

Se llama domótica a los sistemas capaces de automatizar una vivienda o edificación de cualquier tipo, y que aporta servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera de la edificación. Se podría definir como la integración de tecnología en el diseño inteligente de un recinto cerrado. El término domótica proviene de la unión de las palabras *domus* que significa casa en latín y autónomo del griego *aftónomos* ("que se gobierna a sí mismo").

1.1.1. Aspectos principales

Los servicios que ofrece la domótica se pueden agrupar según cinco ámbitos principales. A continuación, se define y se indican ejemplos de cada uno.

- Programación y ahorro energético: en muchos casos no es necesario sustituir los aparatos o sistemas del hogar/edificio por otros que consuman menos energía sino realizar una gestión eficiente de los mismos.
 - Climatización y calderas: programación y zonificación, por ejemplo, el uso de un termostato.
 - Encender o apagar sistemas de luz.
 - Con un mando a distancia o control central se puede accionar un producto o agrupación de productos y activar o desactivar el funcionamiento de un sensor.
 - Gestión eléctrica.
- Confort: conlleva todas las actuaciones que se puedan llevar a cabo para mejorar la comodidad de una vivienda o edificio. Dichas actuaciones pueden ser de carácter tanto pasivo como activo.

- Iluminación: apagado general de todas las luces, automatización del apagado/encendido de cada punto de luz, regulación del nivel de luminosidad.
 - Automatización de los distintos sistemas dotándolos de control eficiente y de fácil manejo.
 - Control vía internet.
 - Generación de programas de forma sencilla para el usuario.
- Seguridad: consiste en una red de seguridad encargada de proteger tanto los bienes patrimoniales, como la seguridad personal y la vida.
 - Alarms de intrusión: se utilizan para detectar o prevenir la presencia de personas extrañas a una vivienda o edificio.
 - Detectores y alarmas de detección de incendios.
 - Comunicaciones: son los sistemas o infraestructuras de comunicaciones que posee el edificio.
 - Transmisión de alarmas.
 - Intercomunicaciones.
 - Control remoto desde internet, PC, mandos inalámbricos (por ejemplo, Wi-Fi).
 - Accesibilidad: bajo este mecanismo se incluyen las aplicaciones o instalaciones de control remoto del entorno que favorecen la autonomía personal de personas con limitaciones funcionales, o discapacidad.

1.1.2. Arquitectura

Desde el punto de vista de donde reside la inteligencia del sistema domótico, hay varias arquitecturas diferentes:

- Arquitectura centralizada: un controlador recibe información de múltiples sensores y, una vez procesada, genera las órdenes oportunas para los actuadores.
- Arquitectura distribuida: toda la inteligencia del sistema está distribuida entre los módulos, sean sensores o actuadores. Suele ser típico de los sistemas cableados o redes inalámbricas.
- Arquitectura mixta: sistemas con arquitectura descentralizada en cuanto a que disponen de varios pequeños dispositivos capaces de adquirir y procesar la información de múltiples sensores y transmitirla al resto de dispositivos distribuidos por el edificio.

En la figura [1.1] se puede ver el esquema de un sistema de domótica genérico. El componente principal es el controlador, por allí pasa toda la información de los sensores y actuadores, del lado derecho de la figura se muestran ejemplos de actuadores y del lado izquierdo, ejemplos de sensores. La interfaz es muy importante porque es el nexo entre el usuario y el sistema, en la figura se muestra

debajo del controlador las interfaces como un teclado, móvil, interruptor o interfaz web.

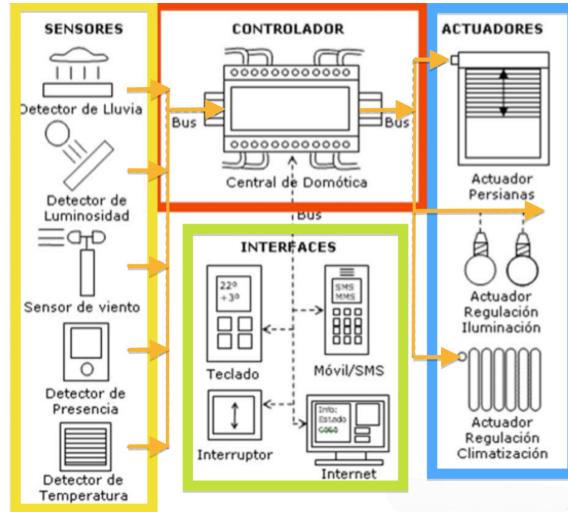


FIGURA 1.1. Sistema de domótica genérico.

1.2. Estado del arte

En la actualidad existe una amplia variedad de sistemas ofrecidos por empresas multinacionales como Fibaro, iHaus, Sonoff, ABB o Schneider Electric, que se encuentran enmarcados dentro de los sistemas de automatización y control de edificios/casas. Estos sistemas fueron tenidos en cuenta para la toma de decisiones en lo referente al desarrollo del trabajo, y por ello se resumen algunas características de estos en la tabla [1.1].

TABLA 1.1. Comparación de equipos en el mercado

Marcas	Conectividad	Interfaz	Monitoreo y Configuración
iHaus	Zigbee	Display	Software propietario
Fibaro	Z-wave	No aplica	WebServer
Sonoff	Wifi/RF	Display	Software propietario
ABB	KNX	Display	WebServer
Schneider E.	BACNet	No aplica	Software propietario

Si bien estos dispositivos se pueden adaptar a edificios de cualquier tipo, uno de los problemas más usuales en cuanto a las conexiones inalámbricas es el alcance, es decir, la distancia entre el dispositivo central y el nodo. Los dispositivos más comunes como los de iHaus, Fibaro y Sonoff son efectivos en cuanto al alcance en ámbitos residenciales, es decir, no están preparados para grandes distancias y muros anchos debido a que pierden la conectividad.

Otro aspecto a tener en cuenta en este tipo de dispositivos es la interfaz de usuario. Los sistemas de domótica en la actualidad proponen una gran variedad de

opciones para visualizar la información de los sensores y actuadores, pero algunas opciones no terminan siendo aptas para un sistema de gestión en un edificio público y además se debe agregar mas hardware para dicho requerimiento.

En la figura [1.2] se puede ver un esquema del sistema de iHaus como representación general de un sistema de domótica comercial 1.2. Éste cuenta con una central y nodos que funcionan como sensores y actuadores.

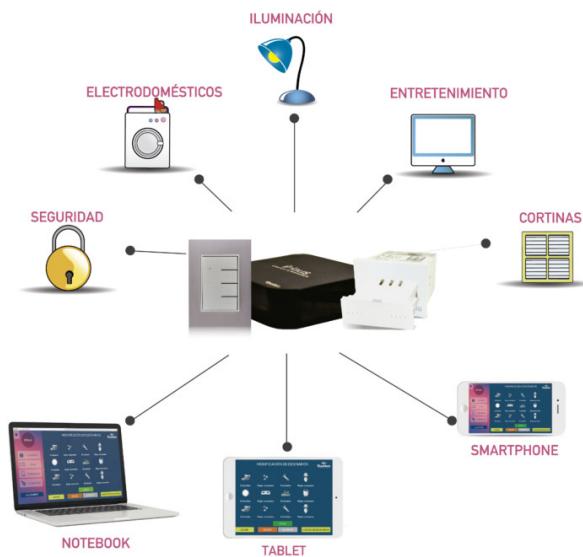


FIGURA 1.2. Sistema comercial de la marca iHaus.

1.3. Motivación

Uno de los principales desafíos en la economía actual se refiere a la reducción en el uso de distintos tipos de energías (eléctrica, térmica, etc.) y la huella de CO₂ en los existentes edificios públicos utilizando tecnologías de la información, servicios de monitoreo y manejando el consumo de la energía. Se tiene especial atención a los edificios históricos que son generalmente menos eficientes energéticamente y requieren estrictas restricciones de despliegue para evitar daños por amplia actualización. Un ámbito muy resignado por los desarrolladores de tecnología son los edificios públicos, debido a la complejidad en la instalación de este tipo de dispositivos. En la Argentina, en el marco del *Programa de Uso Racional y Eficiente de la Energía (PRONUREE) en Edificios Públicos*, que tiene como objetivo reducir los niveles de consumo energético de la administración pública nacional mediante:

- La implementación de medidas de mejora de eficiencia energética.
- La implementación de criterios para la gestión de la energía.

- La concientización del personal en el uso racional de la energía.

Para cumplir con dicho marco se propone el desarrollo de un kit capaz de generar una red de sensores y actuadores que sean de fácil instalación y además que se pueda extender mediante el uso de la tecnología modular. Disponer de estos dispositivos en edificios públicos tiene como beneficios:

- Monitoreo remoto de oficinas, aulas y espacios públicos.
- Encendido y apagado de luces y aires acondicionados.
- Mejora de la eficiencia energética de los edificios públicos.

En la figura [1.3] se puede ver un esquema de la propuesta planteada.

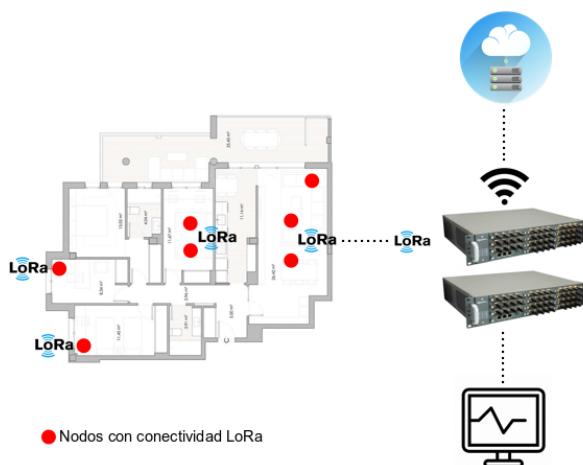


FIGURA 1.3. Esquema del sistema planteado.

