

## CARRERA DE ESPECIALIZACIÓN EN INTERNET DE LAS COSAS

MEMORIA DEL TRABAJO FINAL

### **Monitoreo y gestión remota de red de sensores en invernaderos**

**Autor:**  
**Ing. Facundo Andrioli Villa**

Director:  
Dr. Lic. Pablo Ventura (UNC-KeyLab)

Jurados:  
Nombre del jurado 1 (pertenencia)  
Nombre del jurado 2 (pertenencia)  
Nombre del jurado 3 (pertenencia)

*Este trabajo fue realizado en la ciudad de Monte Cristo,  
entre agosto de 2023 y diciembre de 2024.*



## *Resumen*

En la presente memoria se describe la implementación de una red de sensores colocados en invernaderos para la recopilación de información en tiempo real. Este trabajo, desarrollado para la empresa Wentux, tiene como objetivo mejorar el control y la eficiencia de los invernaderos, lo que permite a los usuarios acceder a los datos y gestionar alarmas desde cualquier lugar.

Se aplicaron conocimientos referidos a protocolos de comunicación, procesamiento de mensajes, desarrollo de aplicaciones web y gestión de datos en la nube.



## *Agradecimientos*

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



# Índice general

<b>Resumen</b>	<b>I</b>
<b>1. Introducción general</b>	<b>1</b>
1.1. Sistema de monitoreo y gestión en invernaderos . . . . .	1
1.1.1. Problemática actual . . . . .	1
1.1.2. Wentux . . . . .	1
1.1.3. Oportunidades de mejora . . . . .	2
1.2. Motivación . . . . .	2
1.3. Estado del arte . . . . .	2
1.4. Alcance y objetivos . . . . .	3
1.5. Requerimientos . . . . .	4
<b>2. Introducción específica</b>	<b>5</b>
2.1. Esquema general del sistema . . . . .	5
2.2. Tecnologías de hardware . . . . .	6
2.2.1. Microcontrolador . . . . .	6
2.2.2. Sensores . . . . .	7
2.3. Tecnologías de firmware . . . . .	8
2.3.1. MicroPython . . . . .	8
Elección de MicroPython . . . . .	8
2.4. Protocolos de comunicación . . . . .	9
2.4.1. Protocolo ESP-NOW . . . . .	9
2.4.2. Protocolo MQTT . . . . .	10
2.5. Aplicación web progresiva . . . . .	11
2.6. Servidor de internet de las cosas . . . . .	11
2.6.1. Servidor IoT . . . . .	11
2.6.2. OpenRemote . . . . .	12
<b>3. Diseño e implementación</b>	<b>13</b>
3.1. Arquitectura del sistema . . . . .	13
3.1.1. Nodos sensores . . . . .	13
3.1.2. Nodo central . . . . .	14
3.1.3. Servidor IoT . . . . .	14
3.2. Desarrollo del firmware . . . . .	14
3.2.1. Nodos sensores y módulo relés . . . . .	14
Nodos sensores . . . . .	14
Módulo relés . . . . .	15
3.2.2. Nodo Central . . . . .	16
Loop AP . . . . .	17
Loop CL . . . . .	18
3.3. Desarrollo de la aplicación web progresiva . . . . .	19
3.3.1. Login de la PWA . . . . .	19

3.3.2. Pagina principal . . . . .	20
3.3.3. Pantalla de configuración . . . . .	22
3.4. Implementación del servidor IoT . . . . .	24
3.4.1. Login . . . . .	24
3.4.2. Agente MQTT y activos . . . . .	25
Agente MQTT . . . . .	25
Activos (Assets) . . . . .	26
3.4.3. Reglas . . . . .	27
3.4.4. Dashboard . . . . .	27
<b>4. Ensayos y resultados</b>	<b>29</b>
4.1. Banco de pruebas . . . . .	29
4.2. Prueba de componentes . . . . .	30
4.2.1. Prueba de sensores . . . . .	30
Resultados . . . . .	30
4.2.2. Prueba del módulo de relés . . . . .	31
Resultados . . . . .	31
4.3. Pruebas del sistema: Microcontrolador y Firmware . . . . .	32
4.4. Pruebas sobre la aplicación web progresiva . . . . .	34
4.4.1. Prueba de inicio de sesión . . . . .	34
4.4.2. Prueba de visualización de datos de sensores . . . . .	35
4.4.3. Prueba de control de relés . . . . .	36
4.4.4. Prueba de configuración . . . . .	37
4.5. Pruebas sobre el servidor OpenRemote . . . . .	38
4.5.1. Pruebas de recepción de datos desde el broker MQTT . . . . .	38
4.5.2. Pruebas de publicación de comandos a los relés . . . . .	39
4.5.3. Pruebas de almacenamiento y gestión de datos . . . . .	40
<b>5. Conclusiones</b>	<b>43</b>
5.1. Conclusiones generales . . . . .	43
5.2. Próximos pasos . . . . .	44
<b>Bibliografía</b>	<b>45</b>

# Índice de figuras

2.1. Diagrama en bloques del sistema.	6
2.2. Módulo ESP32-C3.	7
2.3. Sensor LM35.	7
2.4. Sensor HTU21.	7
2.5. Sensor BME280.	8
2.6. Sensor MH-Z19.	8
2.7. Modelo ESP-NOW.	9
2.8. Arquitectura MQTT [1].	10
2.9. Arquitectura OpenRemote [2].	12
3.1. Esquema del sistema.	13
3.2. Diagrama de flujo de los nodos sensores.	15
3.3. Diagrama de flujo del módulo relés.	16
3.4. Diagrama de flujo del nodo central.	17
3.5. Diagrama de flujo del loop AP.	18
3.6. Diagrama de flujo del loop CL.	18
3.7. Pantalla de login.	19
3.8. Pantalla de login con datos incorrectos.	20
3.9. Pantalla principal.	20
3.10. Pantalla principal.	21
3.11. Pantalla de configuración.	23
3.12. Pantalla de configuración.	23
3.13. Pantalla de login de OpenRemote.	24
3.14. Pantalla de login de OpenRemote con datos incorrectos.	25
3.15. Configuración del agente mqtt.	25
3.16. Configuración del activo.	26
3.17. Pantalla principal de los activos.	27
3.18. Pantalla principal de las reglas.	27
3.19. Pantalla principal del dashboard.	28
4.1. Prototipo utilizado para las pruebas.	30
4.2. Datos del sensor LM35.	31
4.3. Datos del sensor MHZ19.	31
4.4. Prueba relé.	32
4.5. Conexión de nodos al gateway.	33
4.6. Comando para reiniciar a los nodos.	33
4.7. Nodo reiniciado.	34
4.8. Error de login.	35
4.9. Mensaje sin iniciar sesión.	35
4.10. Comparación de datos.	36
4.11. Actualización periódica de datos.	36
4.12. Prueba relés.	37
4.13. Registro de error.	38

4.14. Verificación de conexión de openremote con el broker. . . . .	38
4.15. Recepción de datos en OpenRemote. . . . .	39
4.16. Envió de comandos desde OpenRemote. . . . .	39
4.17. Verificación de cambio de estado del relé. . . . .	40
4.18. Opción de historia del atributo. . . . .	41
4.19. Grafico del sensor de temperatura. . . . .	41

# Índice de tablas

1.1. Comparación de soluciones comerciales . . . . .	3
--	---



*Dedicado a... [OPCIONAL]*



# Capítulo 1

## Introducción general

En este capítulo se presenta una visión general de los sistemas de monitoreo y gestión de invernaderos, y se abordan los desafíos actuales y las oportunidades de mejora. Además, se describen las motivaciones del trabajo, sus objetivos, el alcance de la solución propuesta y los requerimientos.

### 1.1. Sistema de monitoreo y gestión en invernaderos

La gestión eficiente de los invernaderos es crucial para maximizar la productividad agrícola, especialmente en un contexto global donde la demanda de alimentos sigue en aumento. Los invernaderos, al ofrecer un entorno controlado para el cultivo, permiten optimizar las condiciones de crecimiento de las plantas. Sin embargo, la evolución de las tecnologías de monitoreo y gestión ha revelado tanto desafíos persistentes como nuevas oportunidades para mejorar estos sistemas.

#### 1.1.1. Problemática actual

La producción agrícola en invernaderos ha evolucionado considerablemente en respuesta a la creciente demanda de cultivos y al aumento de la población mundial. En este contexto, se enfrentan desafíos claves que afectan la eficiencia y productividad:

- Descentralización geográfica: la supervisión de invernaderos dispersos resulta difícil, ya que complica la obtención y el análisis de datos en tiempo real y puede derivar en respuestas tardías a cambios críticos en el entorno de cultivo.
- Falta de unificación en la gestión: los sistemas actuales suelen ser fragmentados, lo que dificulta la implementación de un control eficiente y coordinado en todos los aspectos del cultivo.
- Limitaciones tecnológicas: la infraestructura existente no siempre soporta la recopilación continua y precisa de datos ambientales, lo que afecta la toma de decisiones informada.

#### 1.1.2. Wentux

Wentux [3] es una empresa argentina que se especializa en el desarrollo de soluciones tecnológicas para la gestión y monitoreo de invernaderos. Ofrece productos que optimizan el ambiente controlado de los cultivos mediante el uso de tecnologías avanzadas, como sensores de humedad de suelo, controladores de

CO<sub>2</sub>, sistemas de riego y kits para hidroponía. Estos productos están orientados a mejorar la eficiencia y productividad en entornos agrícolas de precisión, especialmente en invernaderos.

La empresa enfrenta varias problemáticas relacionadas con la descentralización geográfica de los invernaderos y la falta de un sistema unificado que permita una gestión centralizada y eficiente. Además, busca soluciones que puedan facilitar el monitoreo remoto y el control en tiempo real de las condiciones ambientales, lo que podría aumentar considerablemente la capacidad de respuesta a los cambios críticos en el entorno de cultivo.

### 1.1.3. Oportunidades de mejora

A la luz de estos desafíos, surgen varias oportunidades para mejorar la eficiencia y efectividad de los sistemas de monitoreo y gestión en invernaderos, orientadas a satisfacer las necesidades tanto de los productores como de la industria en general:

- Implementación de tecnologías avanzadas: la integración de sensores y sistemas de monitoreo más sofisticados puede permitir una recopilación de datos más precisa y en tiempo real, lo que provoca una mejora en la capacidad de respuesta ante cambios en el entorno.
- Centralización y unificación del control: la adopción de soluciones que permitan un control unificado y centralizado de múltiples invernaderos puede facilitar la gestión y optimizar los recursos, para asegurar condiciones óptimas de cultivo en todas las instalaciones.
- Mejora en la accesibilidad de la información: desarrollar interfaces más accesibles para los usuarios puede mejorar la capacidad para monitorear y ajustar condiciones de cultivo de manera eficiente desde cualquier ubicación.

## 1.2. Motivación

La motivación para este trabajo surge de los desafíos que enfrentan los agricultores al gestionar invernaderos dispersos geográficamente. La falta de soluciones integrales para el monitoreo y control remoto limita la eficiencia y productividad. Este trabajo busca desarrollar una solución basada en Internet de las Cosas (IoT) que permita un control centralizado y optimizado, alineado con las necesidades de Wentux Tecnoagro [3] y con el interés de aplicar tecnologías IoT para mejorar la gestión agrícola.

## 1.3. Estado del arte

En el mercado actual existen diversas empresas que ofrecen soluciones comerciales diseñadas para optimizar la gestión de invernaderos. Estas herramientas proporcionan una amplia gama de funcionalidades que permiten el control automatizado de parámetros como temperatura, humedad, riego y ventilación. En la tabla 1.1 se muestra una comparativa de algunas de las principales soluciones comerciales disponibles y se destacan sus características.

TABLA 1.1. Comparación de soluciones comerciales.

Empresa	Características
Growcast [4]	Sistema de monitoreo y automatización ambiental con sensores para temperatura, humedad, CO <sub>2</sub> y capacidades de control de riego, iluminación y ventilación. Incluye una aplicación móvil para el monitoreo y control en tiempo real.
Pulse Grow [5]	Sistema especializado en la medición precisa de temperatura, humedad, punto de rocío y déficit de presión de vapor (VPD). Ofrece alertas y ajustes remotos a través de una aplicación móvil.
TrolMaster [6]	Sistema modular que permite el control ambiental, de riego y fertilización. Ofrece un sistema altamente flexible y escalable y permite la integración de múltiples dispositivos para una gestión avanzada de invernaderos.

Estas empresas destacan por su capacidad para integrar tecnología avanzada en el control y monitoreo de invernaderos, lo que facilita una gestión eficiente y adaptada a las necesidades específicas de cada operación agrícola.

## 1.4. Alcance y objetivos

El objetivo principal de este trabajo fue implementar un sistema que permitiera el monitoreo y la gestión remota de invernaderos, para mejorar la eficiencia y la capacidad de respuesta en la gestión de cultivos. Esta propuesta incluyó la implementación de una red de sensores en los invernaderos que recopilan información en tiempo real. Además, se desarrolló una aplicación web progresiva (PWA) para el monitoreo local y un servidor IoT para la gestión remota de datos. Este sistema permitió a los usuarios acceder a la información y controlar los invernaderos desde cualquier lugar, facilitando una gestión eficiente de datos y alarmas.

Dentro del alcance de este trabajo se incluyó:

- El diseño y desarrollo de un protocolo de comunicación basado en ESP-NOW entre los nodos sensores y el sistema embebido central.
- La creación de una PWA para el monitoreo local de los equipos en los invernaderos.
- La implementación de un servidor en la nube para el almacenamiento y gestión de datos recopilados por los sensores.
- El establecimiento de la comunicación cliente-servidor a través del protocolo MQTT para la transmisión de datos desde el sistema embebido central al servidor en la nube.
- La posibilidad de control remoto de los invernaderos y sus dispositivos desde la aplicación web.
- La gestión de alarmas y administración de los datos recibidos por los dispositivos conectados.

El presente trabajo no incluyó:

- El desarrollo del hardware del sistema embebido central, que ya estaba funcionando.
- Mantenimiento y actualizaciones a largo plazo del sistema.

## 1.5. Requerimientos

A continuación se presentan los requerimientos del trabajo:

### 1. Requerimientos funcionales

- a) El sistema debe permitir que los módulos ESP-NOW se comuniquen con el módulo central y puedan intercambiar datos.
- b) Los módulos ESP-NOW deben contar con una configuración de bajo consumo para permitir su uso con baterías.
- c) El usuario deberá tener la capacidad de habilitar o deshabilitar los distintos módulos disponibles.
- d) El módulo central debe ser capaz de auto detectar los módulos que estén dentro de su alcance.
- e) Se implementará un servidor de forma local con el software OpenRemote para el monitoreo remoto de los datos.
- f) El módulo central se conectará al servidor a través del protocolo MQTT.
- g) El módulo central debe proporcionar una interfaz web embebida para acceder a los datos de los sensores y relés del invernadero de manera local.

### 2. Requerimientos de documentación

- a) Se documentarán las bibliotecas para implementar la red de sensores.
- b) Se documentará el proceso general del desarrollo de la PWA y sus bibliotecas y/o frameworks utilizados.
- c) Se documentará el procedimiento de instalación y puesta en marcha del software OpenRemote y sus dependencias en el servidor.

