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

Índice de figuras

Índice de tablas

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 y el alcance de la solución propuesta.

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. 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, CO2 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.

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.

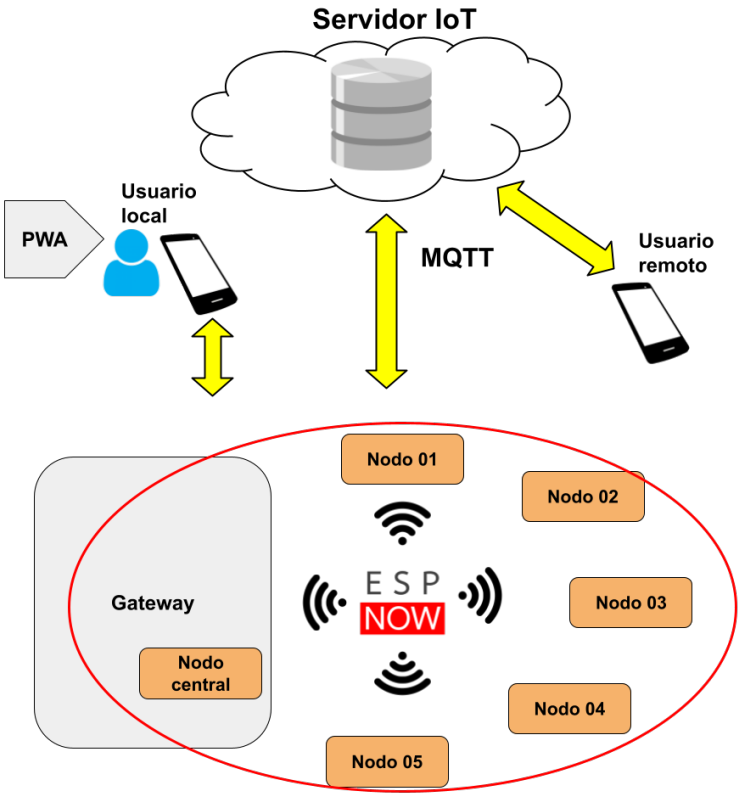

FIGURA 2.1. Diagrama en bloques del sistema.

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.

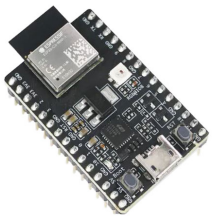

FIGURA 2.2. Módulo ESP32-C3.

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.

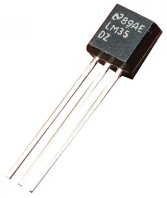

FIGURA 2.3. Sensor LM35.

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

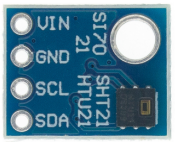

FIGURA 2.4. Sensor HTU21.

- 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.

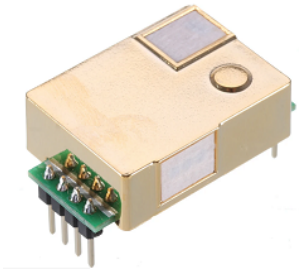

FIGURA 2.6. Sensor MH-Z19.

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].

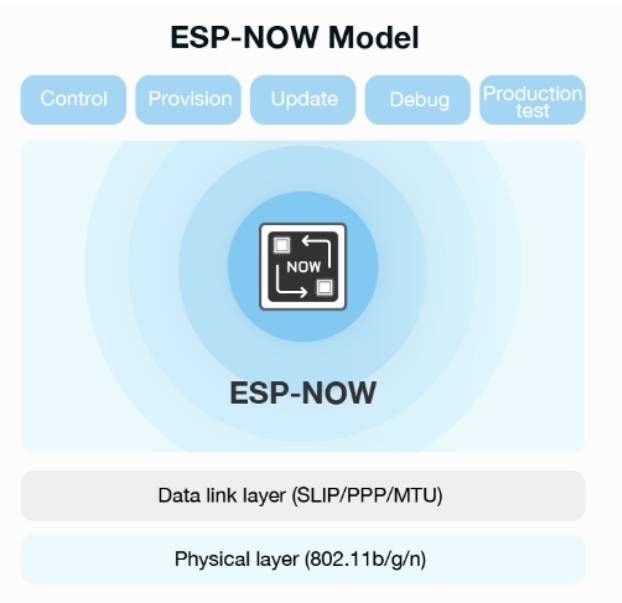

FIGURA 2.7. Modelo ESP-NOW.