1.4. Objetivos y alcances

En esta sección se hablará de los objetivos y alcances que tiene este proyecto.

1.4.1. Objetivos

El propósito de este proyecto es desarrollar un prototipo operativo de un sistema de control, monitoreo y supervisión de ciertas funciones y/o parámetros de los edificios, con la capacidad de visualizar la información de interés en un display e informar alertas. Un requisito importante es que su instalación requiera de una intervención mínima. Éste desarrollo permitirá maximizar la eficiencia del edificio, al reducir el consumo de energía y también generar alertas de prevención.

1.4.2. Alcance

Para la realización de este proyecto se desarrollará un primer prototipo operativo del sistema donde se tendrá en cuenta el hardware y software con interfaz

de comunicación LoRa (*Long Range*). El presente proyecto incluye los siguientes aspectos:

- Modelado del sistema.
- Desarrollo del firmware de los nodos y el gateway.
- Adquisición de datos de una serie de sensores en cada nodo de la red.
- Transmisión de datos entre los nodos y el gateway mediante el uso del protocolo LoRa a 915 MHz.
- Uso de base de datos en el gateway para guardar la información de cada nodo.
- Visualización de los datos adquiridos y parámetros de configuración en una aplicación web.
- Realización de tests y documentación detallados.

Capítulo 2

Introducción específica

Todos los capítulos deben comenzar con un breve párrafo introductorio que indique cuál es el contenido que se encontrará al leerlo. La redacción sobre el contenido de la memoria debe hacerse en presente y todo lo referido al proyecto en pasado, siempre de modo impersonal.

2.1. Funcionamiento general del sistema

2.2. Hardware y Firmware

2.3. Requerimientos

2.4. Planificación

Capítulo 3

Diseño e implementación

En éste capítulo se describe la estructura de la solución adoptada y el funcionamiento del hardware, software y firmware desarrollado específicamente para el cumplimiento de los requerimientos del sistema.

3.1. Solución adoptada

En base a los requerimientos enumerados en el capítulo 2, se desarrolló un sistema utilizando las tecnologías indicadas en la tabla [3.1].

TABLA 3.1. Tecnologías utilizadas en el desarrollo del sistema

Nodo	Gateway
Aplicación	Firmware en C/C++ y software en Python.
Sistema operativo	Linux
Hardware	Raspberry pi 3(Cortex-A53) y ESP32(Xtensa LX6)

Se implementó el hardware del nodo en una placa de circuito impreso de diseño específico, descrito en la sección 3.3. Contiene un microcontrolador de arquitectura MIPS(*Microprocessor without Interlocked Pipeline Stages*) de 32 bits. Cada nodo posee un circuito integrado de comunicación LoRa con su respectiva antena, sensores y actuadores, ésta permanece a la espera de recepción de comandos, conforme al protocolo propietario desarrollado.

Además se utilizó una raspberry pi como gateway descrito en la sección 3.4, a la cual se le conecta un hardware similar al nodo para utilizarlo como transceptor LoRa. La raspberry corre el sistema operativo Linux en la tarjeta microSD. También se desarrolló una aplicación en Python como *Backend* de la interfaz web creada en *Node-red*.

3.2. Arquitectura de funcionamiento

En ésta sección se explica la arquitectura utilizada para la comunicación inalámbrica del sistema. Se toma como referencia la arquitectura distribuida descrita en la sección 1.1.2. Ésta indica que la toma de decisiones por parte del sistema se

encuentra en todos los dispositivos, no solo en uno central. En la figura [3.1] se muestra el diagrama de secuencia del sistema. Éste se encuentra montado sobre una red inalámbrica local que utiliza el protocolo LoRa entre gateway y los nodos.

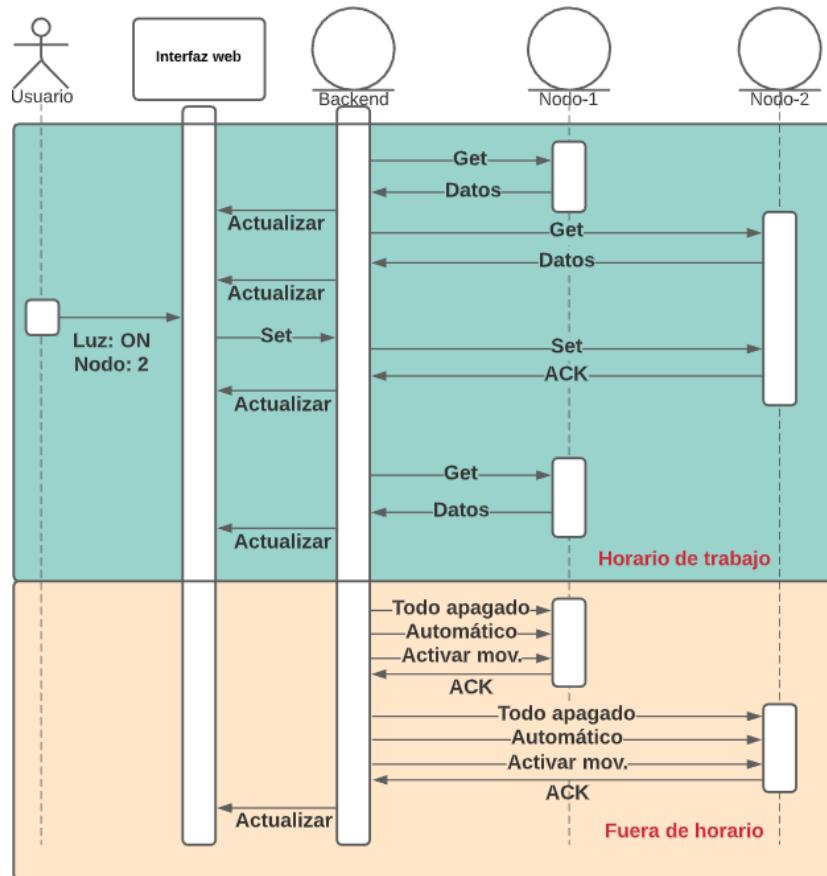


FIGURA 3.1. Diagrama de secuencia del sistema.

La arquitectura utilizada posee dos acciones posibles, estas son *Set* y *Get*. El objetivo de esta arquitectura es la de poder recibir datos con el comando *Get* y configurar parámetros con el comando *Set*, de ésta manera el sistema se reduce a una arquitectura simple y elemental que no genera grandes complicaciones en la programación de los dispositivos. En este sistema, cuando existe un nodo o una red de nodos, el gateway se encuentra enviando el comando *Get* a cada nodo en intervalos de 20 segundos. Ésto permite que la interfaz web se actualice constantemente. Al no ser un sistema crítico no se necesitan tiempos de respuesta rápidos por lo que 20 segundos es un tiempo prudente. El nodo no tiene permitido enviar datos sin antes haber recibido el comando *Get*, la única excepción es cuando está en modo automático y tiene activado el sensor de movimiento.

3.3. Nodos

El nodo es el dispositivo del sistema del cual se extrae la información de los sensores y se utilizan sus actuadores para generar cambios en dispositivos externos al sistema inalámbrico como ser un aire acondicionado. En esta sección se explicara el hardware y parte del firmware del nodo.

3.3.1. Hardware

El hardware del nodo contiene los siguientes componentes:

1. Una fuente AC-DC de 220v a 5v. Figura [3.2].
2. Sensor de temperatura y humedad. Figura [3.3].
3. Modulo Heltec LoRa v2 con antena adaptada a 915MHz. Figura [3.4].
4. Circuito de leds infrarrojos. Figura [3.5].
5. Circuito relay. Figura [3.6].
6. Circuito de pulsadores. Figura [3.7].
7. Circuito de leds indicadores. Figura [3.8].

En la figura [3.2] se tiene el circuito de la fuente. Se decidió buscar en el mercado una fuente compacta y pequeña debido al tamaño final que debe tener el producto. Se utilizó la fuente compacta HLK-PM01 de la empresa Hi-Link.

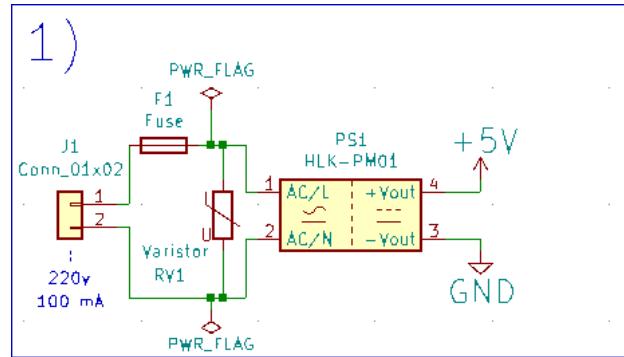


FIGURA 3.2. Circuito de fuente AC-DC.

En la figura [3.3] se muestra el circuito del sensor de temperatura y humedad. El sensor es el conocido DHT11 que tiene un rango de temperatura de 0 a 50 °C con 5 % de precisión y un rango de humedad de 20 al 80 % con una precisión del 5 %.