### 3. Requerimientos de la interfaz

- a) La PWA será la interfaz principal para obtener los datos que se recolectan.
- b) La PWA configurará y monitoreará la red.
- c) La PWA requerirá acceso con usuario y contraseña.
- d) La PWA deberá enviar comandos a través de ESP-NOW a los nodos sensores para ejecutar funciones solicitadas por el usuario.

### 4. Requerimientos confidencialidad

- a) Se deberá mantener confidencialidad sobre algunos aspectos de los secretos comerciales, métodos de trabajo y de la información.
- b) Se deberá comprometer a mantener en el futuro dicha conducta.

## Capítulo 2

# Introducción específica

En este capítulo se presentan las tecnologías utilizadas en el desarrollo de este trabajo y se detallan sus características fundamentales de funcionamiento y sus especificaciones técnicas.

### 2.1. Esquema general del sistema

En la figura 2.1 se muestra el diagrama en bloques del sistema. Se pueden observar:

- Red de sensores: recopila datos del entorno del invernadero.
- Nodo central: recibe datos de los sensores ESP-NOW y los envía al servidor a través de MQTT.
- Comunicación MQTT: facilita la transferencia de datos entre el módulo central y el servidor.
- Servidor IoT: almacena y procesa los datos recibidos de los sensores.
- PWA: permite el monitoreo y control del sistema en la red local.
- Red sensores y actuadores: recopilan datos y controlan dispositivos en el invernadero.
- Usuario remoto: accede a los datos recopilados a través del servidor IoT.
- Usuario local: accede a los datos a través de la PWA.

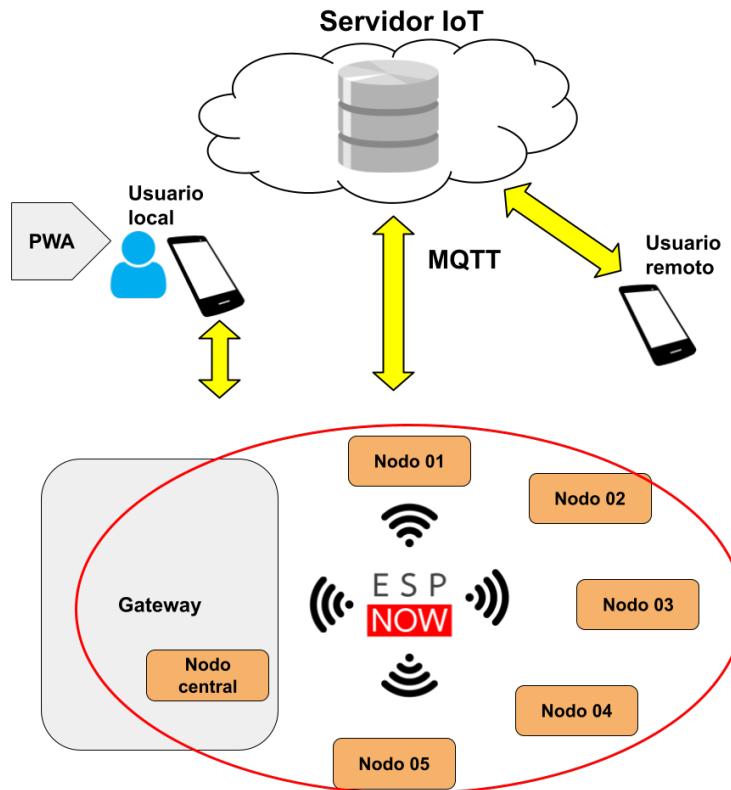


FIGURA 2.1. Diagrama en bloques del sistema.

## 2.2. Tecnologías de hardware

Los componentes de hardware fueron impuestos por la empresa Wentux, por lo que no se tuvo ninguna influencia en la selección del microcontrolador y los sensores.

### 2.2.1. Microcontrolador

Para los nodos sensores y el gateway del sistema, se utilizó la placa de desarrollo ESP32-C3 de Espressif Systems, un microcontrolador eficiente y versátil, ideal para aplicaciones de IoT. Cuenta con un núcleo RISC-V de 32 bits, que combina rendimiento y bajo consumo de energía, lo que optimiza la operación de los dispositivos en el sistema de monitoreo y gestión. Este microcontrolador admite tanto Wi-Fi como Bluetooth 5 (LE) y ofrece múltiples opciones de conectividad inalámbrica. Además de estas tecnologías, soporta el protocolo ESP-NOW, una solución de comunicación inalámbrica propietaria de Espressif [7].

En la figura 2.2 se puede observar el módulo.

El ESP32-C3 también ofrece soporte para actualizaciones OTA (Over-the-Air), lo que facilita la actualización remota del firmware. Además, su amplio conjunto de interfaces de comunicación como UART, I2C, SPI, y ADC permite la integración con diversos sensores y actuadores, necesarios para la operación del sistema en el invernadero.

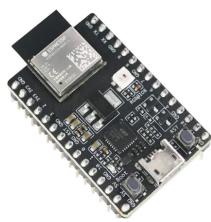


FIGURA 2.2. Módulo ESP32-C3.

La información completa sobre el microcontrolador y sus especificaciones técnicas está disponible en el sitio web oficial de Espressif [8].

### 2.2.2. Sensores

A continuación, se describen brevemente los sensores utilizados en el sistema.

- Sensor LM35 [9]: es un sensor de temperatura analógico que proporciona una salida de voltaje linealmente proporcional a la temperatura. Figura 2.3.

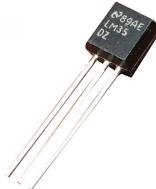


FIGURA 2.3. Sensor LM35.

- Sensor HTU21 [10]: es un sensor digital de humedad relativa y temperatura. Comunica sus lecturas a través de una interfaz I2C. Figura 2.4.



FIGURA 2.4. Sensor HTU21.

- Sensor BME280 [11]: es un sensor ambiental que mide la presión atmosférica, la humedad y la temperatura. Se comunica a través de I2C o SPI. Figura 2.5.



FIGURA 2.5. Sensor BME280.

- Sensor MH-Z19 [12]: es un sensor de dióxido de carbono basado en tecnología infrarroja no dispersiva (NDIR). La información se puede obtener a través de las interfaces UART o PWM. Figura 2.6.

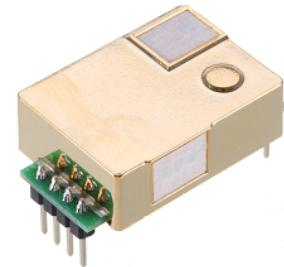


FIGURA 2.6. Sensor MH-Z19.

## 2.3. Tecnologías de firmware

En esta sección se presenta la tecnología utilizada en el trabajo, explicando qué es y las razones de su elección.

### 2.3.1. MicroPython

MicroPython es una versión reducida y optimizada de Python 3, escrita en C, diseñada para ejecutarse en microcontroladores con recursos limitados (como la memoria y la capacidad de procesamiento). A diferencia de otros lenguajes de programación, MicroPython es interpretado, lo que significa que el código no se compila previamente, sino que se interpreta durante la ejecución.

Cuenta con un compilador cruzado, que convierte scripts de Python en bytecode que puede ser ejecutado eficientemente en hardware. Ofrece una biblioteca estándar adaptada a entornos limitados, lo que permite a los desarrolladores escribir código de alto nivel sin recurrir a lenguajes más complejos como C o ensamblador.

Al ser de código abierto, MicroPython está disponible para ser usado y modificado por cualquier persona. Este enfoque abierto, junto con su capacidad de funcionar en hardware limitado, lo hace ideal para la creación de aplicaciones embebidas [13] [14].

#### Elección de MicroPython

MicroPython fue elegido para este trabajo debido a varias ventajas clave:

- Desarrollo ágil: al ser una versión optimizada de Python, permite un desarrollo rápido y eficiente, lo que es esencial para iterar y ajustar la funcionalidad del sistema y agregar nuevas características sin demoras.
- Pruebas y depuración sencillas: la capacidad de ejecutar scripts interactivos facilita la detección y corrección de errores en tiempo real, lo que acelera el proceso de desarrollo y depuración.
- Facilidad de integración con IoT: el ecosistema de MicroPython incluye bibliotecas que permiten integrar fácilmente protocolos de comunicación como ESP-NOW y MQTT, esenciales para la transmisión de datos desde los sensores al gateway y de este al servidor IoT.
- Soporte y comunidad activa: cuenta con una amplia comunidad y documentación [15].

## 2.4. Protocolos de comunicación

En esta sección se presentan los protocolos utilizados en el trabajo.

### 2.4.1. Protocolo ESP-NOW

ESP-NOW es un protocolo de comunicación inalámbrica desarrollado por Espressif para sus dispositivos. A diferencia de los protocolos convencionales que operan en varias capas del modelo OSI, ESP-NOW se basa exclusivamente en la capa de enlace de datos (capa 2), lo que simplifica la comunicación al reducir las cinco capas del modelo OSI a una sola, esto se puede observar en la figura 2.7. Esta arquitectura optimizada le permite ser extremadamente eficiente en términos de recursos, ya que ocupa menos CPU y memoria flash en los dispositivos, crucial para aplicaciones IoT con restricciones de energía y recursos [16].

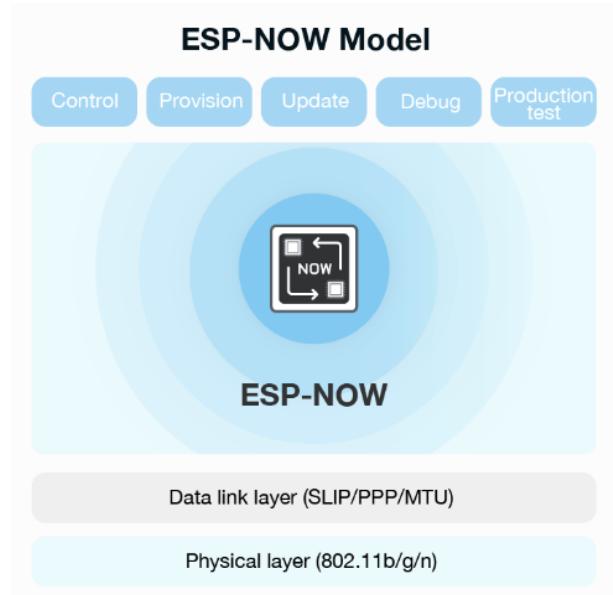


FIGURA 2.7. Modelo ESP-NOW.

ESP-NOW permite la transmisión directa de datos entre dispositivos sin necesidad de una red Wi-Fi o internet. Además, puede funcionar junto con Wi-Fi y Bluetooth LE, y así ofrecer flexibilidad para integrarse en sistemas híbridos.

El protocolo admite transmisión *unicast* y *broadcast*, lo que facilita la comunicación eficiente entre múltiples dispositivos en redes distribuidas. También optimiza el consumo energético, lo que permite a los dispositivos operar en modo de baja potencia, esencial en nodos sensores que operan con baterías y donde la optimización de consumo es crítica. Por último, ESP-NOW soporta encriptación, que garantiza la seguridad en la transmisión de datos entre los nodos [17].

#### 2.4.2. Protocolo MQTT

MQTT (*Message Queuing Telemetry Transport*) es un protocolo de mensajería ligero y eficiente, diseñado para redes con ancho de banda limitado o donde se necesita una comunicación de baja latencia [1] [18]. Este se ejecuta sobre TCP/IP y utiliza una arquitectura de publicación/suscripción, donde:

- *Broker*: es el servidor central que recibe mensajes de los publicadores y los distribuye a los suscriptores. Gestiona todo el flujo de datos.
- Clientes: son dispositivos que actúan como publicadores (envían mensajes al *broker*) o suscriptores (reciben mensajes según el tema suscrito).
- Temas: los mensajes se organizan por temas (*topics*). Los publicadores envían datos asociados a un tema y los suscriptores reciben esos datos si están suscritos al tema correspondiente.

En la figura 2.8 se puede observar la arquitectura del protocolo.

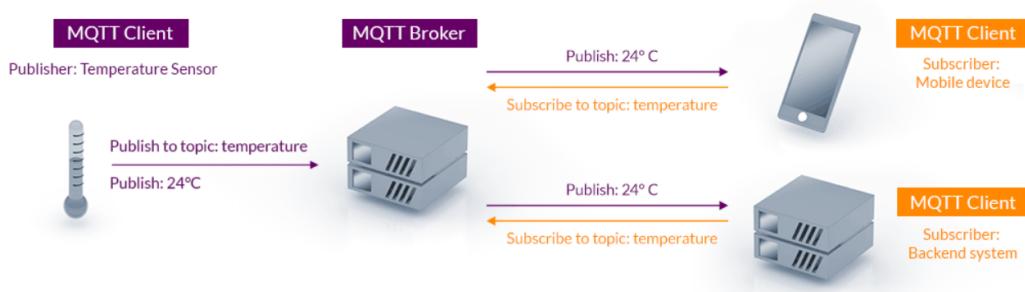


FIGURA 2.8. Arquitectura MQTT [1].

Algunas características claves del protocolo son:

- Retención de mensajes: el *broker* puede almacenar el último mensaje publicado en un tema, y enviarlo a nuevos suscriptores cuando se conectan.
- Calidad de Servicio (QoS): Ofrece tres niveles de garantía de entrega:
  - QoS 0: el mensaje se entrega una sola vez, sin confirmación.
  - QoS 1: el mensaje se entrega al menos una vez.
  - QoS 2: el mensaje se entrega exactamente una vez.
- Conexión persistente: mantiene la conexión abierta con TCP/IP.

- *Last Will and Testament* (LWT): notifica al *broker* si un cliente se desconecta inesperadamente.

## 2.5. Aplicación web progresiva

Una Aplicación Web Progresiva (PWA) es una aplicación web que utiliza tecnologías avanzadas para ofrecer una experiencia similar a la de una aplicación nativa en dispositivos móviles o de escritorio. Aunque se accede a través de un navegador, las PWA pueden instalarse en el dispositivo y ejecutarse como si fueran aplicaciones normales, sin necesidad de descargarlas desde una tienda de aplicaciones. Las PWA combinan las mejores características de las aplicaciones web (accesibilidad desde cualquier navegador y plataforma) con las ventajas de las aplicaciones nativas (rendimiento rápido, funcionamiento sin conexión y capacidad de enviar notificaciones) [19] [20].

Algunas ventajas clave de las PWA son [21]:

- Multiplataforma: funcionan en cualquier dispositivo con un navegador, sin necesidad de adaptar el código a diferentes sistemas operativos.
- Instalables: se pueden agregar directamente desde el navegador, lo que facilita el proceso para el usuario.
- Funcionan sin conexión: utilizan tecnologías como los *Service Workers* [22], que permiten que la aplicación funcione incluso sin conexión a internet, permitiendo el acceso a datos almacenados previamente.
- Notificaciones: las PWA pueden enviar notificaciones al dispositivo, para mantener al usuario informado e interactuar con la aplicación, similar a las aplicaciones nativas.
- Menor consumo de almacenamiento: al no requerir una instalación completa como una aplicación nativa, ocupan menos espacio en el dispositivo.
- Menor costo de desarrollo y mantenimiento: se desarrollan utilizando tecnologías web estándar (HTML, CSS, JavaScript), por lo que no es necesario crear versiones separadas para diferentes sistemas operativos (Android, iOS).