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.

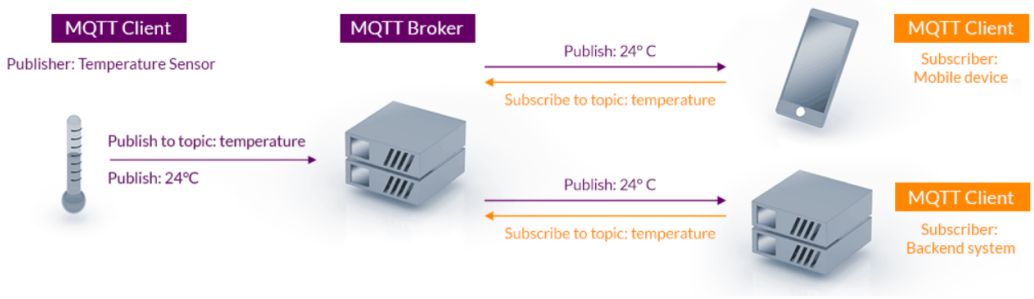

FIGURA 2.8. Arquitectura MQTT [1].

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.
- Instalable: se puede agregar directamente desde el navegador, lo que facilita el proceso para el usuario.
- Funciona 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.

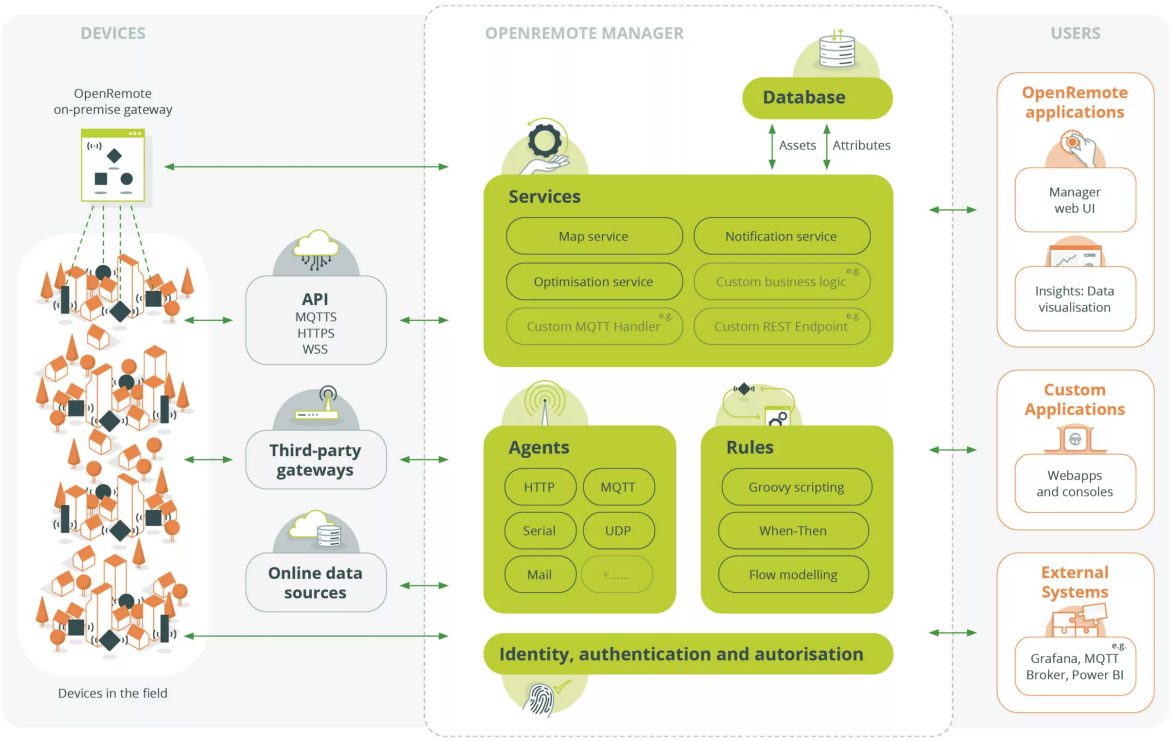

FIGURA 2.9. Arquitectura OpenRemote [2].