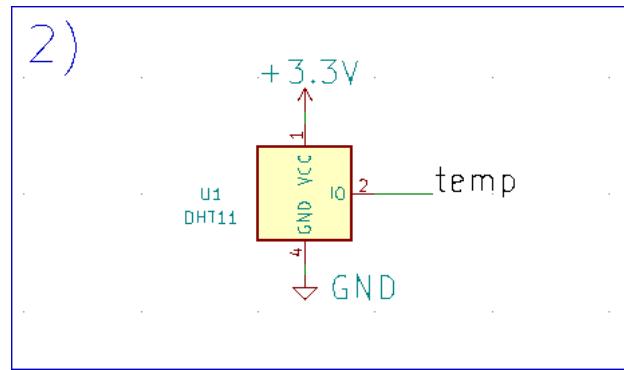


FIGURA 3.3. Circuito de sensor de temperatura.

En la figura [3.4] se ve el circuito de conectores que ofician de zócalo para conectar el modulo externo Heltec LoRa v2. Éste módulo fue elegido debido a que ofrece un microcontrolador esp32 de arquitectura Tensilica Xtensa LX6 de 32bits por lo que se utiliza como controlador principal del nodo, además posee un display oled

y el circuito integrado SX1276 de la empresa Semtech, con conector Hirose U.FL para antenas miniatura de RF de hasta 6 GHz.

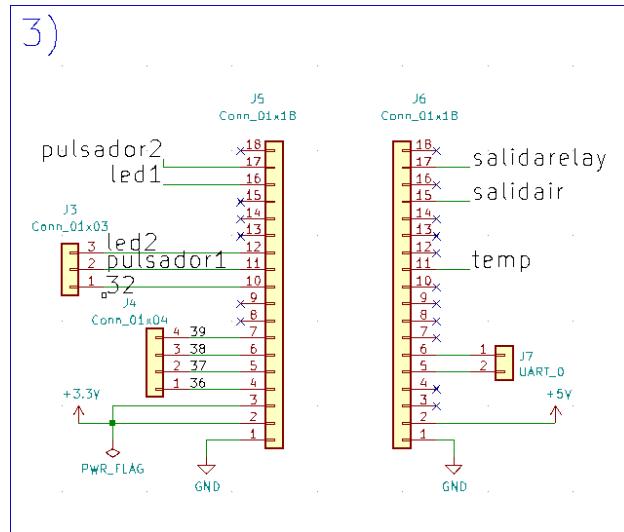


FIGURA 3.4. Conectores para el modulo Heltec LoRa v2.

En la figura [3.5] se ve un circuito típico para el uso de leds habilitado por transistor. Se utilizaron dos leds debido a que en las pruebas de alcance se obtuvo mayor rendimiento del haz infrarrojo con respecto a los aires acondicionados. El led infrarrojo es el actuador para el encendido y apagado de los aires acondicionados.

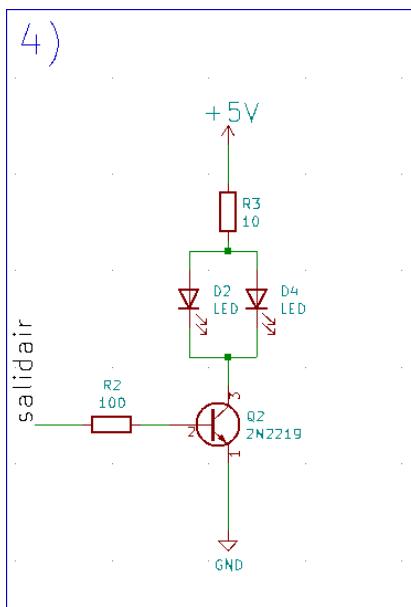


FIGURA 3.5. Circuito de leds infrarrojos.

En la figura [3.6] se tiene un circuito típico de relay con su respectivo conector de salida. El relay es el actuador para el encendido y apagado de la luz.

En la figura [3.7] se muestra el circuito de los pulsadores, éstos son el pulsador de *Modo* que se utiliza para sincronizar el nodo con el gateway y además para cambiar el modo de manual a automático, y el pulsador para encendido y apagado de la luz.

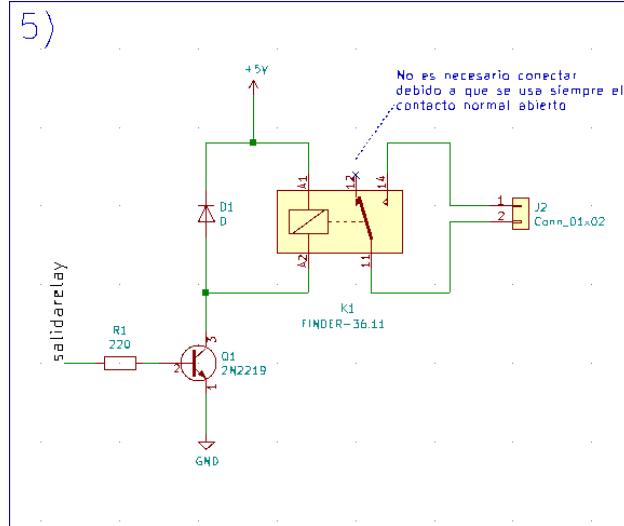


FIGURA 3.6. Circuito de relay.

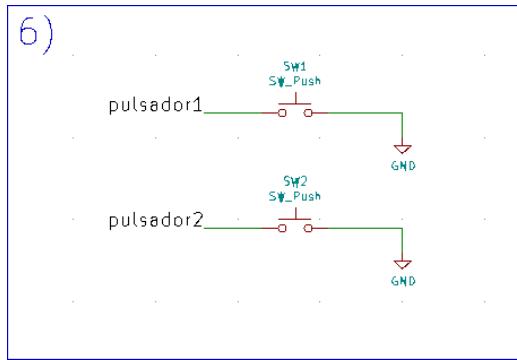


FIGURA 3.7. Circuito de pulsadores.

En la figura [3.8] se muestra el circuito de leds indicadores, se utilizan para *debugging* del sistema.

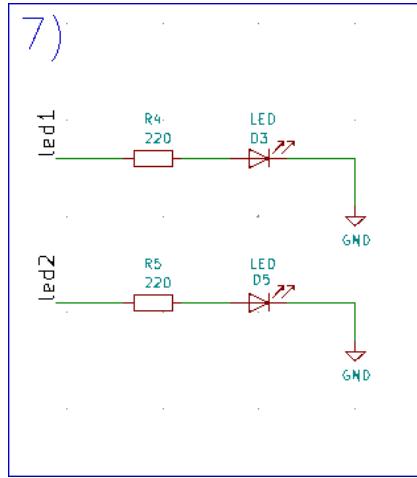


FIGURA 3.8. Circuito de leds indicadores.

La imagen renderizada de la placa de circuito impreso desarrollada en la herramienta libre Kicad, se muestra en la figura [3.9].

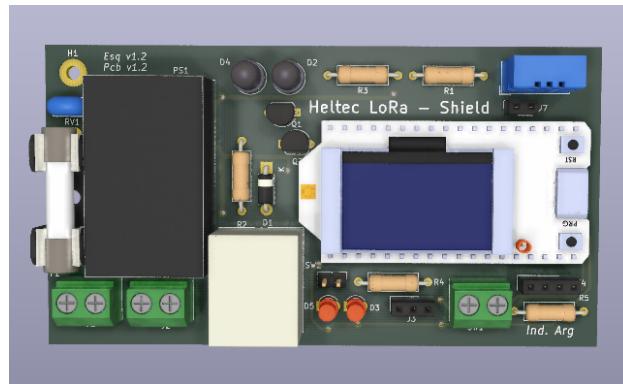


FIGURA 3.9. Render de placa de circuito impreso.

En la figura [3.10] se puede ver la imagen de la pcb terminada. Ésta es una pcb de doble capa, realizada de forma casera con el método de insolación ultravioleta.



FIGURA 3.10. Placa de circuito impreso.

3.3.2. Firmware

La arquitectura del firmware se diseño teniendo en cuenta algunos patrones de diseño de arquitectura y conceptos de programación orientada a objetos en C/C++. Para el desarrollo del firmware se utilizó la biblioteca que provee el fabricante del módulo Heltec LoRa, éste provee los drivers de la siguiente lista:

- Display Oled 128x64.
- Protocolo LoRa.

Además se utilizó una librería creada para el uso de infrarrojos con el microcontrolador ESP32, con el objetivo de agilizar el desarrollo ya que existe una gran variedad de aires acondicionados y protocolos muy variados. Se encontró que la librería provee 92 protocolos, es decir, 92 dispositivos distintos a los cuales se los puede manipular con infrarrojo. Si bien la API de la empresa *Espressif*(ESP-IDF), está desarrollada completamente sobre el sistema operativo de tiempo real freeRTOS. Se utilizó como API principal la de Arduino debido a que el sistema no necesita ser de tiempo real, ésta provee soluciones que agilizan el desarrollo en este tipo de proyectos.

El firmware posee un arquitectura en capas, esto permite la separación de las partes que componen el sistema. En la figura [3.11] se pueden ver las capas que componen el sistema.

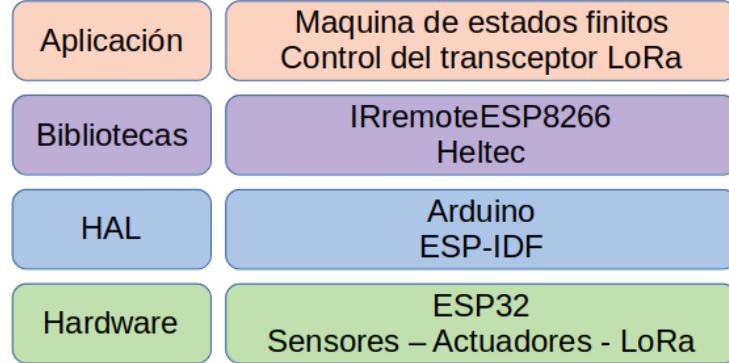


FIGURA 3.11. Arquitectura en capas del firmware del nodo.