## 2.6. Servidor de internet de las cosas

En esta sección se describe qué es un servidor IoT y OpenRemote.

### 2.6.1. Servidor IoT

Un servidor IoT (*Internet of Things*) es una plataforma o sistema que permite la gestión, almacenamiento y procesamiento de datos generados por dispositivos IoT conectados a una red. Estos dispositivos pueden ser sensores, actuadores, cámaras, entre otros, que recopilan información en tiempo real y la envían al servidor para su análisis o toma de decisiones.

### 2.6.2. OpenRemote

OpenRemote [23] [2] es una plataforma de código abierto para gestionar y controlar redes de dispositivos IoT y sistemas conectados. Facilita la automatización y el manejo de datos en aplicaciones como ciudades inteligentes, edificios automatizados, monitoreo ambiental y gestión de energía.

Características principales de OpenRemote:

- Gestión de dispositivos: permite conectar y controlar dispositivos IoT de diferentes tipos, independientemente del fabricante o del protocolo que utilicen.
- Panel de control personalizable: los usuarios pueden crear interfaces gráficas personalizadas para monitorear y controlar dispositivos en tiempo real.
- Procesamiento de datos: recibe datos de los dispositivos IoT y los procesa, para generar informes, alertas o acciones automatizadas basadas en reglas predefinidas.
- Protocolos de comunicación: soporta una amplia variedad de protocolos de IoT como MQTT, HTTP, entre otros.
- Automatización y reglas: los usuarios pueden definir reglas y flujos de trabajo para automatizar respuestas o acciones en función de los datos recibidos.

En la figura 2.9 se puede observar la arquitectura general del servidor.

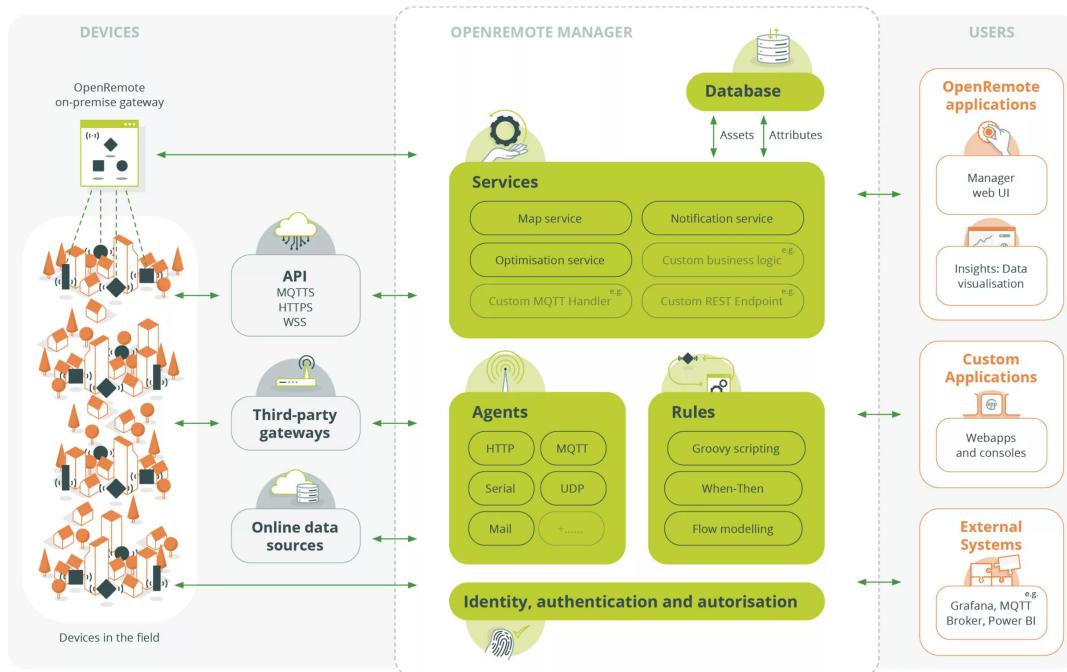


FIGURA 2.9. Arquitectura OpenRemote [2].

## Capítulo 3

# Diseño e implementación

En este capítulo se describen las soluciones planteadas al problema resuelto en este trabajo.

### 3.1. Arquitectura del sistema

En la figura 3.1 se puede observar un esquema simple del sistema, donde se tiene:

- Nodos sensores.
- Nodo central.
- Aplicacion web progresiva.
- Broker.
- Servidor IoT.

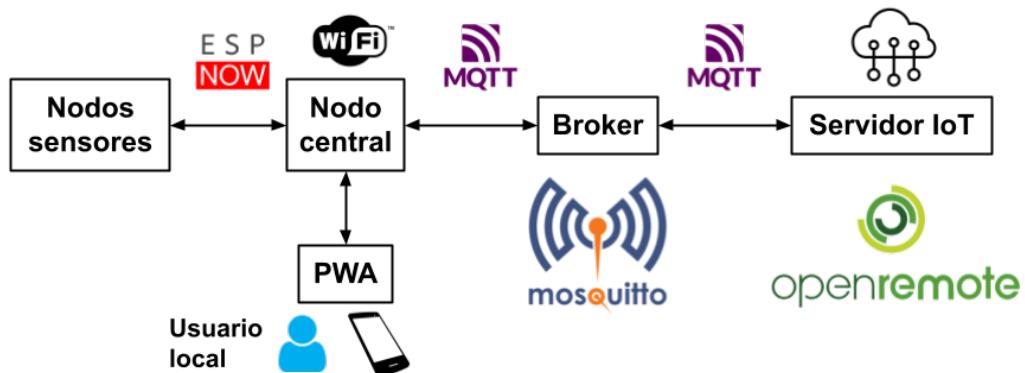


FIGURA 3.1. Esquema del sistema.

En esta sección se describen los principales módulos que conforman el sistema, detallando su función y cómo interactúan entre sí.

#### 3.1.1. Nodos sensores

Están conformados por un módulo ESP32-C3 y un sensor. Esta es la capa de percepción, ya que los nodos son responsables de captar la información del entorno. Los datos recopilados son enviados al nodo central (gateway) a través del protocolo de comunicación ESP-NOW. En este contexto, también se incluye un módulo

específico para gestionar los relés, cuyo estado puede ser consultado y modificado, por lo que se establece una *comunicación bidireccional* con el *gateway*. Sin embargo, en el caso de los sensores, la *comunicación es unidireccional*, ya que solo envían datos.

### 3.1.2. Nodo central

Como se mencionó, este nodo se comunica con los sensores a través de ESP-NOW. Actúa como la capa de transporte, encargándose de transferir los datos capturados por la capa de percepción. Además, el *gateway* integra una interfaz web embedida para visualizar los datos de los sensores y el estado de los relés. Esta interfaz está implementada utilizando un socket y, en caso de ausencia de conexión a internet, genera un *Access Point* (AP) local al cual el usuario puede conectarse para monitorear el estado del invernadero. El nodo central se conecta a internet mediante Wi-Fi para enviar los datos a un *broker MQTT* (Mosquitto), empleando certificados TLS para asegurar la transmisión.

### 3.1.3. Servidor IoT

Este componente corresponde a la capa de procesamiento, donde se almacenan, analizan y gestionan los datos enviados desde el nodo central. Para esta función, se utilizará *OpenRemote* como plataforma de servidor IoT, la cual facilita la integración de dispositivos y la gestión de datos. El servidor se conecta al *broker MQTT* para recibir la información de los sensores y los relés. Además de almacenar y visualizar los datos en tiempo real, el servidor permite generar informes, configurar alertas y ejecutar acciones automatizadas basadas en reglas predefinidas, como la activación de relés o notificaciones ante condiciones anómalas.

## 3.2. Desarrollo del firmware

El firmware del trabajo fue desarrollado utilizando el lenguaje de programación python, específicamente bajo el *framework micropython*. Para su implementación, se empleó el entorno de desarrollo Visual Studio Code, que facilitó la edición y depuración del código.

En esta sección se presentan los diagramas de flujo que describen los procesos clave de los nodos sensores, el módulo de relés y el nodo central. Estos diagramas ilustran la lógica implementada en el firmware y cómo se gestiona la comunicación eficiente entre sensores, actuadores, la interfaz web y el servidor IoT.

### 3.2.1. Nodos sensores y módulo relés

#### Nodos sensores

En la figura 3.2 se presenta el diagrama de flujo de los nodos sensores.

Cada sensor tendrá un script personalizado, ya que son diferentes entre sí, lo que implica que la forma de obtener las lecturas de los datos variará en cada caso. Sin embargo, el procedimiento para enviar los datos será el mismo para todos.

Primero, se inicializa y configura ESP-NOW para enviar mensajes en modo *broadcast* (difusión) a todos los dispositivos cercanos, sin la necesidad de conocer sus direcciones MAC individuales. A continuación, el nodo busca el canal en el que

se encuentra el *gateway*. Una vez encontrado, se envía un mensaje de verificación para confirmar que el canal es el correcto. Si no se recibe respuesta, el nodo se reinicia, comenzando el proceso nuevamente.

Cuando el canal es verificado, se toma la lectura de los datos del sensor y se envía esta información al *gateway* vía ESP-NOW, de forma encriptada, utilizando una librería específica de micropython llamada *cryptolib* [15].

Una vez que la información ha sido enviada, el dispositivo entra en *modo DeepSleep* (sueño profundo) durante un tiempo determinado, con el fin de ahorrar energía. Al despertar, el módulo se reinicia como si hubiese sido encendido nuevamente.

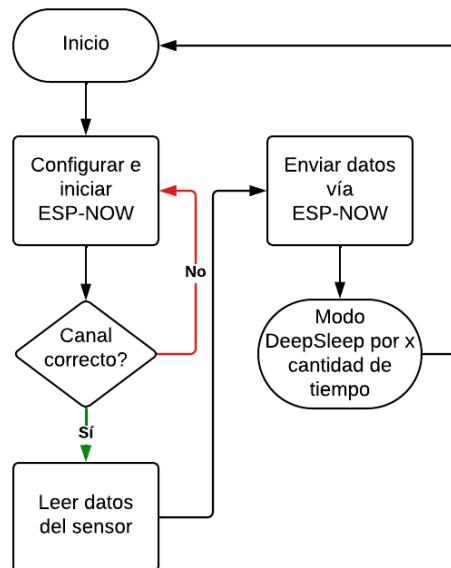


FIGURA 3.2. Diagrama de flujo de los nodos sensores.

### Módulo relés

El nodo encargado de los relés sigue un funcionamiento similar al de los nodos sensores en cuanto a la configuración inicial de ESP-NOW, por lo que no es necesario volver a detallar ese proceso. La principal diferencia radica en que, en lugar de tomar lecturas de un sensor, este nodo inicializa los relés y envía el estado en el que se encuentran al *gateway*.

Una vez enviados los estados, el nodo queda a la espera de recibir instrucciones para modificar alguno de los relés. Si se recibe una solicitud para cambiar el estado de uno o más relés, el nodo ejecuta el cambio y, a continuación, envía nuevamente la información actualizada al *gateway*. Al igual que en los nodos sensores, la información se envía de forma encriptada para garantizar la seguridad de los datos transmitidos. De la misma manera, cualquier información que llega al nodo, como instrucciones para cambiar el estado de los relés, es desencriptada antes de ser procesada.

Este nodo no entra en *modo DeepSleep*, ya que es necesario garantizar que los relés respondan de manera inmediata a cualquier instrucción de control que pueda recibirse.

En la figura 3.3 se puede observar el diagrama de flujo del módulo relés.

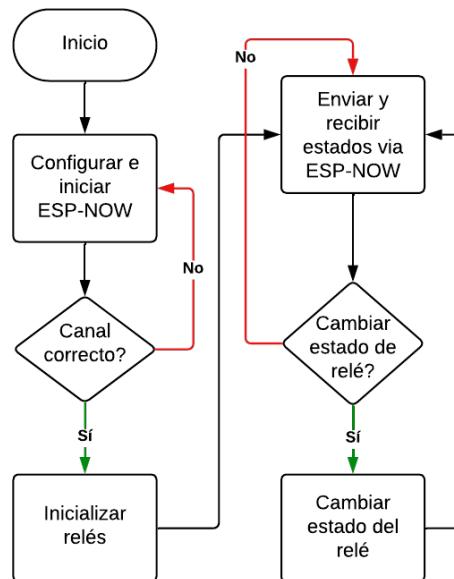


FIGURA 3.3. Diagrama de flujo del módulo relés.

### 3.2.2. Nodo Central

El *gateway* puede iniciar en dos modos diferentes: *Modo Cliente* (CL) o *Modo Access Point* (AP), dependiendo de la configuración cargada en el dispositivo. Una vez determinado el modo de operación, se configura e inicia la comunicación ESP-NOW en *modo de difusión*.

- Modo AP: en este caso, el *gateway* actúa como un punto de acceso local, lo que significa que no está conectado a internet. Se inicia el servidor HTTP para proporcionar una interfaz web accesible a través de una red local, permitiendo al usuario monitorear y gestionar el sistema directamente sin necesidad de conexión externa.
- Modo CL: al iniciar en este modo significa que está conectado a internet a través de una red Wi-Fi. Se establece una conexión segura con un *broker* MQTT. Para ello, se utilizan un nombre de usuario, una contraseña y certificados TLS para asegurar la comunicación. Además, el *gateway* se suscribe a un tópico específico del *broker* MQTT, lo que le permite recibir mensajes relacionados con los sensores. Por último se inicia el servidor HTTP, lo que permite el acceso a la interfaz web embebida para monitorear y gestionar el sistema desde cualquier lugar con acceso a internet.

En la figura 3.4 se puede observar el diagrama de flujo general del nodo central.

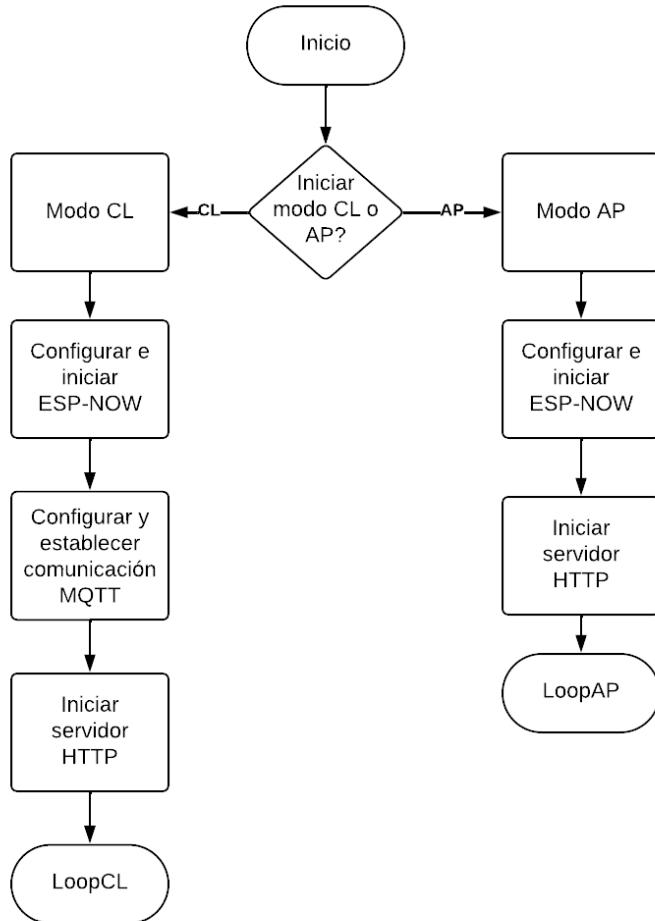


FIGURA 3.4. Diagrama de flujo del nodo central.