Capítulo 3

Diseño e implementación

En este capítulo se van a describir 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.

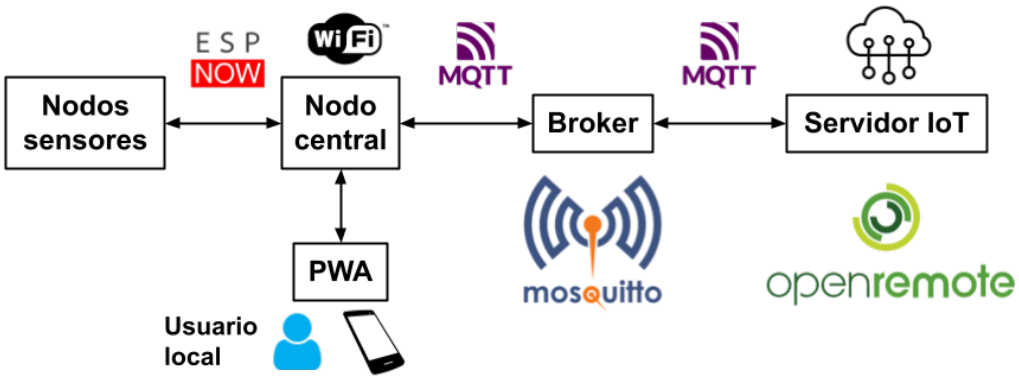

FIGURA 3.1. Esquema del sistema.

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 embebida 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.

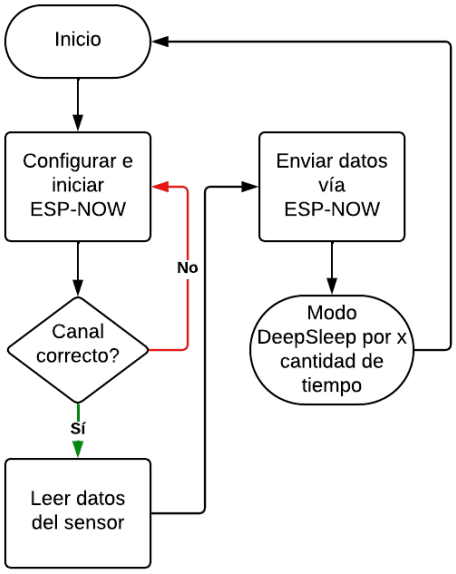

FIGURA 3.2. Diagrama de flujo de los nodos sensores.

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.

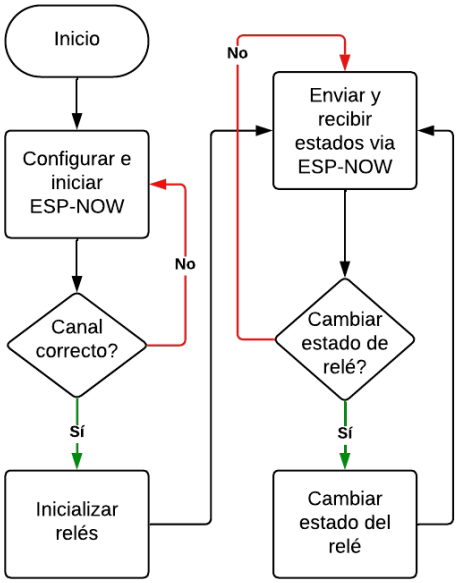

FIGURA 3.3. Diagrama de flujo del módulo relés.

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.