El diagrama de flujo del sistema es el que se muestra en la figura [3.12].

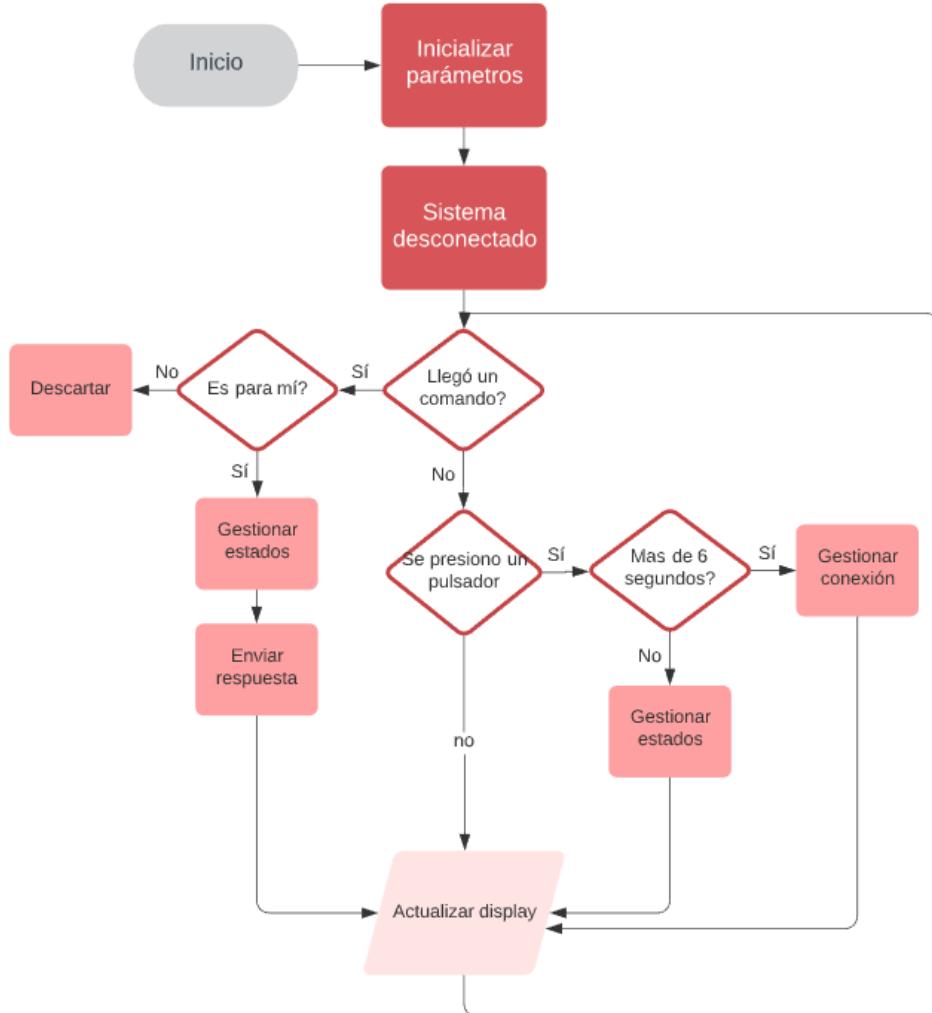


FIGURA 3.12. Diagrama de flujo del nodo.

La estructura de inicialización y el loop principal del sistema se puede ver en la sección de código [3.1].

```

1 #include <Arduino.h>
2 #include "main.h"
3
4 void setup()
5 {
6     Serial.begin(115200);
7     fsm_machine_init(&serialdata, &control, &lora_node);
8 }
9
10 void loop()
11 {
12     fsm_machine_control(&serialdata, &control, &lora_node);
13     fsm_button_check(&control, &serialdata, &lora_node);
14 }
```

CÓDIGO 3.1. Estructura principal de código. Archivo *main.cpp*

Dentro del archivo de funciones de la maquina de estados finitos (*fsm.cpp*), se tienen dos funciones principales. Éstas funciones se pueden ver en la sección de código [3.2].

```

1 void fsm_machine_init(serialdata_t *serialdata, control_t *control,
2 lora_node_t *lora_node);
3 void fsm_machine_control(serialdata_t *serialdata, control_t *control,
4 lora_node_t *lora_node);
```

CÓDIGO 3.2. Funciones de la maquina de estados finitos

La primera se encarga de inicializar todos los parámetros necesarios para que el sistema funcione. En la siguiente lista se ponen en conocimiento los parámetros inicializados antes de empezar el loop principal:

- Display oled y tamaño de fuente.
- Los pines de entrada y salida:
 - Salida a relay.
 - Salida a leds infrarrojos.
 - Entrada de señal del sensor de temperatura y humedad.
 - Salida a leds indicadores.
 - Entrada de pulsadores.
- Estados predeterminados del sistema ante un reset.
- Parámetros del protocolo LoRa como ser la frecuencia de 915MHz.

La segunda función tiene como objetivo controlar todos los estados del sistema y actualizarlos cada vez que llegue un comando o que la temperatura se actualice. Se utilizaron tres estructuras principales que se listan a continuación:

- Control
- Lora node
- Serial data

La primera se muestra en la sección de código [3.3].

```

1
2 typedef struct {
3     general_states_s    generalStates;
4     working_states_s   workingStates;
5     relay_states_s    relayStates;
6     mov_states_s      movStates;
7     a_conditioner_s   aConditioner;
8     pin_struct_t       pin;
9     dht_t              dhtVal;
10    aa_struct_t        aa_conf;
11    uint16_t            nodeId;
12 } control_t;

```

CÓDIGO 3.3. Estructura de control del nodo

Los primeros 5 elementos de la estructura corresponden a variables del tipo *ENUM* y cada una puede tomar los estados que se muestran en la tabla [3.2]:

TABLA 3.2. Estados de periféricos y variables del nodo

generalStates	workingStates	relayStates	movStates	aConditioner
Automático	Working	ON	Activado	ON
Manual	Not working	OFF	Desactivado	OFF

Una variable muy importante del sistema es la ultima de la estructura, es un *unsigned int 16* llamado *nodeId*. El problema que se tiene es que el gateway debe permitir una gran cantidad de nodos, tantos como sea posible y eso requiere que cada nodo tenga su propio numero de identificación con el objetivo de que no existan ambigüedades a la hora de la comunicación entre un nodo y un gateway, es decir, para que el gateway identifique a quien enviar y de quien recibe los paquetes. Se optó por utilizar un ID que se crea en cada nodo automáticamente y es único. La solución fue usar la dirección MAC de 64 bits que tiene cada microcontrolador esp32 y acortarla a 16 bits debido a que con esa cantidad es suficiente para un sistema de éste tipo.

La segunda estructura utilizada es la que se muestra en la sección de código [3.4]. En esta estructura se tiene un flag de conexión que cambia si el sistema se encuentra conectado o desconectado, *nodeChar* es un array donde se guarda la palabra NODE:ID y que luego sirve para comparar con comandos recibidos, finalmente *stateNode* es una variable del tipo *ENUM* que dentro tiene los estados que se listan en la tabla [3.3].

```

1
2 typedef struct lora_node_t{
3     bool           loraConnectionFlag;
4     char          nodeChar[20];
5     lora_state_s   stateNode;
6 }lora_node_t;

```

CÓDIGO 3.4. Estructura de control del protocolo LoRa

La estructura *serialdata* se puede ver en la sección de código [3.5]. Ésta estructura fue muy importante en la primera etapa de desarrollo del sistema porque se utilizó para hacer debugging a través del puerto serie propio del nodo, antes de tener una conexión inalámbrica a través de LoRa e inclusive antes de tener un gateway

TABLA 3.3. Estados de la variable stateNode.

stateNode
Conectado
Desconectado
Alarma

que se comunique con éste. Contiene a todos los comandos que deben ser comparados por el nodo cuando recibe datos de forma inalámbrica, por lo cual no se descartó su uso.

```

1  typedef struct serialdata_t
2 {
3     bool           newData;
4     char          *startMarker;
5     char          *endMarker;
6     char          rc;
7     char          receivedChars [NUMCHARS];
8     bool           configFlag;
9     bool           getDataFlag;
10    uint16_t      nodeId;
11    bool           nodeConection;
12    commands_c    commands;
13 } serialdata_t;
14 }
```

CÓDIGO 3.5. Estructura de control de comandos por el puerto serie.

Finalmente se explica la función *fsm button check* que se ve en la sección de código [3.1]. Ésta función tiene como objetivo hacer un polling de los pines de entrada de los pulsadores. Implementa funciones de antirebote para verificar los pines, ya que los pulsadores al ser un sistema mecánico presentan mucho ruido y esto interfiere en las entradas generando una medición errónea.

3.4. Gateway

El gateway es el dispositivo del sistema que maneja la información de todos los nodos y las guarda en base de datos. Se encarga de preguntar a cada nodo el estado de sus periféricos y sensores para así actualizar la base de datos, también realiza la gestión de conexión con los nodos y envío de comandos para setear parámetros. En esta sección se explicara el hardware y software del gateway.

3.4.1. Hardware

El hardware del gateway contiene los siguientes componentes:

1. Raspberry pi 3. Figura [3.13].
2. Modulo Heltec LoRa v2 con antena adaptada a 915MHz. Figura [3.14].

En la figura [3.13] se puede ver el primer componente listado anteriormente. La Raspberry es un ordenador de placa única y de bajo costo, éstos fueron los motivos por el cual se la eligió, además de tener mucho soporte por la comunidad que la utiliza. En éste proyecto se encarga de gestionar la información que envían los nodos y tenerla disponible en la base de datos para actualizar la interfaz web que se detallará en la sección [3.5].