### Loop AP

En la figura 3.5 se puede observar el diagrama de flujo del loopAP.

Una vez que el *gateway* se ha iniciado en Modo AP y se ha configurado correctamente la comunicación ESP-NOW, entra en un bucle continuo de monitoreo y control, conocido como LoopAP. En este modo el *gateway* realiza las siguientes acciones:

- Monitoreo de nodos: el *gateway* espera recibir datos de los nodos sensores y del módulo de relés a través de ESP-NOW. Al recibir información, ya sea de sensores o del estado de los relés, estos datos se envían y muestran en la PWA en tiempo real.
- Verificación de cambios en la PWA: si no se recibe información desde los nodos, el *gateway* revisa si hubo un cambio en el estado de los relés en la PWA. Si detecta un cambio, envía la actualización al nodo correspondiente mediante ESP-NOW.
- Reinicio del ciclo: luego de procesar cualquier evento, el *gateway* regresa al inicio del ciclo, quedando en espera de nuevos datos o cambios.

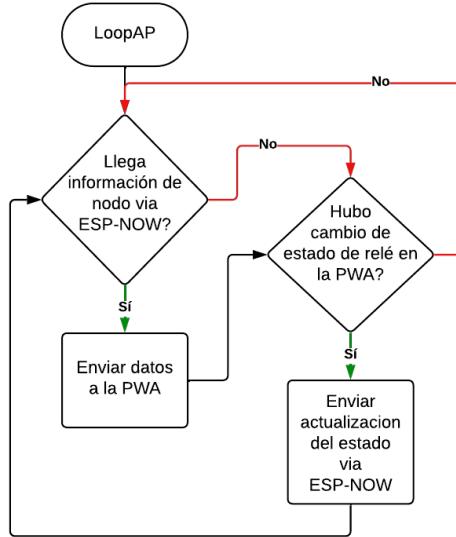


FIGURA 3.5. Diagrama de flujo del loop AP.

### Loop CL

En este modo, los datos de los sensores no solo se muestran localmente en la PWA, sino que también se publican en el *broker* MQTT, lo que permite que sean almacenados y procesados remotamente. Esto garantiza una mayor capacidad de monitoreo y análisis. Otra diferencia importante es la sincronización con el servidor IoT. Mientras que en el LoopAP los cambios en el estado de los relés solo se controlan desde la PWA, en el LoopCL también se monitorea si el estado de los relés ha sido modificado desde el servidor IoT. Esto posibilita la gestión remota de los dispositivos, permitiendo que el usuario controle y supervise el invernadero desde cualquier lugar con acceso a internet, una funcionalidad que no está disponible en el modo AP.

En la figura 3.5 se presenta el diagrama de flujo del loopCL.

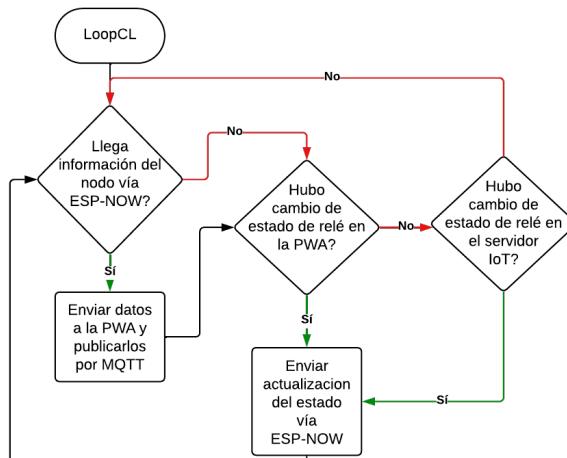


FIGURA 3.6. Diagrama de flujo del loop CL.

### 3.3. Desarrollo de la aplicación web progresiva

La aplicación web progresiva (PWA) fue desarrollada utilizando tecnologías web estándar como HTML [24], CSS [25] y JavaScript [26]. Su objetivo principal es proporcionar a los usuarios una interfaz intuitiva y accesible para visualizar la información generada por los sensores y monitorear los estados de los relés. Además, la PWA permite a los usuarios interactuar con los relés, facilitando el cambio de estado de estos dispositivos en tiempo real. También se incluye la funcionalidad para editar la configuración del *gateway*, ofreciendo un control integral y eficiente del sistema.

#### 3.3.1. Login de la PWA

En la figura 3.7 se muestra la pantalla de inicio de sesión, donde el usuario debe ingresar su nombre de usuario y contraseña para acceder al sistema.

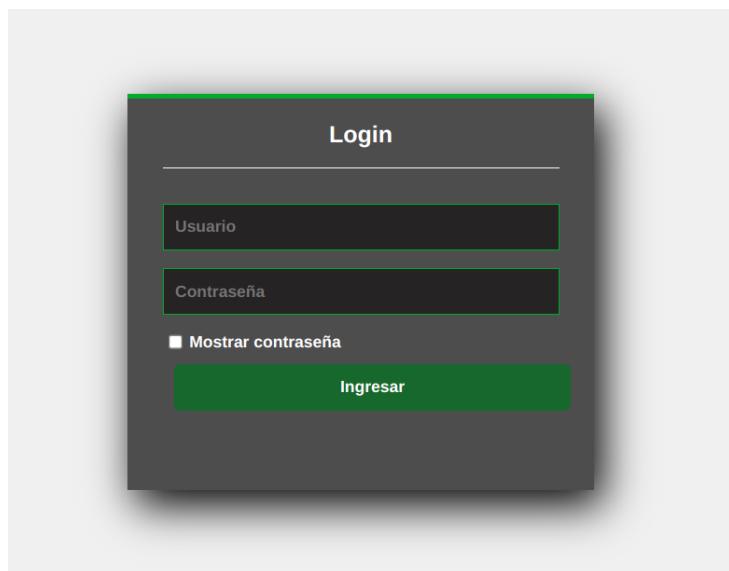


FIGURA 3.7. Pantalla de login.

El sistema cuenta con un método de validación basado en *localStorage*, lo que garantiza que, incluso si se conoce la URL de acceso a las distintas pantallas, no se puede ingresar sin haber iniciado sesión previamente.

En caso de que se cometiera un error al ingresar las credenciales, el sistema desplegará un mensaje indicando que son incorrectas, tal como se muestra en la figura 3.8.

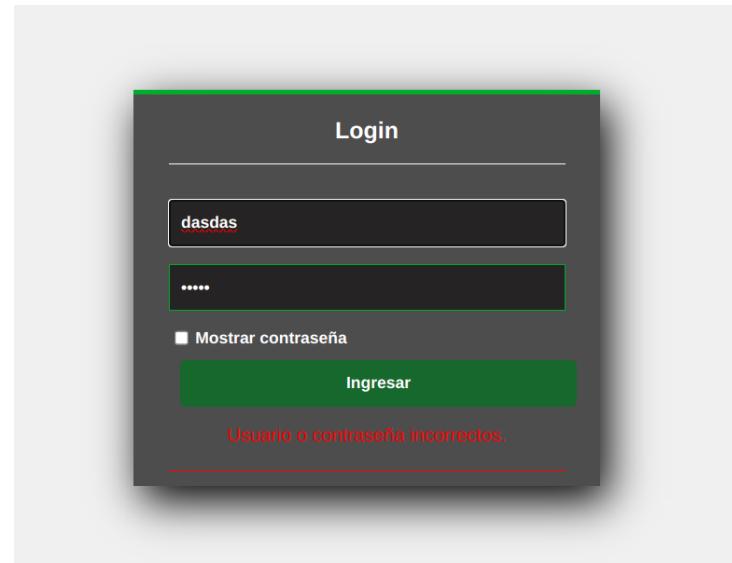


FIGURA 3.8. Pantalla de login con datos incorrectos.

### 3.3.2. Pagina principal

La figura 3.9 muestra la pantalla principal de la aplicación web progresiva (PWA) sin ningún sensor ni relé cargado. En esta pantalla, los usuarios pueden acceder a dos funciones principales mediante botones ubicados en la parte superior derecha: uno para navegar a la página de configuración y otro para cerrar sesión. La página también está preparada para desplegar la información de los sensores y relés, que aparecerán en las secciones correspondientes una vez se carguen los datos.

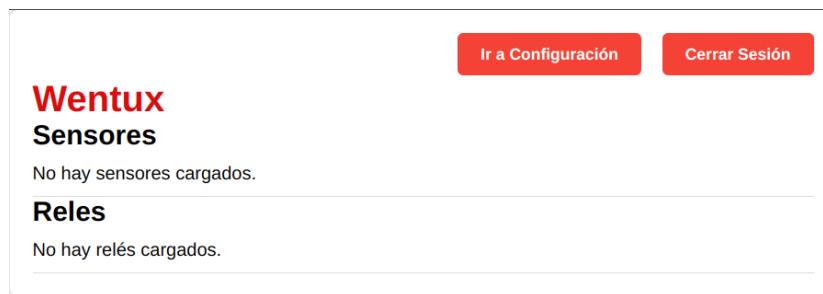


FIGURA 3.9. Pantalla principal.

En la figura 3.10 se muestra la pantalla principal de la PWA con los sensores y relés ya cargados. En esta interfaz, los usuarios pueden visualizar en tiempo real la información de los sensores y los relés conectados al sistema.

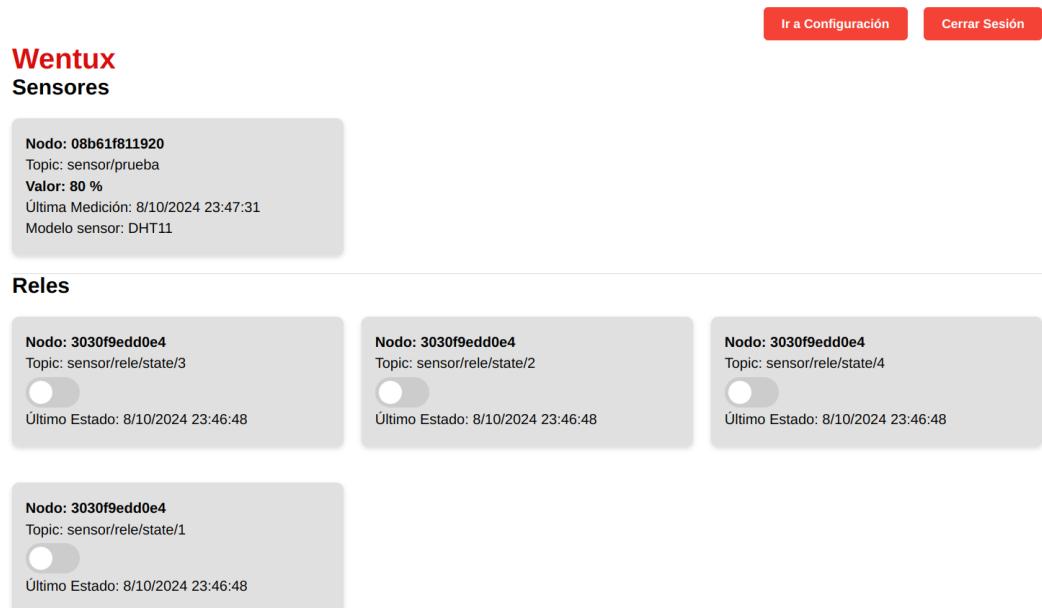


FIGURA 3.10. Pantalla principal.

Cada sensor presenta los siguientes atributos:

- Nodo: este campo muestra la dirección MAC del nodo al que está conectado el sensor.
- Topic: define la variable que está siendo medida por el sensor. Puede variar según el tipo de sensor, por ejemplo, sensor/temp para la temperatura, sensor/hum para la humedad, o sensor/co2 para la concentración de CO2, entre otros.
- Valor: indica el valor actual de la variable que el sensor está midiendo.
- Última medición: muestra la fecha y hora de la última vez que se recibió un dato válido desde el sensor.
- Modelo sensor: especifica el nombre o tipo del sensor utilizado.

En la sección de relés, la PWA permite a los usuarios interactuar y controlar los dispositivos conectados:

- Nodo: al igual que los sensores, este campo muestra la dirección MAC del nodo al que está asociado el relé.
- Topic: define el relé específico que se está controlando, utilizando un formato del tipo sensor/rele/state/X, donde X representa el número del relé.
- Botón on/off: cada relé tiene un interruptor visual que permite al usuario encender o apagar el dispositivo manualmente.
- Último estado: registra la fecha y hora del último cambio en el estado del relé.

### 3.3.3. Pantalla de configuración

La página de configuración permite al usuario ajustar los parámetros clave del sistema para su correcto funcionamiento en diferentes modos operativos, dependiendo de si el dispositivo actúa como Cliente (CL) o como Punto de Acceso (AP). La configuración se realiza a través de un formulario con varios campos que permiten personalizar aspectos como la conexión Wi-Fi, el broker MQTT, y otros detalles del sistema embebido.

En las figuras 3.11 y 3.12 se puede observar la pantalla de configuración donde tenemos:

Modos de Operación:

- Cliente (CL): en este modo, el dispositivo se conecta a una red Wi-Fi externa.
- Punto de Acceso (AP): en este modo, el dispositivo no tiene acceso a Internet.

Campos principales:

- SSID y Contraseña: estos campos permiten configurar la red Wi-Fi a la que se conectará el dispositivo en modo Cliente.
- SSID y Contraseña del AP: estos datos se utilizan cuando el dispositivo está en modo Punto de Acceso, permitiendo a otros dispositivos conectarse directamente a este.
- Cliente ID, MQTT Broker, Usuario y Contraseña MQTT: son los parámetros necesarios para configurar la comunicación con el servidor de MQTT, que es fundamental para transmitir los datos de los sensores y controlar los relés remotamente.
- Puerto: especifica el puerto utilizado para la conexión MQTT.

Configuración de Fecha y Hora:

- cuando el dispositivo está en modo Cliente sincroniza automáticamente la fecha y la hora a través de Internet, por lo que los campos de Fecha y Hora en la página de configuración están deshabilitados.
- Cuando el dispositivo está en modo AP, es necesario ingresar manualmente la fecha y la hora para asegurar que el sistema embebido, que no tiene acceso a Internet, opere en el tiempo correcto.