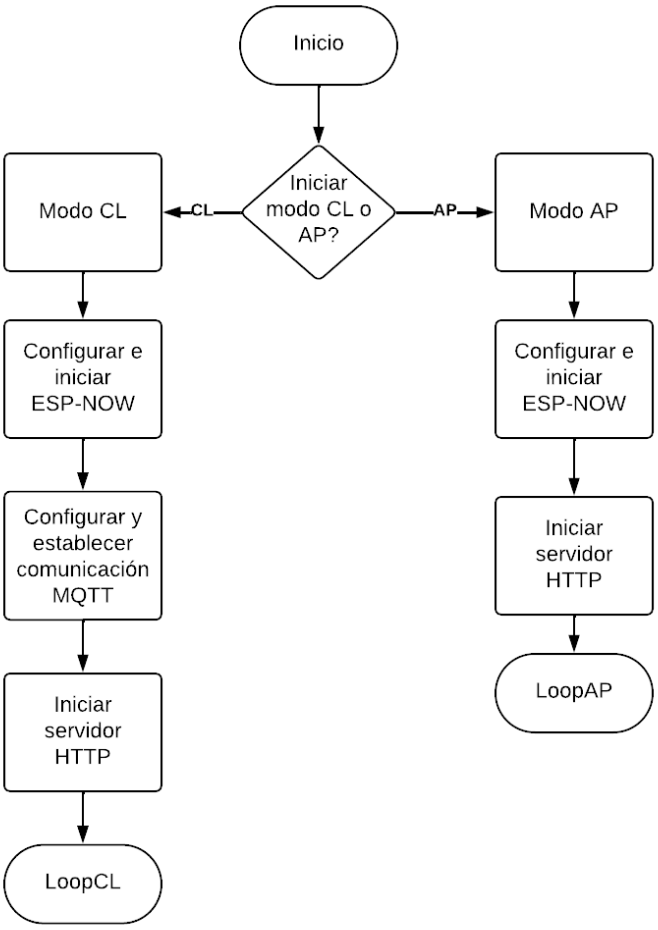

FIGURA 3.4. Diagrama de flujo del nodo central.

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.

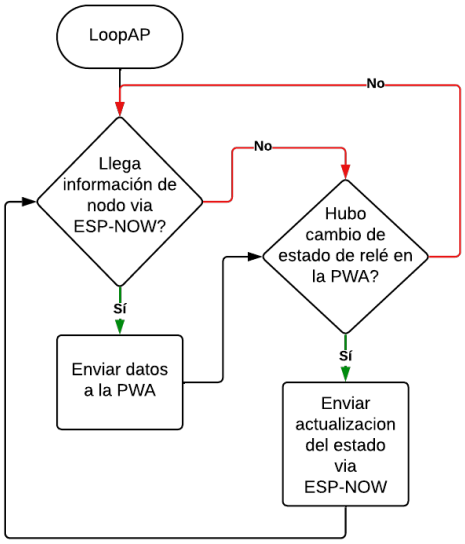

FIGURA 3.5. Diagrama de flujo del loop AP.

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.

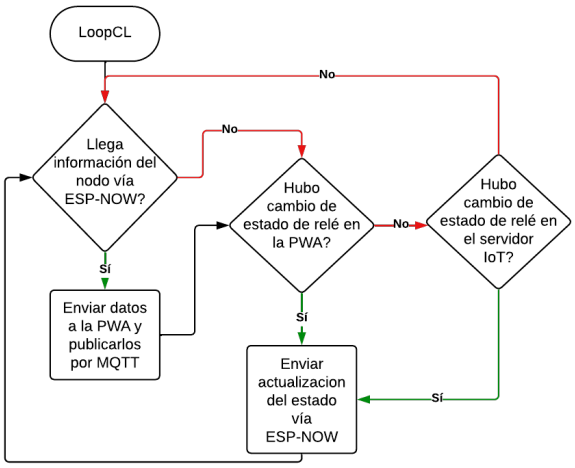

FIGURA 3.6. Diagrama de flujo del loop CL.

3.3. Desarrollo de la aplicacion web progresiva

3.4. Implementación del servidor IoT

3.5. Despliegue del sistema

Capítulo 4

Ensayos y resultados

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

4.1. Banco de pruebas

4.2. Prueba de componentes

4.3. Pruebas sobre el firmware

4.4. Pruebas sobre la aplicación web progresiva

4.5. Pruebas sobre el servidor OpenRemote

Capítulo 5

Conclusiones

5.1. Conclusiones generales

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puedo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.

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.

[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.
[23] OpenRemote. *OpenRemote*. URL: https://www.openremote.io/.