FIGURA 3.13. Raspberry Pi 3.

El componente número dos de la lista es el módulo que se ve en la figura [3.14]. Éste módulo creado por la empresa Heltec es igual al que se utiliza en el nodo con la diferencia de que el firmware es completamente distinto. Éste gestiona la UART que incorpora, se encarga de recibir y enviar comandos a través del protocolo LoRa utilizando el circuito integrado SX1276 de la empresa Semtech, y funciona como un periférico de la Raspberry.



FIGURA 3.14. Módulo Heltec LoRa v2.

Los dispositivos listados se conectan a través de un cable USB como el de la figura [3.15].



FIGURA 3.15. Cable USB.

3.4.2. Software y Firmware

El software desarrollado para la Raspberry pi 3 fue escrito en Python que es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. A continuación se listan los procesos que gestiona el software desarrollado.

1. Base de datos creada en SQLITE.
2. Archivos .txt y .csv utilizados para el *Backend*.
3. Puerto serie conectado al transceptor LoRa.
4. Comandos enviados y recibidos.

El sistema se inicializa verificando la conexión USB con el transceptor LoRa. Si la conexión no existe, el proceso se cierra automáticamente. En el caso de que la conexión sea satisfactoria el proceso crea hilos que gestionan los siguientes ítems:

- Envío de comando *Get* a los nodos, éste comando se utiliza para que el nodo responda con información acerca de sus sensores y actuadores, todo ésto lo hace utilizando la información de la base de datos creada al agregar nuevos nodos.
- Recepción de datos por parte de los nodos, actualizando la base de datos en SQLITE.
- Gestión de los archivos que se crean al interactuar con la interfaz web.

El firmware del transceptor LoRa es muy simple. Su tarea es la de transmitir los comandos que recibe por la UART a través del protocolo LoRa y luego si recibe algo a través de LoRa, lo retransmite por la UART. El diagrama de flujos es el que se muestra en la figura [3.16]. EL firmware fue escrito en C/C++.

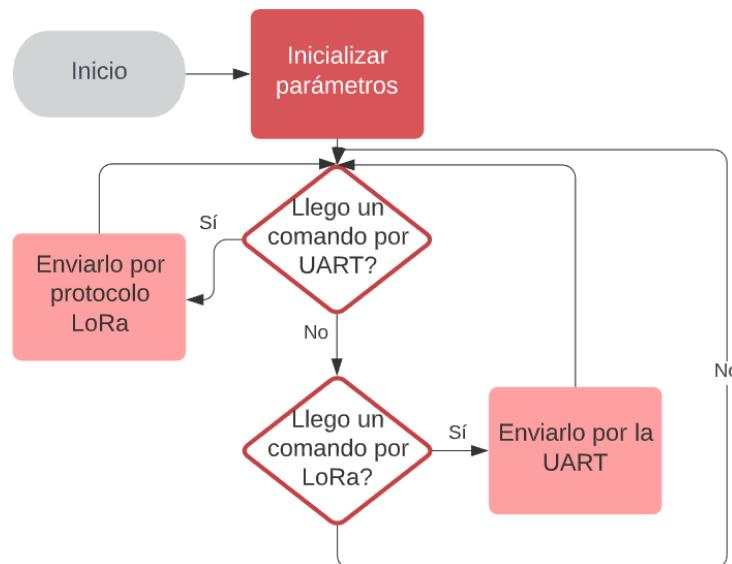


FIGURA 3.16. Diagrama de flujo del transceptor.

3.5. Interfaz web

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo. En este trabajo se comenzó a utilizar *Flask* que es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente, pero surgieron problemas por la complejidad de las funcionalidades que se le tuvo que dar a la interfaz, por lo que se decidió utilizar finalmente *Node-red* que es una herramienta de programación visual, que permite el rápido desarrollo de un *dashboard* para el sistema, de manera que agilizó el proceso completamente. Se creó un *dashboard* intuitivo para el uso del usuario, que contiene las siguientes pestañas:

- Información
- Configuración
- Reset

La pestaña de información se puede ver en la figura [3.17]. En ésta pestaña se tiene la información de los nodos como ser sensores y estados, y también botones y solapa de elección de protocolo del aire acondicionado para cada nodo. A continuación se detalla de manera mas precisa las funcionalidades.

1. Encendido y apagado de luz.
2. Encendido y apagado de aire acondicionado.
3. Elección del protocolo del aire acondicionado.
4. Elección de modo manual o automático.
5. Activar o desactivar movimiento.

La pestaña de configuración se puede ver en la figura [3.18]. Se detallan las funcionalidades de dicha pestaña en la siguiente lista.

1. Agregar un nodo nuevo al sistema y base de datos.
2. Elegir el horario de entrada del personal del edificio.
3. Elegir el horario de salida del personal del edificio.
4. Habilitar o deshabilitar el uso de los horarios de entrada y salida.

La pestaña de reset se puede ver en la figura [3.19]. Luego de un reseteo del gateway o apagón el sistema no se re establece automáticamente. Ésta opción fue agregada para casos en los que se necesite intentar conectar todos los nodos que se encuentran en la base de datos si es que alguno se reinicio por algún motivo. Una vez que se presiona el botón de reset, el gateway suspende los hilos que están en proceso y comienza a enviar un comando a cada nodo de la base de datos para conectarlo nuevamente al sistema.

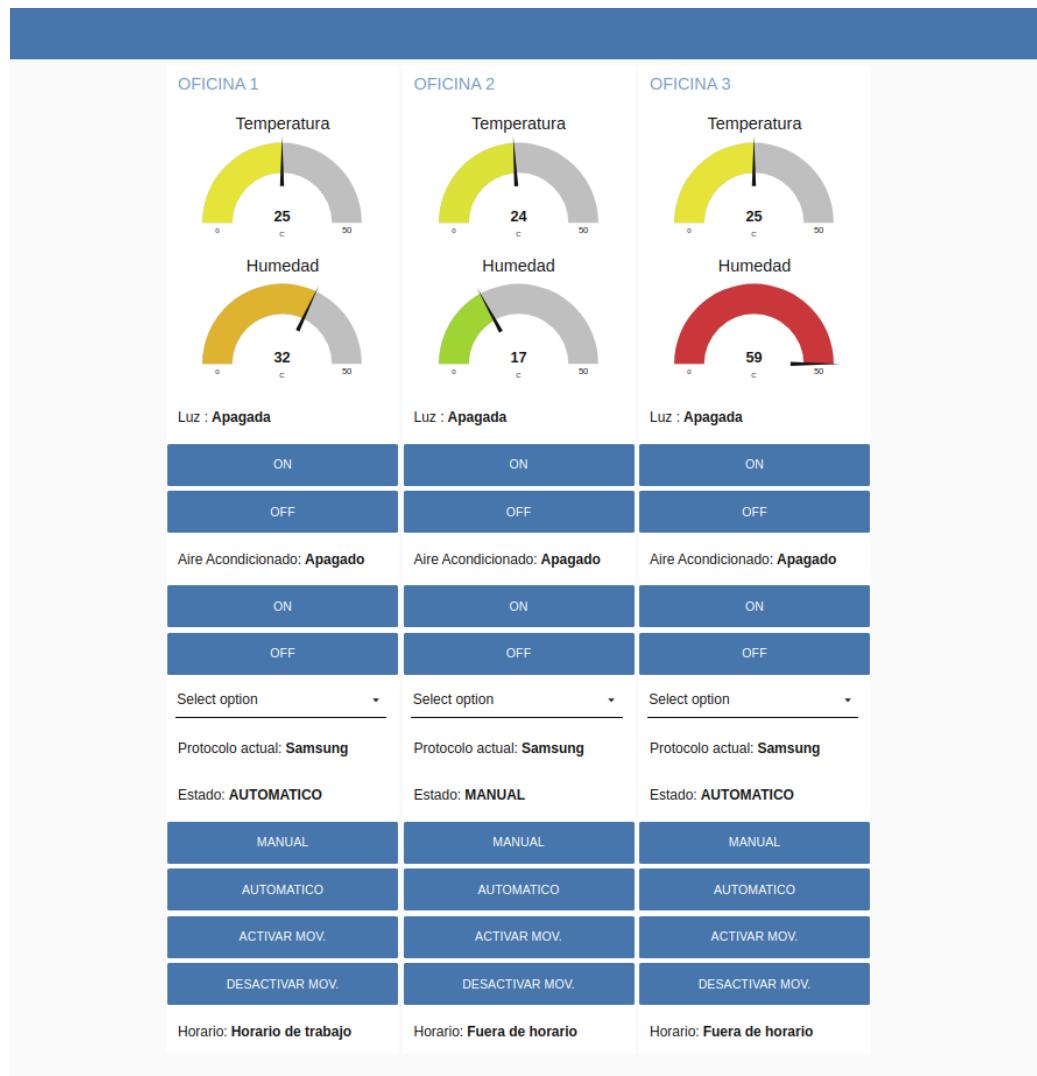


FIGURA 3.17. Pestaña de información de la interfaz web.

Agregar nuevo dispositivo	Horario de entrada	Horario de salida
	Hora ▼ 0 ▲	Hora ▼ 0 ▲
PRESIONE PARA AGREGAR	Minutos ▼ 0 ▲	Minutos ▼ 0 ▲
	Horario elegido 7:30	Horario elegido 18:30
HORARIOS <input type="button" value="HABILITAR"/> <input type="button" value="DESHABILITAR"/>		

FIGURA 3.18. Pestaña de configuración de la interfaz web.