Botones:

- Guardar Configuración: este botón guarda los cambios realizados en los parámetros del sistema. Al hacer clic, se edita el archivo de configuración en el sistema y luego se reinicia el dispositivo para aplicar los nuevos parámetros. Este reinicio asegura que el sistema funcione con la configuración más reciente sin necesidad de intervención manual adicional.
- Volver a Inicio: este botón redirige al usuario de vuelta a la pantalla principal de la aplicación, donde puede visualizar los datos de los sensores y relés configurados en el sistema.

Con esta configuración flexible, el sistema puede adaptarse tanto a escenarios con conexión a Internet como a entornos sin conectividad, lo que lo hace ideal para ser

implementado en diversos contextos, como invernaderos o áreas rurales donde la conectividad puede ser limitada.

## Página de Configuración

Modo de Operación:

Punto de Acceso (AP)

SSID:

quepasapatejode

Contraseña:

losvilla08

SSID del AP:

ESP32\_AP

Contraseña del AP:

12345678

Cliente ID:

dispositivo1

MQTT Broker:

192.168.1.11

FIGURA 3.11. Pantalla de configuración.

MQTT Usuario:

wentux

MQTT Contraseña:

wentux123

Puerto:

8884

Fecha:

08/10/2024

Hora:

23:46

FIGURA 3.12. Pantalla de configuración.

### 3.4. Implementación del servidor IoT

Para la implementación del servidor IoT se utilizó OpenRemote, el cual se despliega de manera local mediante el uso de Docker [27]. Docker permite levantar de forma rápida y eficiente el entorno del servidor, facilitando la gestión de contenedores y asegurando que todo el sistema funcione de manera óptima y sin conflictos de dependencias. Este enfoque permite que el servidor esté operativo de manera ágil y estandarizada, sin la necesidad de configuraciones manuales complejas.

Se explicará el proceso de inicio de sesión, la configuración del agente MQTT para conectar con el broker, la creación y configuración de los objetos correspondientes a los sensores y relés mediante Thing Asset, la definición y uso de reglas para automatizar acciones y, finalmente, la configuración del dashboard para la visualización de los datos.

#### 3.4.1. Login

En la figura 3.13 se presenta la pantalla de inicio de sesión, donde el usuario debe ingresar su nombre de usuario o correo electrónico, junto con la contraseña, para acceder al sistema. Si las credenciales ingresadas son incorrectas, el sistema mostrará un mensaje de advertencia indicando el error, como se ilustra en la figura 3.14

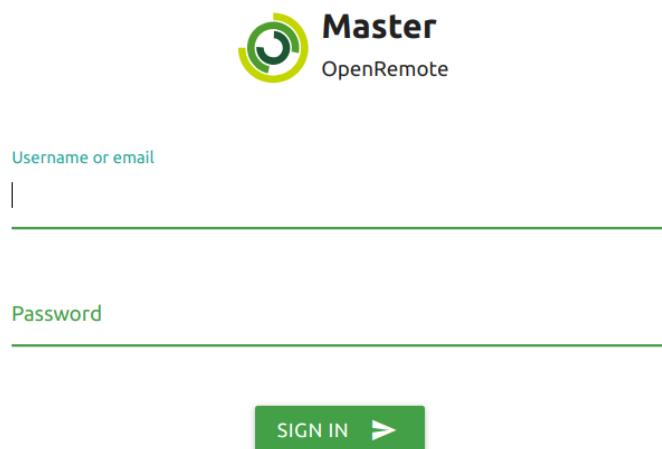


FIGURA 3.13. Pantalla de login de OpenRemote.

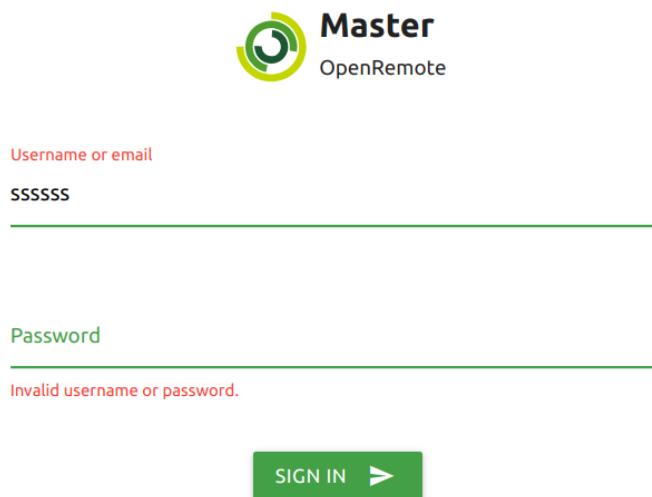


FIGURA 3.14. Pantalla de login de OpenRemote con datos incorrectos.

### 3.4.2. Agente MQTT y activos

Se explicará cómo usar el Agente MQTT en OpenRemote y cómo utilizar los activos (assets) para gestionar la información de los sensores y relés a través de la plataforma.

#### Agente MQTT

OpenRemote cuenta con un Agente MQTT (Cliente) que permite conectar el servidor a un broker MQTT externo. Para establecer esta conexión, es necesario configurar el host del broker, el Cliente ID, el puerto de conexión, así como las credenciales de acceso, que incluyen el usuario y la contraseña. Una vez configurados estos parámetros, el agente se encarga de gestionar los datos que se transmiten entre los sensores y el servidor. Esta integración garantiza que la información fluya de manera eficiente y pueda ser utilizada para la visualización y control de dispositivos a través de la plataforma.

En la figura 3.15 se puede observar la pantalla de configuración del agente mqtt.

> agentStatus	Connection status	CONNECTED
> clientId	Text	cliente123
> host	Host or IP address	192.168.1.11
> port	TCP IP port number	1883
> usernamePassword	Username and password	<pre>{   "username": "wentux",   "password": "wentux123" }</pre>

FIGURA 3.15. Configuración del agente mqtt.

### Activos (Assets)

En OpenRemote, los activos representan los objetos correspondientes a los sensores y relés del sistema. Cada activo contiene atributos que reflejan diferentes tipos de datos, como valores de temperatura, humedad o estado de los relés. Para vincular estos atributos al agente MQTT, se utiliza la opción de Agent Link, que permite asociar un atributo a un tópico específico de publicación o suscripción en el broker MQTT, asegurando que los datos recibidos o enviados se manejen correctamente. Esta configuración facilita la integración de los dispositivos físicos con el sistema de gestión IoT. Esto se ilustra en la figura 3.16, donde se muestra el nombre del activo, que en este caso es 'Rele', y su tipo, que es 'boolean'.

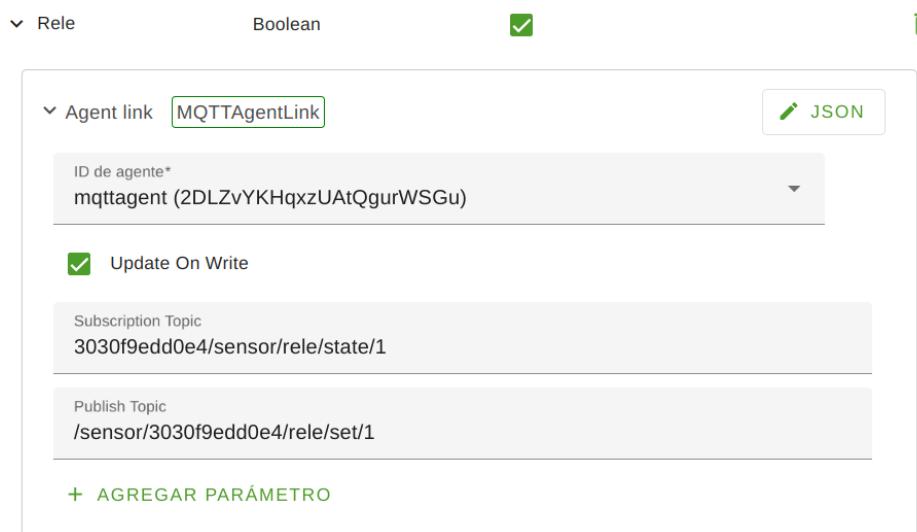


FIGURA 3.16. Configuración del activo.

En la figura 3.17 se muestra la pantalla principal de los activos. En la parte izquierda se puede observar el *mqttagent* junto con los activos asociados. También se presenta la información del sensor 1, donde los campos de notas y posición están incorporados al activo por defecto y no pueden deshabilitarse. Se han agregado los atributos 'rele', 'rele 2' y 'temp', que están suscritos al broker MQTT para recibir información. En el caso de los relés, son checkboxes que funcionan con lógica booleana y pueden publicar su estado en caso de que cambien manualmente o a través de reglas.

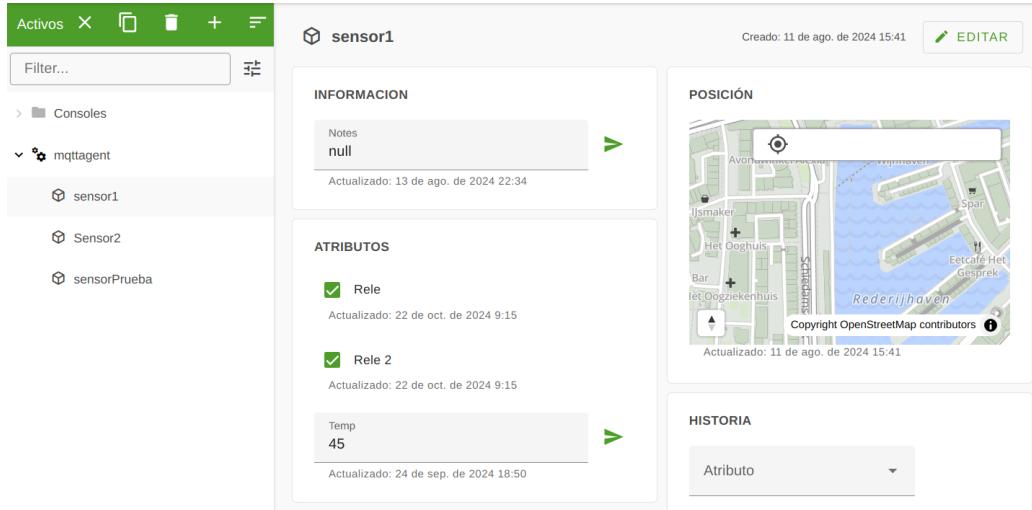


FIGURA 3.17. Pantalla principal de los activos.

### 3.4.3. Reglas

OpenRemote incluye un motor de reglas que permite a los usuarios automatizar tareas de diversas maneras. Un tipo de regla que destaca es la regla 'Cuando-Entonces', que facilita la creación de acciones basadas en eventos de forma sencilla.

Con las reglas 'Cuando-Entonces', podemos establecer una condición que, al cumplirse, activa una acción. Por ejemplo, en la figura 3.18, se muestra el activo sensor1, donde si el atributo 'temp' supera los 40 grados, entonces el relé del activo sensor1 se activa mientras que el relé 2 se apaga.



FIGURA 3.18. Pantalla principal de las reglas.

### 3.4.4. Dashboard

OpenRemote ofrece una funcionalidad de *dashboard* totalmente configurable, que permite a los usuarios visualizar los datos de sus sensores y dispositivos de manera gráfica y sencilla. Este panel puede personalizarse utilizando diversos widgets según las necesidades del trabajo. En la figura 3.19 se muestra un ejemplo de *dashboard* donde se ha configurado un gráfico de líneas para monitorear la temperatura registrada por un sensor, así como la visualización del estado de dos relés. Esta flexibilidad en la configuración permite adaptar la interfaz a distintos casos de uso, ofreciendo una solución visual intuitiva para la monitorización y el control de dispositivos.

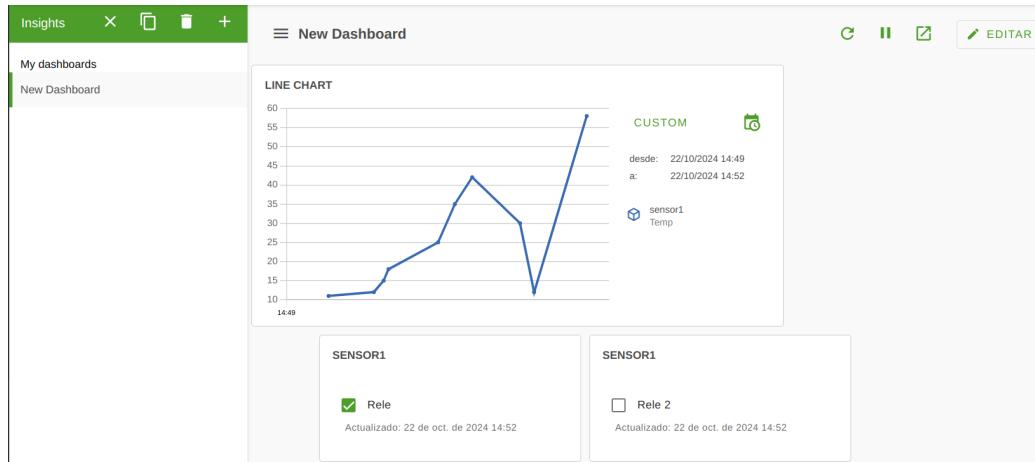


FIGURA 3.19. Pantalla principal del dashboard.

## Capítulo 4

# Ensayos y resultados

Este capítulo se presentan los ensayos realizados para verificar la funcionalidad de los componentes individuales y del sistema completo, junto con el análisis de los resultados obtenidos.

### 4.1. Banco de pruebas

En esta sección se describe el entorno de pruebas que se creó para simular y evaluar el rendimiento del sistema en condiciones controladas. El objetivo del banco de pruebas fue verificar la correcta comunicación entre los nodos sensores, el módulo de relés y el gateway, asegurando que los datos fluyeran de manera precisa y que el sistema reaccionara adecuadamente a los comandos.

Para recrear este entorno, se utilizaron dos nodos sensores, el sensor LM35 y el sensor MHZ19. Además, se integró un nodo con un módulo de relés, cuyos estados fueron simulados mediante leds, lo que facilitó la visualización de los cambios.

La transmisión de los datos desde los nodos hacia el gateway se realizó mediante el protocolo ESP-NOW, lo que permitió una comunicación inalámbrica eficiente entre los dispositivos. Los nodos sensores capturaron la información ambiental y la enviaron al gateway, mientras que el nodo de relés reportó el estado actual de los relés utilizando también ESP-NOW.

El gateway, estuvo conectado a una computadora para monitorear la recepción de los datos a través de la consola, validando que toda la información llegara correctamente. Este enfoque permitió realizar un seguimiento detallado del funcionamiento de los componentes y su interacción dentro del sistema.

En la figura 4.1 se puede visualizar los componentes utilizados para el banco de pruebas.