FIGURA 3.19. Pestaña de reset de la interfaz web.

Capítulo 4

Ensayos y Resultados

En este capítulo se detallan los ensayos realizados para comprobar el correcto funcionamiento del firmware de los nodos y el *Gateway*, como así también del sistema en general.

4.1. Ensayos de comunicación

En esta sección se detallan los ensayos de comunicación entre el nodo y el *gateway* a partir de una terminal de comandos en Linux. Estos ensayos son muy importantes debido a que la comunicación inalámbrica es el requerimiento principal del proyecto. El objetivo es comprobar que existe una correcta comunicación entre los dos dispositivos.

4.1.1. Banco de pruebas

El banco de pruebas de ésta sección se componen de los elementos listados en la tabla [4.1] :

TABLA 4.1. Banco de pruebas para ensayos de comunicación

Componentes	Pertenencia	Función
Raspberry Pi	Gateway	Funciona como nexo entre el transceptor y una terminal para enviar comandos por UART.
Transceptor LoRa	Gateway	Conectado a través de la UART, recibe comandos y los envía por LoRa (915 MHz).
Display	Gateway	Conectado al puerto HDMI de la raspberry pi.
Transceptor LoRa	Nodo	Recibe comandos por LoRa, los procesa, ejecuta y envía una respuesta.

La propuesta de éste banco es la de tener los componentes mínimos y necesarios al hacer las pruebas de conectividad y de *Set/Get* mediante una terminal de comandos. En la figura [4.1] se puede ver el esquema de conexión básico.

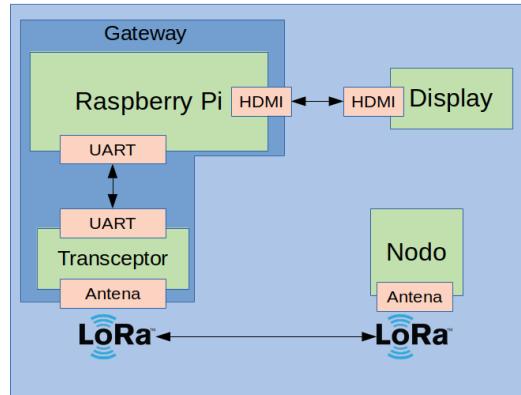


FIGURA 4.1. Banco de pruebas para el ensayo de comunicación.

4.1.2. Pruebas

Pruebas de conexión

El nodo posee dos estados de conectividad inalámbrica, éstos estados son conectado y desconectado. En ésta prueba se logra realizar la conexión inalámbrica de un nodo con el *gateway*. Los pasos realizados fueron los siguientes:

1. Con el nodo encendido, presionar el botón *Modo* por seis segundos.

En la figura [4.2] se puede ver que el *gateway* recibe el comando NODE:ID, donde ID corresponde al nodo utilizado. Cuando el *gateway* recibe éste comando, envía un ACK al nodo y éste pasa al estado conectado.



FIGURA 4.2. Ensayo de conexión de un nuevo nodo.

Pruebas de Set y Get

Las pruebas realizadas en esta sección corresponden al envío de comandos para configurar o setear algún parámetro del nodo y también recibir información de éste. Los pasos realizados fueron los siguientes:

1. Enviar el comando *Get* y esperar a recibir un array de datos.
2. Enviar el comando correspondiente para configurar el estado del sistema al modo *Manual* y esperar un ACK.
3. Enviar nuevamente el comando *Get* para corroborar el seteo del parámetro.

En la figura [4.3] se puede ver el paso 1, donde en la figura [4.3a] se envía el comando y en la figura [4.3b] se recibe el ACK.

En la figura [4.4] se puede ver el paso 2, donde en la figura [4.4a] se envía el comando y en la [4.4b] se recibe el ACK.

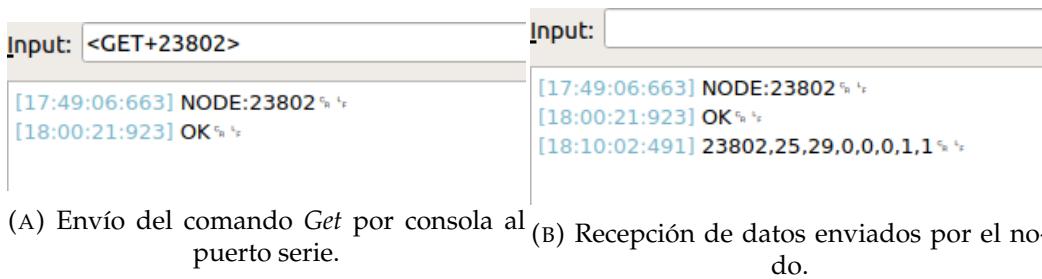


FIGURA 4.3

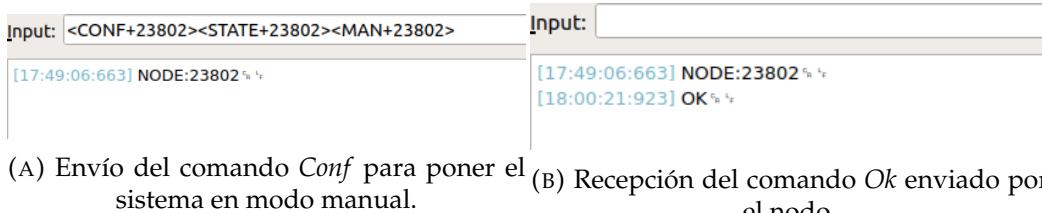


FIGURA 4.4

Finalmente en la figura [4.5] se puede ver la recepción del comando *Get* que corresponde al paso 3 y se corrobora el cambio en el parámetro.

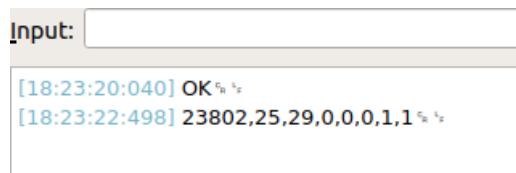


FIGURA 4.5. Recepción de datos enviados por el nodo.

Resultados

- Se comprobó que la arquitectura de comandos utilizada funciona correctamente.
- El envío y recepción de datos toma un tiempo menor a 2 segundos por comando.
- El nodo procesa satisfactoriamente la información.

4.2. Ensayos de sensores y actuadores

En esta sección se detallan los ensayos de uso de los periféricos en el nodo, mandados por el *gateway*. El objetivo de estos ensayos es corroborar el correcto funcionamiento de la placa de circuitos impreso desarrollada, como también los sensores y actuadores utilizados.

4.2.1. Banco de pruebas

El banco de pruebas de ésta sección se componen de los elementos listados en la tabla [4.2]:

TABLA 4.2. Banco de pruebas para ensayos de comunicación

Componentes	Pertenencia	Función
Raspberry Pi	Gateway	Funciona como nexo entre el transceptor y una terminal para enviar comandos por UART.
Transceptor LoRa	Gateway	Conectado a través de la UART, recibe comandos y los envía por LoRa (915 MHz).
Display	Gateway	Conectado al puerto HDMI de la raspberry pi.
Transceptor LoRa	Nodo	Recibe comandos por LoRa, los procesa, ejecuta y envía una respuesta.
Periféricos	Nodo	Sensor de movimiento, sensor de temperatura y humedad, relay y led infrarrojo

En la figura [4.6] se puede ver el esquema de conexionado básico.

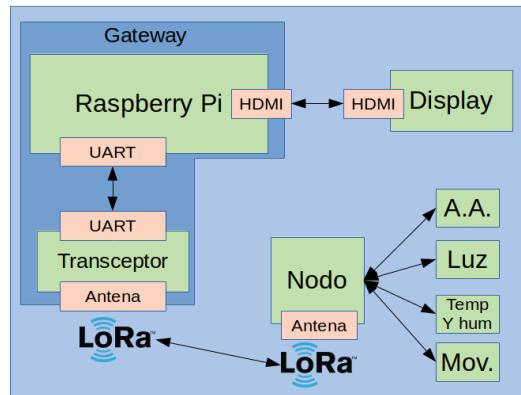


FIGURA 4.6. Banco de pruebas para el ensayo de sensores y actúadores.

4.2.2. Pruebas

Las pruebas realizadas fueron de encendido y apagado de luz y aire acondicionado, detección de movimiento y sensado de temperatura y humedad.

Se comprobó el correcto funcionamiento de forma visual en los dispositivos y además utilizando el comando *Get* para determinar si hubo un cambio en el estado de las salidas.

Pruebas de encendido y apagado de luces

Los pasos realizados en éste ensayo fueron los siguientes:

1. Enviar el comando para encender la luz, esperar un ACK.
2. Enviar el comando *Get* y esperar un array de datos de respuesta.
3. Enviar el comando para apagar la luz, esperar un ACK.
4. Enviar el comando *Get* y esperar un array de datos de respuesta.

En la figura [4.7] se puede ver el paso 1 y 2, donde en la figura [4.7a] se envía el comando y en la figura [4.7b] se recibe el ACK y el array de datos.

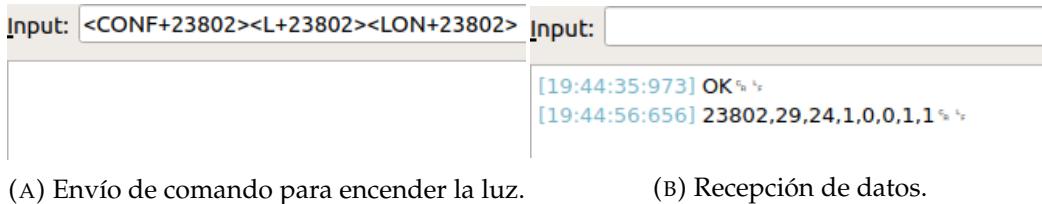


FIGURA 4.7

En la figura [4.8] se puede ver el paso 3 y 4, donde en la figura [4.8a] se envía el comando y en la figura [4.8b] se recibe el ACK y el array de datos.

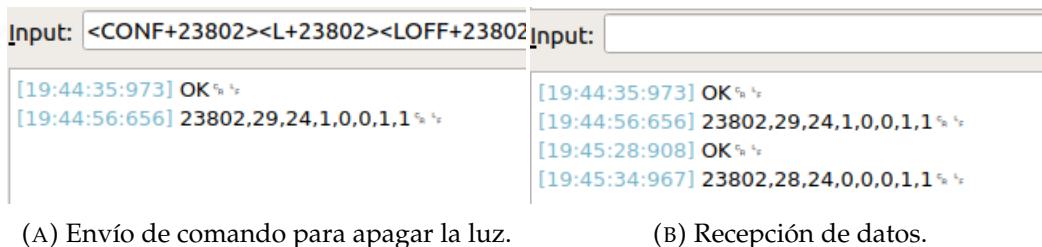


FIGURA 4.8

Si se comparan los dos arrays de datos en la figura [4.8b] se puede ver claramente el cambio de la variable en la columna 4.

Pruebas de encendido y apagado de aire acondicionado

Los pasos realizados en éste ensayo fueron los siguientes:

1. Enviar cambio de protocolo de aire acondicionado y esperar un ACK.
2. Enviar el comando para encender el aire acondicionado, esperar un ACK.
3. Enviar el comando *Get* y esperar un array de datos de respuesta.
4. Enviar el comando para apagar el aire acondicionado, esperar un ACK.
5. Enviar el comando *Get* y esperar un array de datos de respuesta.

En la figura [4.9] se puede ver la secuencia de comandos recibidos por el *gateway* correspondiente a la secuencia descrita anteriormente.

Si se comparan los dos arrays de datos en la figura [4.9] se puede ver claramente el cambio de la variable en la columna 5.

```
[19:48:57:635] OK %h %f
[19:49:22:521] OK %h %f
[19:49:35:640] 23802,27,25,0,1,46,0,1 %h %f
[19:50:11:055] OK %h %f
[19:50:23:726] 23802,27,25,0,0,46,0,1 %h %f
```

FIGURA 4.9. Respuestas del nodo a los comandos utilizados.

Pruebas de sensores

El sistema posee dos sensores, un sensor de humedad y temperatura y un sensor de movimiento. El objetivo de este ensayo es corroborar el correcto funcionamiento de estos. Los pasos para los ensayos del sensor de temperatura y humedad fueron los siguientes:

1. Enviar el comando *Get* para conocer el valor de temperatura y humedad actual.
2. Acercar una fuente de calor al sensor.
3. Enviar el comando *Get* para corroborar el cambio de temperatura y humedad.

En la figura [4.10] se puede ver la respuesta del sensor a los dos envíos del comando *Get* y se puede corroborar que los valores de temperatura y humedad han cambiado. Estos valor se pueden ver en el array, en las columnas 2 (temperatura) y 3(humedad).

```
[20:39:37:093] 23802,27,23,1,0,46,0,1 %h %f
[20:40:21:363] 23802,32,95,1,0,46,0,1 %h %f
```

FIGURA 4.10. Respuesta al cambio de temperatura y humedad en el nodo.

Finalmente se hizo el ensayo del sensor de movimiento, para el que se siguieron los pasos:

1. Enviar el comando para cambiar el estado del sistema a *Automático* y esperar un *ACK*.
2. Enviar el comando para activar el sensor de movimiento y esperar un *ACK*.
3. Mover un objeto frente al sensor de movimiento.

En la figura [4.11] se puede ver la secuencia descrita anteriormente. Se reciben los dos *ACK* y luego se procede al paso 3 que genera el envío del comando de alarma del nodo al *gateway*.

Resultados

- Se comprobó que los sensores y actuadores funcionan correctamente.
- El envío de datos al sensar movimiento tiene un tiempo menor a 2 segundos y lo hace de forma correcta.

```
[20:46:58:513] OK %f
[20:47:05:216] OK %f
[20:47:06:229] AL+23802 %f
[20:47:12:492] AL+23802 %f
[20:47:13:484] AL+23802 %f
[20:47:20:366] AL+23802 %f
[20:47:21:358] AL+23802 %f
[20:47:22:375] AL+23802 %f
```

FIGURA 4.11. Pruebas del sensor de movimiento.

- El circuito impreso desarrollado funciona correctamente.

4.3. Ensayos de integración

En esta sección se detallan los ensayos de comunicación entre el nodo y el *gateway*, para ello se utiliza una interfaz web y un proceso programado en *Python* que maneja las peticiones de la interfaz de usuario creada en *Node-red*. El objetivo de estos ensayos es corroborar el correcto funcionamiento del sistema con el agregado de la interfaz web y simulando el sistema completo.

4.3.1. Banco de pruebas

El banco de pruebas de ésta sección se componen de los elementos listados en la tabla [4.3]:

TABLA 4.3. Banco de pruebas para ensayos de comunicación

Componentes	Pertenencia	Función
Raspberry Pi	Gateway	Ejecuta la aplicación programada en <i>Python</i> y la interfaz web.
Transceptor LoRa	Gateway	Conectado a través de la UART, recibe comandos y los envía por LoRa (915 MHz).
Display	Gateway	Conectado al puerto HDMI de la raspberry pi.
Transceptor LoRa	Nodo	Recibe comandos por LoRa, los procesa, ejecuta y envía una respuesta.
Periféricos	Nodo	Sensor de movimiento, sensor de temperatura y humedad, relay y led infrarrojo

En la figura [4.12] se puede ver el esquemático básico del sistema. Se utiliza la ventana de comandos con el motivo de mostrar que los comandos son enviados correctamente cuando se interactúa con la interfaz web.

4.3.2. Pruebas

Las pruebas realizadas se detallan en la siguiente lista:

1. Ensayo de conexión con interfaz y aplicación.

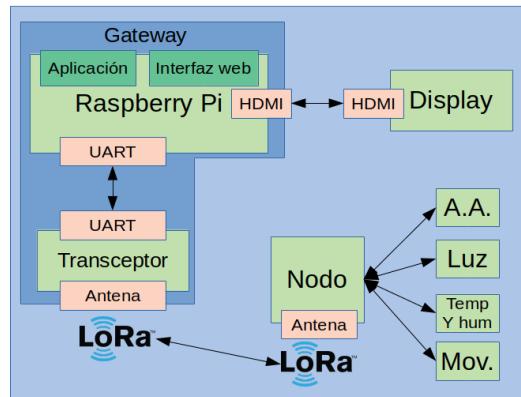


FIGURA 4.12. Banco de prueba para ensayos de integración.

2. Ensayo de encendido y apagado de luz.
3. Ensayo de encendido y apagado de aire acondicionado.
4. Ensayo de sensor de temperatura y humedad.
5. Ensayo de sensor de movimiento.

Pruebas de conexión con interfaz y aplicación

Se realizó el ensayo de conexión de un nodo al sistema mediante el uso de la interfaz web. Para ello se siguen los pasos:

1. Hacer click en la solapa *Configuración*.
2. Hacer click en *Agregar nuevo dispositivo*.
3. Presionar el botón *Modo* del nodo durante seis segundos.

En las figuras [4.13] y [4.14] se ven los pasos 1 y 2. Aquí se selecciona la solapa de configuración para tener disponible el botón de agregado de un nodo al sistema.



FIGURA 4.13. Solapa de configuración.

En la figura [4.15] se muestra la terminal de Linux en la que luego del paso 3 mencionado, llega el comando NODE:ID correspondiente al nodo en cuestión. Finalmente el sistema escribe en un archivo de log, la línea que se muestra en la parte inferior de la figura.



FIGURA 4.14. Ventana de configuración.

```
Received: NODE:23802
Node ID: 23802 added to the list
```

FIGURA 4.15. Terminal de comandos muestra recepción del nodo.

Pruebas de encendido y apagado de luz

Se realizó el ensayo de encendido y apagado de luz mediante el uso de los botones ON y OFF de la interfaz web, correspondientes al nodo en cuestión. Para ello se siguen los pasos:

1. Hacer click en la solapa *Información*.
2. Hacer click en ON.
3. Hacer click en OFF.

En la figura [4.16] se puede ver el paso 1, aquí se selecciona la solapa de información que se encuentra en la parte izquierda de la pagina web.



FIGURA 4.16. Solapa de información.

En la figura [4.17] se muestran los botones que provee la pagina web para el encendido y apagado de luz, también se ve el estado actual. Una vez presionado el botón ON se puede ver en la figura [4.18] el envío del comando al nodo y la recepción del ACK. Luego el sistema envía el comando Get automáticamente para corroborar el cambio y actualizar el estado en la interfaz web.



FIGURA 4.17. Interfaz web luego de presionar el botón ON.

```
Send: <CONF+23802><L+23802><LON+23802> to:23802
Received: OK

Send: <GET+23802> to:23802
Received: 23802,28,31,1,0,0,1,1
```

FIGURA 4.18. Envío y recepción de comandos después de presionar el botón *ON*.

En las figuras [4.19] y [4.20] se puede ver el mismo mecanismo utilizado anteriormente en las figuras [4.17] y [4.18] pero en este caso se apaga la luz.