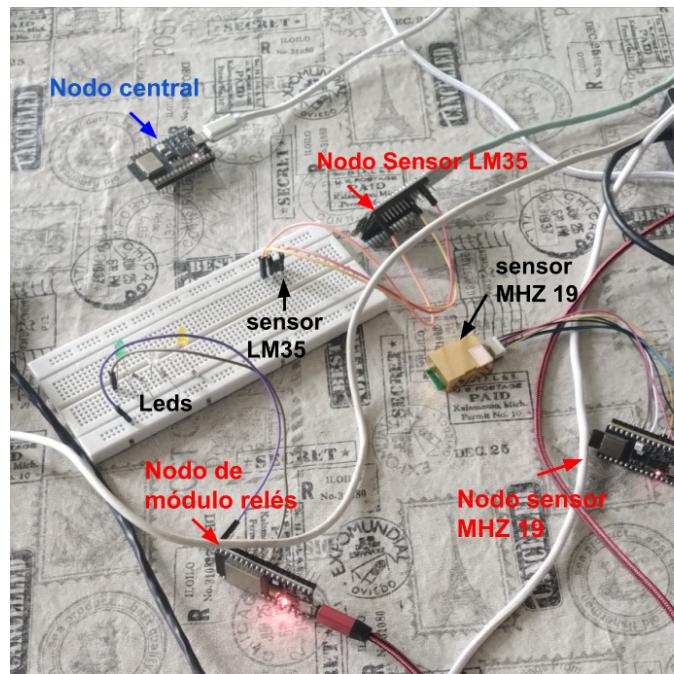


FIGURA 4.1. Prototipo utilizado para las pruebas.

## 4.2. Prueba de componentes

Para asegurar el correcto funcionamiento de los sensores y del módulo de relés, se llevaron a cabo diversas pruebas de hardware y software. Estas pruebas fueron esenciales para validar que los sensores capturaran los datos correctamente y que el módulo de relés respondiera adecuadamente a los comandos recibidos, garantizando así una comunicación precisa entre estos componentes dentro del sistema.

### 4.2.1. Prueba de sensores

Se realizaron pruebas para verificar la precisión de los sensores LM35 y MHZ19 en la recolección de datos de temperatura y CO<sub>2</sub>. Los ensayos incluyeron:

- Prueba de medición individual: se conectaron los sensores a un nodo de manera individual y se comprobó que los valores de temperatura y CO<sub>2</sub> obtenidos fueran coherentes con las condiciones ambientales actuales.
- Prueba de transmisión de datos: una vez que los sensores recolectaron los datos, se probó la transmisión de estos al gateway a través del protocolo ESP-NOW. Se monitoreó la consola del gateway para verificar que los datos llegaran sin pérdidas o alteraciones en el tiempo esperado.

### Resultados

Para el sensor LM35, se aplicó un secador de pelo sobre el sensor para generar variaciones de temperatura. Los resultados de esta prueba se muestran en la figura 4.2, donde se puede observar claramente el cambio en los valores de temperatura registrados.

```
Datos desencriptados: {"modelo": "LM35", "value": 29.25, "topic": "sensor/temp"}
Topic general: 58cf79e38408/sensor/temp, Value: 29.25
Datos desencriptados: {"modelo": "LM35", "value": 29.33, "topic": "sensor/temp"}
Topic general: 58cf79e38408/sensor/temp, Value: 29.33
Datos desencriptados: {"modelo": "LM35", "value": 29.49, "topic": "sensor/temp"}
Topic general: 58cf79e38408/sensor/temp, Value: 29.49
Datos desencriptados: {"modelo": "LM35", "value": 30.54, "topic": "sensor/temp"}
Topic general: 58cf79e38408/sensor/temp, Value: 30.54
Datos desencriptados: {"modelo": "LM35", "value": 30.38, "topic": "sensor/temp"}
Topic general: 58cf79e38408/sensor/temp, Value: 30.38
Datos desencriptados: {"modelo": "LM35", "value": 31.91, "topic": "sensor/temp"}
Topic general: 58cf79e38408/sensor/temp, Value: 31.91
Datos desencriptados: {"modelo": "LM35", "value": 30.95, "topic": "sensor/temp"}
Topic general: 58cf79e38408/sensor/temp, Value: 30.95
Datos desencriptados: {"modelo": "LM35", "value": 30.38, "topic": "sensor/temp"}
Topic general: 58cf79e38408/sensor/temp, Value: 30.38
```

FIGURA 4.2. Datos del sensor LM35.

En el caso del sensor MHZ19, se generó una variación en la concentración de CO<sub>2</sub> mediante una reacción química entre vinagre y bicarbonato de sodio, que liberó CO<sub>2</sub>. Los datos de esta variación se pueden ver en la figura 4.3, con un comportamiento coherente ante el incremento de CO<sub>2</sub>.

```
Datos desencriptados: {"modelo": "MHZ19C", "value": 454, "topic": "sensor/co2"}
Topic general: 1091a83a7f68/sensor/co2, Value: 454
Datos desencriptados: {"modelo": "MHZ19C", "value": 1891, "topic": "sensor/co2"}
Topic general: 1091a83a7f68/sensor/co2, Value: 1891
Datos desencriptados: {"modelo": "MHZ19C", "value": 1953, "topic": "sensor/co2"}
Topic general: 1091a83a7f68/sensor/co2, Value: 1953
Datos desencriptados: {"modelo": "MHZ19C", "value": 2013, "topic": "sensor/co2"}
Topic general: 1091a83a7f68/sensor/co2, Value: 2013
Datos desencriptados: {"modelo": "MHZ19C", "value": 741, "topic": "sensor/co2"}
Topic general: 1091a83a7f68/sensor/co2, Value: 741
Datos desencriptados: {"modelo": "MHZ19C", "value": 723, "topic": "sensor/co2"}
Topic general: 1091a83a7f68/sensor/co2, Value: 723
Datos desencriptados: {"modelo": "MHZ19C", "value": 685, "topic": "sensor/co2"}
Topic general: 1091a83a7f68/sensor/co2, Value: 685
Datos desencriptados: {"modelo": "MHZ19C", "value": 483, "topic": "sensor/co2"}
Topic general: 1091a83a7f68/sensor/co2, Value: 483
```

FIGURA 4.3. Datos del sensor MHZ19.

En ambas pruebas, los datos recolectados por los sensores fueron transmitidos exitosamente al gateway, sin pérdidas ni errores, confirmando la estabilidad y confiabilidad del sistema en la transmisión de la información.

#### 4.2.2. Prueba del módulo de relés

El módulo de relés, simulado con LEDs, fue evaluado en dos aspectos clave:

- Prueba de activación/desactivación: se configuró el sistema para que el gateway enviara comandos de activación y desactivación al nodo de relés a través de ESP-NOW. Se observó que los LEDs respondían correctamente a los comandos, encendiéndose y apagándose de manera precisa.
- Prueba de sincronización: para asegurar que los estados de los relés se mantuvieran sincronizados con el sistema, se verificó que los cambios en los LEDs correspondieran con los comandos enviados por el gateway sin retrasos significativos.

#### Resultados

En la figura 4.4, se muestra el proceso de envío de comandos desde el gateway hacia el nodo de relés. En primer lugar, el gateway envió un comando para activar

el relé, lo que fue recibido correctamente por el nodo, encendiendo el led azul que simula el funcionamiento del relé. Posteriormente, se envió el comando de desactivación por lo que el nodo apagó el led correspondiente.

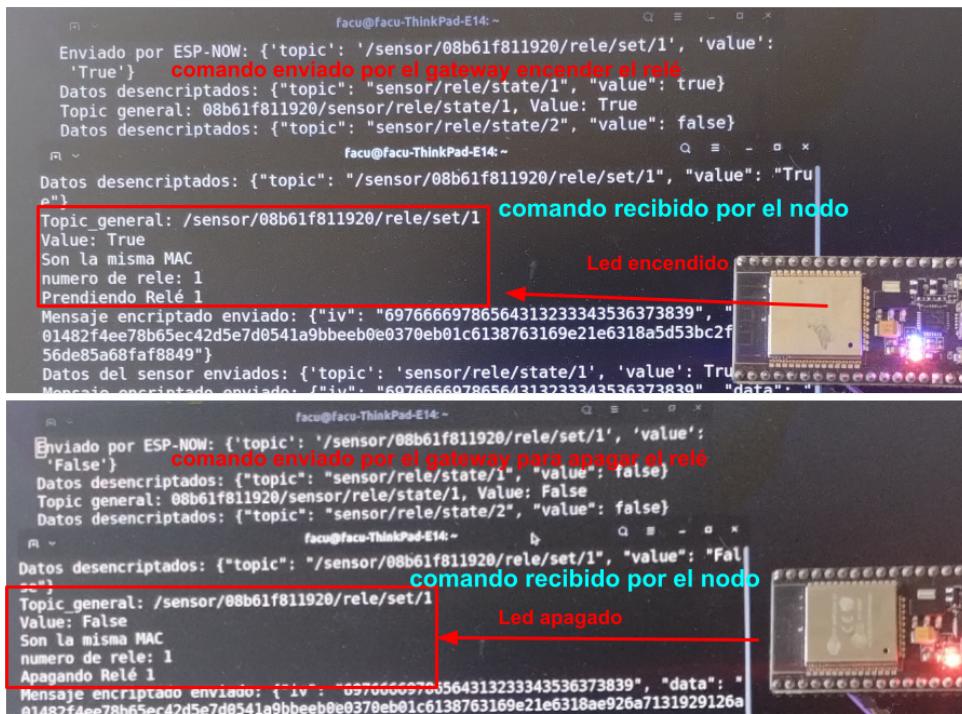


FIGURA 4.4. Prueba relé.

Los resultados de las pruebas de activación/desactivación y sincronización fueron satisfactorios. Los leds respondieron de forma precisa a los comandos enviados, encendiéndose y apagándose sin errores. Además, los cambios en el estado del relé se realizaron sin retrasos significativos.

### 4.3. Pruebas del sistema: Microcontrolador y Firmware

En esta sección se describen las pruebas realizadas sobre el microcontrolador ESP32-C3 y su firmware, evaluando tanto la funcionalidad del hardware como el rendimiento del software embebido.

- Prueba de conectividad y comunicación: se verificó que los nodos sensores y el nodo de relés, basados en ESP32-C3, pudieran establecer conexiones estableces mediante ESP-NOW con el gateway. En la figura 4.5, se puede observar cómo el nodo sensor y el nodo del módulo de relés lograron establecer una conexión exitosa con el gateway.

```
>>> import gateway
Punto de acceso iniciado
SSID: ESP32_AP
Contraseña: 12345678
('192.168.4.1', '255.255.255.0', '192.168.4.1', '0.0.0.0')
RTC configurado desde archivo JSON
Se activó ESP-NOW
Fecha y hora: 24/10/2024 20:9:0
Mensaje enviado: {'palabra_clave': 'reiniciar'}
Listening on ('0.0.0.0', 80)
Datos descriptados: {"palabra_clave": "buscar_canal"}
Nuevo nodo detectado: 08b61f811920 nodo módulo relés
Datos descriptados: {"palabra_clave": "verificar_canal"}
Datos descriptados: {"palabra_clave": "buscar_canal"}
Nuevo nodo detectado: 1091a83a7f68 nodo sensor
Datos descriptados: {"palabra_clave": "verificar_canal"}
```

FIGURA 4.5. Conexión de nodos al gateway.

- Prueba de carga y ejecución del firmware: se realizaron pruebas para validar que el firmware cargado en cada módulo ESP32-C3 se ejecutara correctamente. En los nodos sensores, se verificó la captura y envío de datos al gateway, mientras que en el nodo de relés se comprobó la recepción de comandos y la respuesta adecuada de los relés. Estos ensayos ya fueron detallados previamente en la sección **Prueba de componentes**
- Prueba de fallos en el firmware: se simularon reinicios inesperados o fallos en la carga del firmware en los nodos ESP32-C3. Se evaluó la capacidad del sistema para reiniciar correctamente y recuperar la funcionalidad sin perder la conexión o los datos.
- Prueba de gestión de cambios de canal: se realizó una prueba para verificar que los nodos pudieran sincronizarse correctamente con el gateway después de un reinicio y posible cambio de canal. Al iniciar, el gateway envía un comando de reinicio a los nodos para asegurarse de que todos los dispositivos se encuentren en el mismo canal de comunicación. En la figura 4.6, se puede observar el envío del comando de reinicio por parte del gateway.

```
>>> import gateway
Punto de acceso iniciado
SSID: ESP32_AP
Contraseña: 12345678
('192.168.4.1', '255.255.255.0', '192.168.4.1', '0.0.0.0')
RTC configurado desde archivo JSON
Se activó ESP-NOW
Fecha y hora: 24/10/2024 20:9:0
Mensaje enviado: {'palabra_clave': 'reiniciar'} comando de reinicio
Listening on ('0.0.0.0', 80)
```

FIGURA 4.6. Comando para reiniciar a los nodos.

En la figura 4.7, se puede observar que el nodo de relés recibió el comando de reinicio enviado por el gateway de manera exitosa. Tras recibir el comando, el nodo se reinició y, al completar el proceso de arranque, encontró y se conectó al canal correcto, restableciendo la comunicación con el gateway sin inconvenientes.

```

Datos desencriptados: {"palabra_clave": "reiniciar"}
reiniciando nodo
ets Jul 29 2019 12:21:46          Comando recibido en el nodo

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728          Reinicio del nodo
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with OTA with ESP32
Type "help()" for more information.
>>> import modulo_reles           Inicio correcto
ESP-NOW configurado correctamente.
Enviando mensaje de prueba en el canal: 1
Canal correcto encontrado: 1
Enviando mensaje de verificación...
Canal verificado correctamente.

```

FIGURA 4.7. Nodo reiniciado.

## 4.4. Pruebas sobre la aplicación web progresiva

En esta sección se describen las pruebas realizadas sobre la aplicación web progresiva (PWA) desarrollada para el monitoreo y gestión remota del sistema. Las pruebas incluyen la validación de las funcionalidades clave de la PWA, tales como la autenticación de usuarios, la visualización de datos de sensores, el control de los relés y la configuración de parámetros.

### 4.4.1. Prueba de inicio de sesión

Se llevaron a cabo pruebas para validar el proceso de inicio de sesión en la PWA, tanto con credenciales válidas como inválidas. Los ensayos incluyeron:

- Verificación con credenciales válidas: se ingresaron datos de acceso correctos (nombre de usuario y contraseña) y se comprobó que la aplicación permitiera acceder al sistema sin inconvenientes, redirigiendo al usuario a la interfaz principal.
- Verificación con credenciales inválidas: se ingresaron combinaciones de nombres de usuario y contraseñas incorrectas para asegurar que la PWA rechazaría el acceso de manera adecuada. Como se muestra en la figura 4.8, al intentar acceder con credenciales erróneas, la PWA devolvió un mensaje de error claro que indicaba "Usuario o contraseña incorrectos", sin permitir el acceso al sistema.

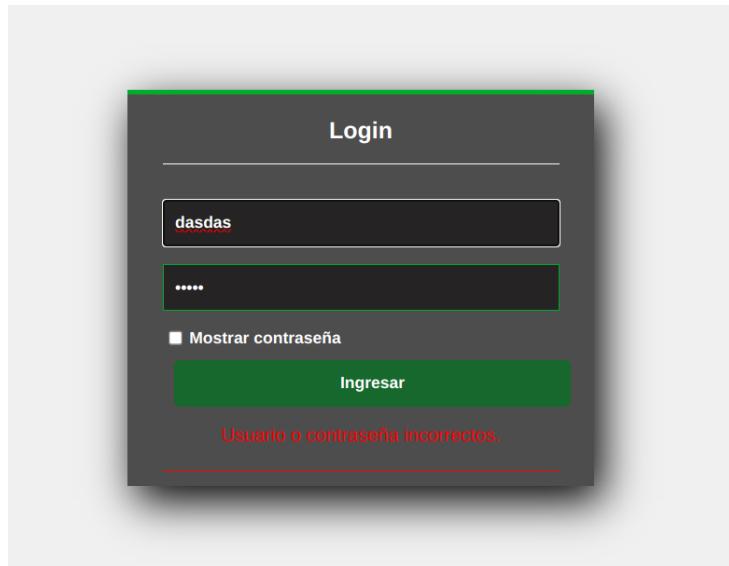


FIGURA 4.8. Error de login.

- Acceso sin iniciar sesión: Además, se probó la seguridad al intentar acceder a una URL o pantalla de la aplicación sin haber iniciado sesión previamente. Como se puede ver en la figura 4.9, la PWA mostró un mensaje de alerta que indicaba "No tiene autorización para ingresar a esta pantalla", impidiendo el acceso a secciones no autorizadas del sistema.

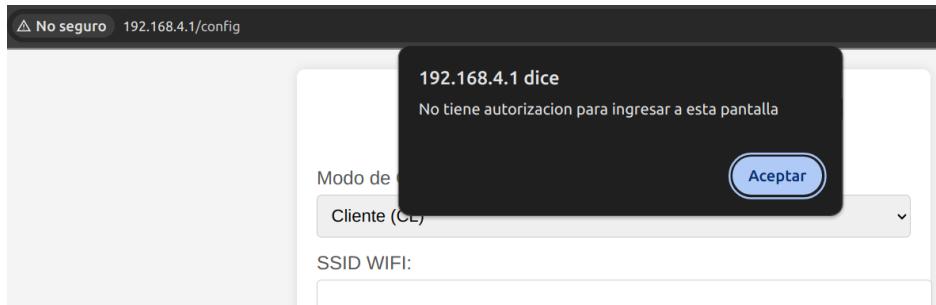


FIGURA 4.9. Mensaje sin iniciar sesión.

#### 4.4.2. Prueba de visualización de datos de sensores

Se realizaron pruebas para comprobar la precisión y actualización de los datos de los sensores que se muestran en la PWA. Las pruebas incluyeron:

- Comparación con datos en consola: se visualizaron los datos de los sensores en la PWA y se compararon con los valores que se registraban simultáneamente en la consola del sistema. Se verificó que ambos datos coincidieran y se actualizaran en tiempo real sin inconsistencias. Esto se puede visualizar en la figura 4.10.

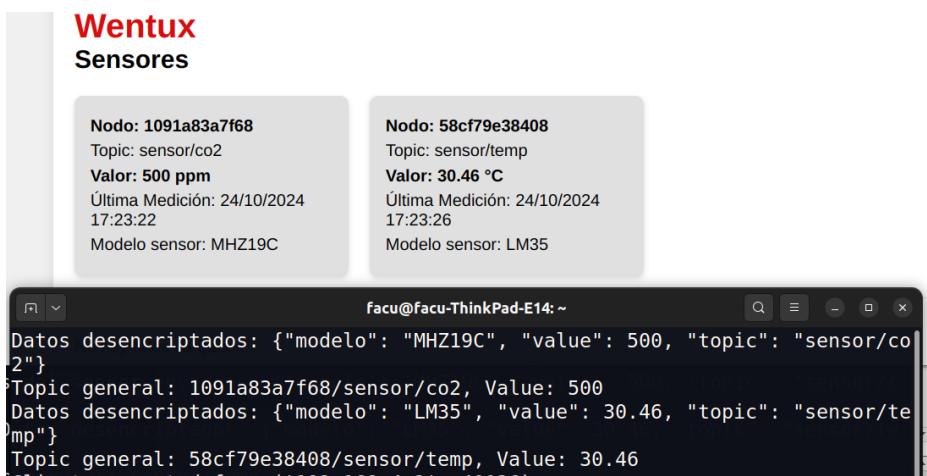


FIGURA 4.10. Comparación de datos.

- Verificación de actualización periódica: se comprobó que los datos de los sensores se actualizan automáticamente en la interfaz de la PWA sin necesidad de recargar la página, asegurando que los valores reflejan el estado más reciente de los sensores. En la figura 4.11 se puede observar cómo los datos se reciben correctamente cada dos minutos, la diferencia de dos segundos entre mediciones se debe a un ligero retraso en el proceso de adquisición de datos por parte de los sensores.

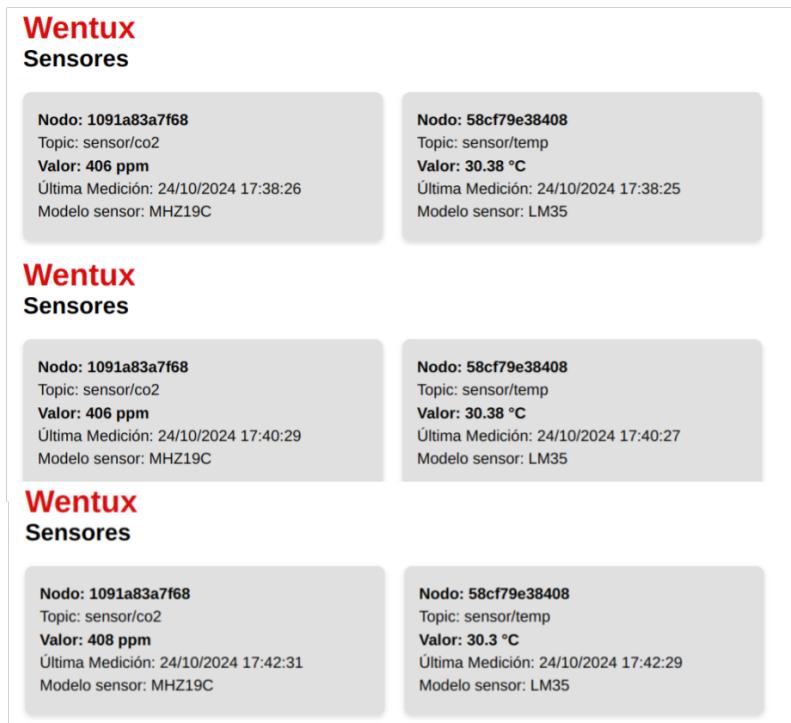


FIGURA 4.11. Actualización periódica de datos.

#### 4.4.3. Prueba de control de relés

El control de los relés desde la PWA fue evaluado para asegurar que los comandos enviados desde la interfaz se reflejaran correctamente en el hardware. Las

pruebas incluyeron:

- Sincronización con la consola: se compararon los estados de los relés mostrados en la PWA (on/off) con los registros de la consola del sistema, asegurando que ambos estuvieran sincronizados. Al activar o desactivar un relé desde la PWA, se verificó que los cambios también se reflejaran correctamente en la consola.
- Respuesta en tiempo real: se comprobó que los cambios en los estados de los relés se reflejaran casi instantáneamente en la PWA después de enviar el comando desde el gateway, asegurando que no hubiera retrasos significativos en la operación.

Como se muestra en la figura 4.12, los estados de los relés en la consola y en la PWA son idénticos, y los cambios se registran sin retrasos, lo que confirma el correcto funcionamiento del sistema.

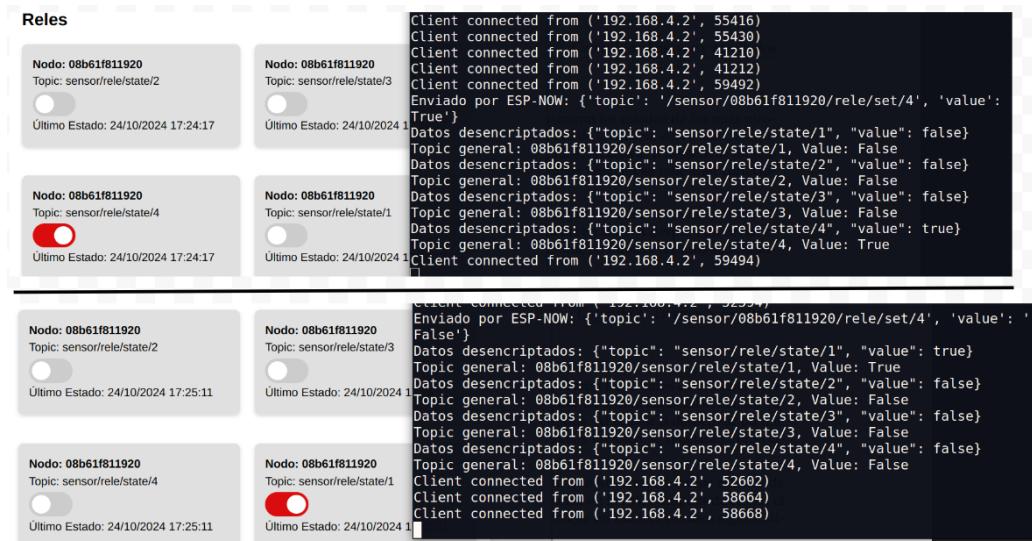


FIGURA 4.12. Prueba relés.

#### 4.4.4. Prueba de configuración

La sección de configuración de la PWA fue evaluada para asegurar que los ajustes del sistema pudieran ser modificados correctamente y que los errores en la entrada de datos fueran gestionados de manera adecuada. La prueba realizada fue la siguiente:

- Registro de errores en consola: se ingresaron datos incorrectos en los campos de configuración de la PWA para verificar que el sistema registrara estos errores de manera apropiada en la consola. Como se muestra en la figura 4.13, al introducir una contraseña de Wi-Fi incorrecta, el sistema registró un mensaje de error claro que indicaba 'contraseña incorrecta'. En respuesta a esto, el sistema cambió automáticamente al modo AP, permitiendo al usuario acceder nuevamente a la página de configuración para corregir el error y establecer los ajustes correctos.

**Página de Config**

Modo de Operación:  
Cliente (CL)

SSID WIFI:  
quepasapatejode

Contraseña WIFI:  
hhhhh

```
load:0x403cc710,len:0x70c
load:0x403ce710,len:0x2bc0
entry 0x403cc710
MicroPython v1.22.2 on 2024-08-26; ESP32C3 module with ESP32C3
Type "help()" for more information.
>>> import gateway
Conectando cliente WiFi a: quepasapatejode
.....Contraseña incorrecta
Cambiando a modo AP
Punto de acceso iniciado
SSID: ESP32_AP
Contraseña: 12345678
('192.168.4.1', '255.255.255.0', '192.168.4.1', '0.0.0.0')
```

FIGURA 4.13. Registro de error.

## 4.5. Pruebas sobre el servidor OpenRemote

Esta sección se describen las pruebas realizadas sobre el servidor IoT para verificar la recepción, almacenamiento y gestión de los datos provenientes de los sensores a través de un broker MQTT. Además, se valida el envío de comandos de control desde el servidor hacia el gateway para la operación de los relés.

### 4.5.1. Pruebas de recepción de datos desde el broker MQTT

- Verificación de suscripción al *broker* MQTT: se comprobó que el servidor OpenRemote estableciera una conexión exitosa con el *broker* MQTT, paso fundamental para la recepción de datos de los sensores. En la figura 4.14, se puede observar tanto la consola del contenedor Docker, donde se registra la conexión correcta, como la interfaz web de OpenRemote, que confirma también el estado de conexión al *broker*.

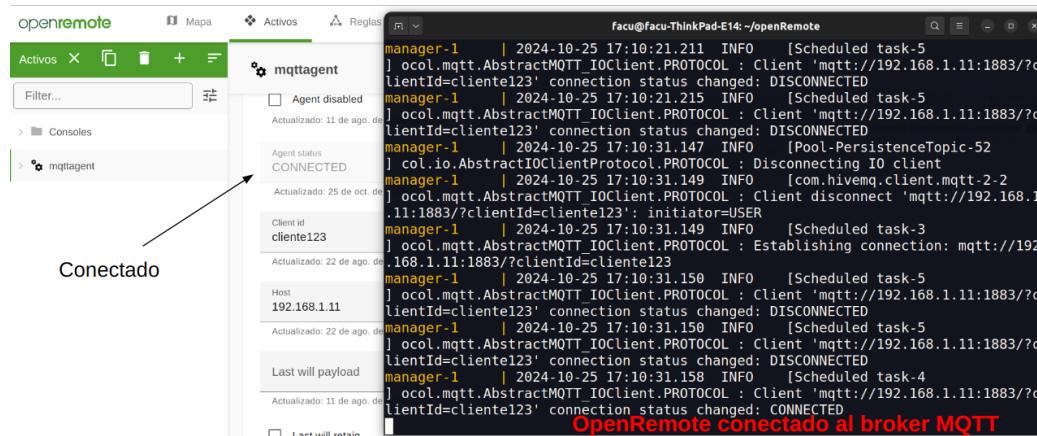


FIGURA 4.14. Verificación de conexión de openremote con el broker.

- Prueba de recepción de datos de sensores: se verificó que el servidor recibiera correctamente los datos de temperatura y CO<sub>2</sub> enviados desde el gateway a través del *broker* MQTT. En la figura 4.15, se muestra la interfaz de OpenRemote, donde los atributos de temperatura y CO<sub>2</sub> están actualizados, y junto a ella, la interfaz de MQTT Explorer [28], confirmando que los valores recibidos por el *broker* coinciden con los visualizados en OpenRemote. Esta coincidencia confirma que los datos se están transmitiendo y procesando correctamente desde el *gateway* hasta el servidor.

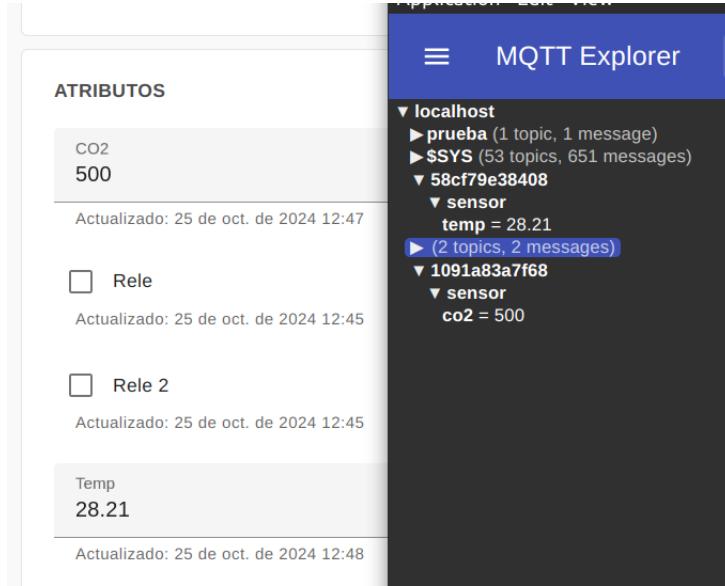


FIGURA 4.15. Recepción de datos en OpenRemote.

- Prueba de actualización en tiempo real: se validó que los datos recibidos se actualizan en tiempo real en el panel de control de OpenRemote, reflejando los cambios de los valores de los sensores sin necesidad de recargar la interfaz. Esto asegura que los usuarios puedan visualizar el valor más reciente de los sensores en el servidor.