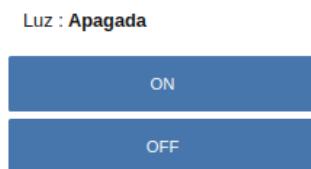


FIGURA 4.19. Interfaz web luego de presionar el botón *OFF*.

```
Send: <CONF+23802><L+23802><LOFF+23802> to:23802
Received: OK

Send: <GET+23802> to:23802
Received: 23802,28,30,0,0,0,1,1
```

FIGURA 4.20. Envío y recepción de comandos después de presionar el botón *OFF*.

Pruebas de encendido y apagado del aire acondicionado

De la misma manera que en la prueba anterior, se procedió a ensayar el encendido y apagado del aire acondicionado, para ésto se siguen los pasos:

1. Hacer click en la solapa *Información*.
2. Elegir la marca del aire acondicionado.
3. Hacer click en *ON*.
4. Hacer click en *OFF*.

En la figura [4.21] se muestra la apertura del menú que contiene las marcas de aires acondicionados y se selecciona una.

Una vez seleccionada la marca, se muestra en la figura [4.22] el correspondiente envío del comando al nodo y la respuesta de éste.

En la figura [4.23] se muestra el estado actual y los botones que provee la interfaz para el encendido y apagado del aire acondicionado.

En la figura [4.24] se ve el envío del comando de encendido del aire acondicionado y la respuesta del nodo.

En las figuras [4.25] y [4.26] se puede ver el mismo mecanismo utilizado anteriormente en las figuras [4.23] y [4.24] pero en este caso se apaga el aire acondicionado.



FIGURA 4.21. Selección de la marca del aire acondicionado en la interfaz de usuario.

```
<CONF+23802><AA+23802><ANUM+23802><46+23802>
Received: OK
```

FIGURA 4.22. Envío y recepción de comandos por la terminal.

Aire Acondicionado: Encendido

ON

OFF

SAMSUNG

Protocolo actual: Samsung

FIGURA 4.23. Interfaz web luego de presionar el botón ON.

```
Send: <CONF+23802><AA+23802><AON+23802> to:23802
Received: OK

Send: <GET+23802> to:23802
Received: 23802,25,82,0,1,46,0,1
```

FIGURA 4.24. Envío y recepción de comandos por la terminal.

Aire Acondicionado: Apagado

ON

OFF

SAMSUNG

Protocolo actual: Samsung

FIGURA 4.25. Interfaz web luego de presionar el botón OFF.

```
Send: <CONF+23802><AA+23802><AOF+23802> to:23802
Received: OK

Send: <GET+23802> to:23802
Received: 23802,25,80,0,0,46,0,1
```

FIGURA 4.26. Envío y recepción de comandos por la terminal.

Pruebas de sensor de temperatura y humedad

En este ensayo se corrobora que los datos mostrados en la interfaz web acerca del sensor de temperatura y humedad se corresponden con los datos recibidos por el nodo.

En las figura [4.27] se pueden ver los datos recibidos por el nodo luego de que el sistema le mande un comando *Get* al nodo, en las columnas 2 y 3 se puede ver el valor de temperatura y humedad respectivamente, y en la figura [4.28] se muestra la interfaz web y se corrobora el correcto funcionamiento de la interfaz mostrando los valores reales recibidos.

```
Send: <GET+23802> to:23802
Received: 23802,26,65,0,0,46,0,1
```

FIGURA 4.27. Recepción de datos, después del envío del comando *Get*

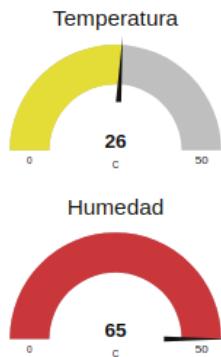


FIGURA 4.28. Interfaz web con los datos de temperatura y humedad

Pruebas de detección de movimiento

En esta sección se ensaya el uso del sensor de movimiento. Los pasos realizados fueron:

1. Hacer click en la solapa *Información*.
2. Poner el sistema en modo Automático.
3. Activar la detección de movimiento.

El sistema se pone en modo automático mediante el uso del botón *Modo* que se encuentra en el nodo, pero también se lo puede hacer mediante la interfaz web.

En la figura [4.29] se puede ver que la interfaz web nos indica que el sistema se encuentra en modo automático y además provee los botones para activar y desactivar el sensor de movimiento.



FIGURA 4.29. Interfaz web con el sistema en modo automático.

Se puede ver en la figura [4.30] que luego del paso 3, se activa el sensor de movimiento y al pasar un objeto por delante, el sistema lo detecta y activa una ventana de emergencia en la interfaz web.

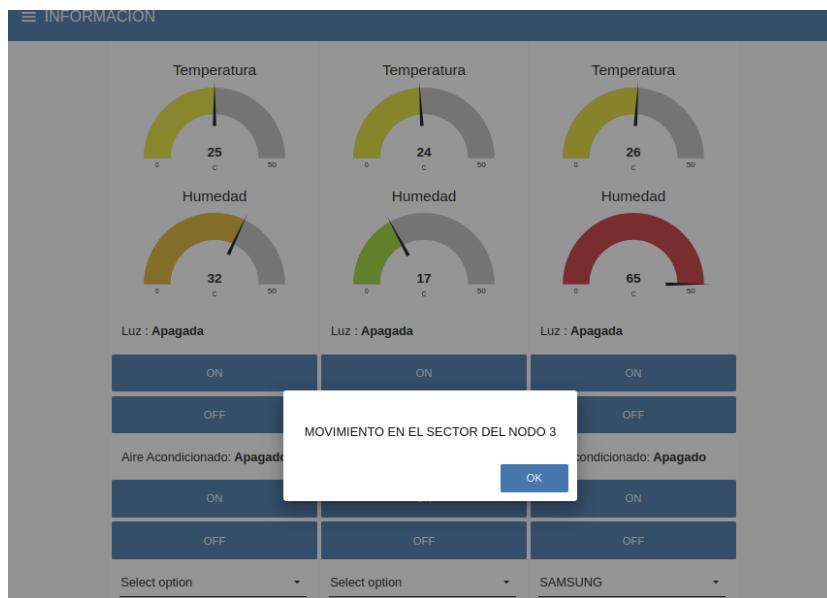


FIGURA 4.30. Interfaz web con ventana pop-up indicando una alerta de movimiento en el sector 3

Resultados

- Se comprobó que la interfaz web funciona correctamente.
- Los comandos son enviados y recibidos correctamente.
- La sincronización de hilos en la aplicación funciona como es esperado.

Capítulo 5

Conclusiones

En este capítulo se realiza un resumen sobre los conocimientos aplicados, el trabajo realizado hasta el momento y problemas que surgieron durante el desarrollo.

5.1. Trabajo realizado

En la presente memoria se documentó la implementación de un prototipo de domótica para edificios públicos. Particularmente se implementó el monitoreo de temperatura y humedad, el control de encendido y apagado de luces y aires acondicionados para diferentes marcas y la detección de movimiento en el área.

Se desarrolló e implementó satisfactoriamente una red de nodos que responden a una central y que proveen una estructura de servicios propia y local sin necesidad de una red Wi-Fi. La implementación permite, no solo generar eficiencia energética en edificios públicos de gran tamaño, sino también la detección de movimiento en las oficinas o áreas donde se encuentre un nodo. Ésto es importante para generar una alarma preventiva. También presenta una interfaz de visualización de fácil uso para el usuario.

Se hicieron modificaciones de los requisitos a lo largo del proyecto debido al planteo del cliente. Los requisitos no fueron cumplidos en su totalidad, quedando para una segunda etapa las tareas de diseño de un hardware más pequeño como así también las pruebas en planta.

Surgieron nuevos riesgos que no estaban considerados al plantear el proyecto y que no pudieron ser mitigados con satisfacción. No obstante, se concluye que la mayoría de los objetivos planteados al comienzo del trabajo fueron alcanzados satisfactoriamente y se han obtenido conocimientos valiosos para la formación del autor.

5.2. Conocimientos aplicados

Durante el desarrollo del este trabajo se aplicaron conocimientos adquiridos a lo largo del año en la Especialización de Sistemas Embebidos. Todas las asignaturas cursadas aportaron conocimientos necesarios para que el trabajo finalmente se encuentre funcionando. Sin embargo, se resaltan a continuación aquellas materias de mayor relevancia para este trabajo.

- Gestión de Proyectos: la elaboración de un plan de proyecto para organizar el trabajo final, facilitó la realización del mismo.
- Ingeniería de Software en Sistemas Embebidos: la creación de un documento de especificaciones de requerimientos permitió ordenar y documentar las necesidades del cliente además de enseñar el uso de sistemas de control de versiones.
- Protocolos de Comunicación: resulto de utilidad para la creación del driver de uno de los componentes.
- Sistemas Operativos de Propósito General: se aplicaron conceptos de uso de hilos y procesos, como también de sincronización entre procesos y uso de variables compartidas.
- Desarrollo de Aplicaciones en Sistemas Operativos: se aplicaron conocimientos del lenguaje de programación Python, uso de ficheros y la programación orientada a objeto.
- Diseño de Circuitos Impresos: aportó al uso de la herramienta libre KiCad.

5.3. Trabajo futuro

Resulta imprescindible identificar el trabajo futuro, para dar continuidad al esfuerzo realizado hasta el momento y poder realizar un producto comercialmente atractivo. A continuación se listan las líneas de trabajo mas importantes:

- Diseñar un prototipo de hardware más pequeño para su uso dentro de cajas de electricidad convencionales.
- Modificar la interfaz web para agregar nodos de forma automática.
- Separar los leds infrarrojos del nodo para poder situarlos mas cerca del aire acondicionado.
- Agregar la posibilidad de actualizar el firmware del gateway de forma remota.