#### 4.5.2. Pruebas de publicación de comandos a los relés

- Prueba de envío de comandos de control desde OpenRemote: se validó que el servidor pudiera enviar comandos al *broker MQTT* para controlar los estados de los relés a través del *gateway*. Al enviar un comando de activación/desactivación desde OpenRemote, se comprobó que el relé respondiera correctamente en tiempo real, sin retrasos significativos, y que el estado del relé se reflejara en la interfaz de OpenRemote. Esto se puede visualizar en la figura 4.16.

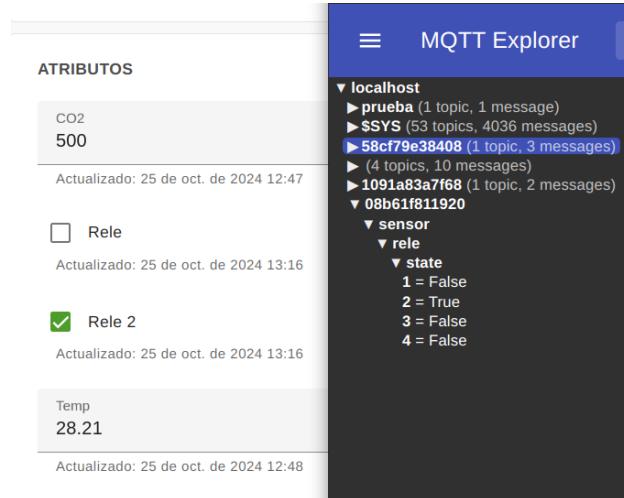


FIGURA 4.16. Envío de comandos desde OpenRemote.

- Sincronización del estado de los relés en OpenRemote: se verificó que los cambios en el estado de los relés se actualizan en la interfaz de OpenRemote, permitiendo a los usuarios verificar si los comandos fueron ejecutados correctamente en el hardware. Esto se observó mediante la comparación entre el estado mostrado en OpenRemote y el estado físico de los leds que simulan los relés. En la figura 4.17, se puede observar el cambio de estado del led que corresponde con lo que se visualiza en la interfaz de OpenRemote y MQTT Explorer.

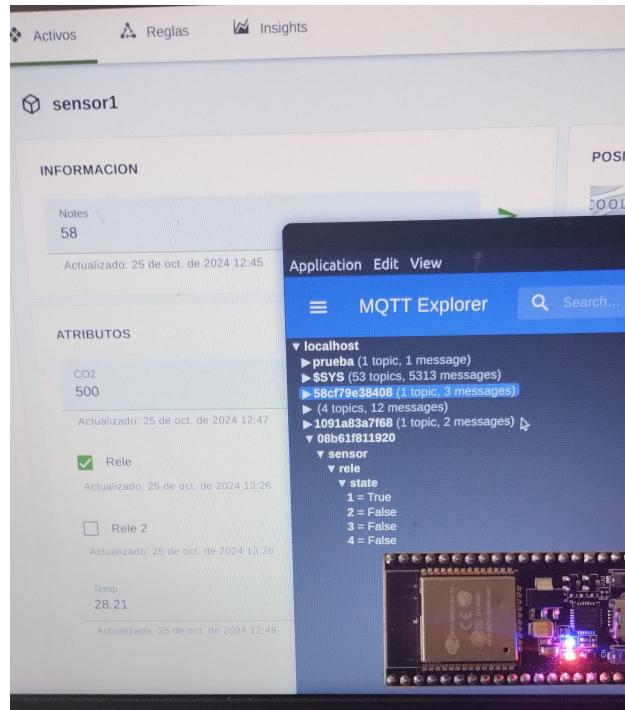


FIGURA 4.17. Verificación de cambio de estado del relé.

#### 4.5.3. Pruebas de almacenamiento y gestión de datos

- Verificación de almacenamiento de datos: se comprobó que OpenRemote almacenara correctamente los datos recibidos en su base de datos interna, permitiendo su consulta en el historial y su análisis posterior. En la figura 4.18, se observa la interfaz de OpenRemote, que ofrece la opción de revisar el historial de datos. Esta funcionalidad permite seleccionar tanto el atributo específico como el periodo de tiempo que se desea analizar, facilitando el acceso a información detallada para el seguimiento y evaluación de los datos recopilados.

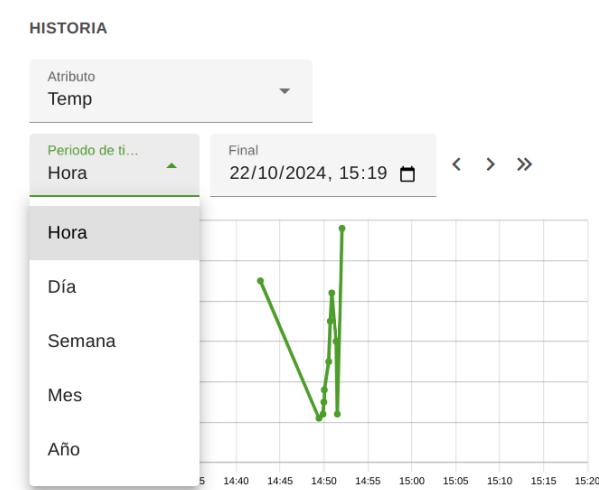


FIGURA 4.18. Opción de historia del atributo.

- Prueba de visualización de datos históricos: se verificó que el sistema permitiera al usuario visualizar los datos históricos de los sensores, facilitando el monitoreo y análisis de las condiciones ambientales en un periodo extendido. En la figura 4.19 se muestra un ejemplo de los datos históricos almacenados, evidenciando que el sistema puede gestionar grandes volúmenes de información sin comprometer el rendimiento.

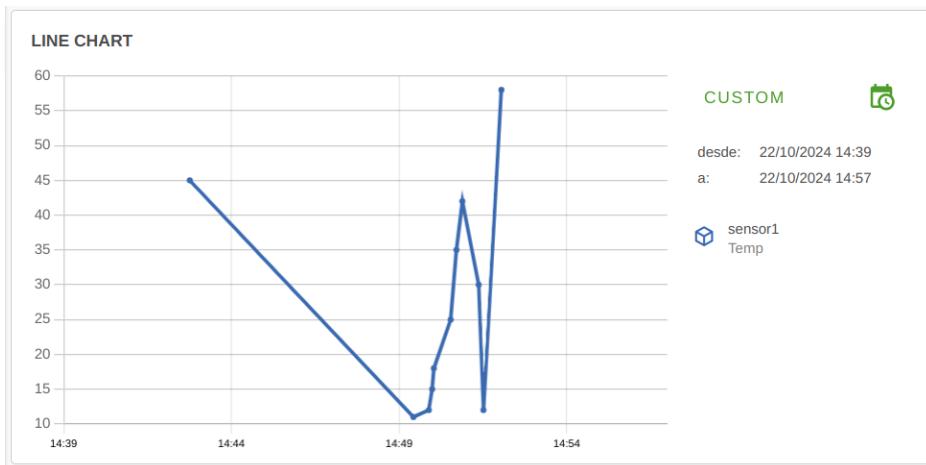


FIGURA 4.19. Grafico del sensor de temperatura.



## Capítulo 5

# Conclusiones

### 5.1. Conclusiones generales

La implementación de un sistema de monitoreo y gestión remota para invernaderos, basada en redes de sensores IoT, representa un avance significativo hacia una agricultura más eficiente y sostenible. Este trabajo ha logrado integrar exitosamente tecnologías de comunicación inalámbrica como ESP-NOW y MQTT, un servidor IoT basado en OpenRemote y una aplicación web para el control y visualización en tiempo real. La solución permite a los usuarios monitorear condiciones ambientales y gestionar dispositivos en invernaderos de forma remota y centralizada, mejorando la capacidad de respuesta ante cambios críticos y optimizando el uso de recursos.

Durante el desarrollo, se alcanzaron los objetivos principales, a pesar de ciertos ajustes realizados sobre la marcha para adaptar el sistema a los recursos embedidos. Aunque la planificación inicial fue flexible, el enfoque modular permitió avanzar en varios frentes simultáneamente. Esto fue especialmente útil cuando surgieron desafíos técnicos, permitiendo progresar en otros aspectos mientras se resolvían bloqueos específicos. Además, el trabajo enfrentó riesgos, siendo el principal la falta de tiempo hacia el final, especialmente en la elaboración de la memoria. La asignación de horas adicionales permitió alcanzar las metas y destacó la importancia de una planificación flexible y un plan de mitigación efectivo.

Para optimizar el sistema, se realizaron algunos cambios clave. Por ejemplo, se eligió el protocolo ESP-NOW en lugar de BLE, aprovechando sus ventajas en dispositivos ESP, lo que simplificó la implementación y garantizó una comunicación eficaz sin depender de una red Wi-Fi. Asimismo, el servidor IoT fue configurado para operar de forma local, en lugar de en la nube, cumpliendo con los requisitos actuales del cliente y permitiendo una posible migración en el futuro. La interfaz web se alojó en el nodo central, lo que limitó algunas funciones avanzadas, pero resultó funcional y adecuada para monitoreo local.

La integración de ESP-NOW y MQTT, junto con un enfoque modular, facilitó una comunicación fluida y segura entre los dispositivos y el servidor, validando la confiabilidad del sistema en diversas condiciones de operación.

Los ensayos y resultados obtenidos corroboraron la precisión del sistema en diversas condiciones, validando su eficacia para mejorar la supervisión y el control en un entorno agrícola. Además, la adaptabilidad del sistema a situaciones con y sin conectividad a internet amplía su aplicabilidad, especialmente en áreas rurales con limitaciones en la infraestructura de red.

En conclusión, este trabajo sienta las bases para el desarrollo de sistemas de monitoreo en agricultura, ofreciendo una solución flexible y que puede escalar con el tiempo. Esto ayuda a mejorar la sostenibilidad y el uso eficiente de los recursos en la agricultura moderna.

## **5.2. Próximos pasos**

Se podrían considerar las siguientes mejoras:

- Integración de algoritmos de inteligencia artificial (IA): incorporar IA para identificar patrones en los datos y proponer acciones preventivas o automatizadas en la gestión de cultivos, lo cual ayudaría a optimizar el rendimiento de los invernaderos.
- Notificaciones en tiempo real: ampliar la funcionalidad de la aplicación web para incluir notificaciones automáticas que alerten al usuario sobre cambios críticos en las condiciones ambientales o fallas en el sistema, mejorando la capacidad de respuesta.
- Personalización avanzada de alarmas: permitir a los usuarios configurar umbrales personalizados y ajustar el tipo de alertas, adaptándose así a distintos tipos de cultivo o necesidades específicas del entorno.
- Migración del servidor IoT a la nube: instalar el servidor en una plataforma en la nube para facilitar el acceso remoto y escalar la solución para supervisar múltiples invernaderos desde una única plataforma centralizada.
- Control remoto avanzado de dispositivos: desarrollar una interfaz que permita no solo monitorear sino también controlar aspectos avanzados de los dispositivos conectados, como la regulación de riego o ventilación de manera precisa.

Estas mejoras ayudarían a llevar el sistema actual a un nivel superior, haciéndolo más robusto, escalable y adaptable a las necesidades de la agricultura moderna.

# Bibliografía

- [1] MQTT. *MQTT: The Standard for IoT Messaging*. URL: <https://mqtt.org/>.
- [2] OpenRemote. *Docs OpenRemote*. URL: <https://docs.openremote.io/docs/introduction/>.
- [3] Wentux. *Wentux*. URL: <https://wentux.empretienda.com.ar/>.
- [4] Growcast. URL: <https://www.growcast.io/>.
- [5] Pulsegrow. URL: <https://pulsegrow.com/>.
- [6] Trolmaster. URL: <https://www.trolmaster.com/>.
- [7] Espressif. *ESP32-C3*. URL: <https://www.espressif.com/en/products/socs/esp32-c3>.
- [8] Espressif. *Docs ESP32-C3*. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32c3/get-started/index.html>.
- [9] Texas Instruments. *LM35 Precision Centigrade Temperature Sensors*. SNIS159H –AUGUST 1999–REVISED DECEMBER 2017. URL: <https://www.ti.com/lit/ds/symlink/lm35.pdf>.
- [10] Measurement Specialties. *HTU21D(F) Sensor*. October 2013. URL: <https://cdn-shop.adafruit.com/datasheets/1899-HTU21D.pdf>.
- [11] BOSCH. *BME280*. February 2024. URL: <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf>.
- [12] Ltd Zhengzhou Winsen Electronics Technology Co. *Intelligent Infrared CO2 Module (Model: MH-Z19)*. Valid from: 2015.03.03. URL: <https://www.winsen-sensor.com/d/files/PDF/Infrared%20Gas%20Sensor/NDIR%20CO2%20SENSOR/MH-Z19%20CO2%20Ver1.0.pdf>.
- [13] MicroPython. *MicroPython*. URL: <https://micropython.org/>.
- [14] CTA Electronics. *¿Qué es el MicroPython? Una introducción a Python-3 simplificado para microcontroladores*. URL: <https://www.ctaelectronics.com/es/micropython/>.
- [15] MicroPython. *Docs MicroPython*. URL: <https://docs.micropython.org/en/latest/>.
- [16] Espressif. *ESP-NOW*. URL: <https://www.espressif.com/en/solutions/low-power-solutions/esp-now>.
- [17] emariete. *Comunicación vía radio para ESP8266 y ESP32 con ESPNOW*. URL: <https://emariete.com/esp8266-esp32-espnow/>.
- [18] AWS. *¿Qué es MQTT?* URL: <https://aws.amazon.com/es/what-is/mqtt/>.
- [19] microsoft. *Introducción a las aplicaciones web progresivas (PWA)*. URL: <https://learn.microsoft.com/es-es/microsoft-edge/progressive-web-apps-chromium/>.
- [20] mozilla. *Progressive web apps*. URL: [https://developer.mozilla.org/en-US/docs/Web/Progressive\\_web\\_apps](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps).
- [21] MARIO VIDAL. *¿Qué son las Progressive Web Apps? ¿Por qué son tan importantes?* URL: <https://www.iebschool.com/blog/progressive-web-apps-analitica-usabilidad/>.

- [22] mozilla. *Service Worker API*. URL:  
[https://developer.mozilla.org/es/docs/Web/API/Service\\_Worker\\_API](https://developer.mozilla.org/es/docs/Web/API/Service_Worker_API).
- [23] OpenRemote. *OpenRemote*. URL: <https://www.openremote.io/>.
- [24] Mozilla. *HTML: Lenguaje de etiquetas de hipertexto*. URL:  
<https://developer.mozilla.org/es/docs/Web/HTML>.
- [25] Mozilla. *CSS*. URL: <https://developer.mozilla.org/es/docs/Web/CSS>.
- [26] Mozilla. *JavaScript*. URL:  
<https://developer.mozilla.org/es/docs/Web/JavaScript>.
- [27] Red Hat. *¿Qué es Docker y cómo funciona?* URL:  
<https://www.redhat.com/es/topics/containers/what-is-docker>.
- [28] Thomas Nordquist. *MQTT Explorer*. URL: <https://mqtt-explorer.com/>.