



Universitat
de les Illes Balears

TRABAJO DE FIN DE GRADO

SISTEMA DE GESTIÓN DOMÓTICA MEDIANTE ESP32: MONITOREO Y CONTROL AVANZADO

Luis Moreno García-Valenzuela

Grado de Ingeniería Electrónica Industrial y Automática

Escuela Politécnica Superior

Tutor: Gabriel Torrens Caldentey

Año Académico 2024-25

Resumen

La creciente demanda de soluciones IoT ha impulsado avances en la automatización del hogar, enfocándose en mejorar la comodidad, seguridad y eficiencia energética. Este trabajo de final de grado desarrolla un sistema de automatización basado en microcontroladores ESP32, que brinda una alternativa más flexible frente a estándares cableados como KNX, especialmente en instalaciones donde no se dispone de cableado adicional y se busca reducir costes.

Este proyecto nace con la motivación de democratizar el acceso a la tecnología de automatización, aprovechando microcontroladores ESP32 reconocidos por su bajo costo, versatilidad y capacidad de comunicación inalámbrica.

El objetivo es diseñar un sistema modular y escalable que optimice el consumo de energía y proporcione una interacción fluida con los dispositivos del hogar, a través de una interfaz web que facilite su control y configuración.

El sistema propuesto sigue una arquitectura multinodo compuesta por SensorNodes, MasterDevice y RoomNodes. Los SensorNodes, equipados con sensores SHT31, recopilan datos de temperatura y humedad y los transmiten al MasterDevice mediante el protocolo ESP-NOW, asegurando una comunicación de baja latencia. El MasterDevice, que ejecuta FreeRTOS, agrega los datos, almacena mediciones históricas y aloja un servidor web para la visualización en tiempo real y la configuración del sistema. La integración de WebSockets permite actualizaciones instantáneas en la interfaz web, mejorando la experiencia del usuario.

Por su parte, los RoomNodes controlan el aire acondicionado y la iluminación, empleando señales infrarrojas a 36.7 kHz y módulos de radiofrecuencia a 433 MHz, respectivamente. Además, se incluyen sensores LD2410C para la detección de presencia y un sensor TSL2591 para medir la luminosidad ambiental, implementando algoritmos que ajustan la intensidad de la iluminación y apagan automáticamente los dispositivos en ausencia de personas. FreeRTOS facilita la gestión de tareas concurrentes, garantizando eficiencia y robustez ante fallos.

Este proyecto se presenta como una alternativa eficiente y económica para entusiastas del DIY y pequeñas implementaciones domóticas, pues permite modernizar dispositivos tradicionales con una inversión mínima, aunque requiere un desarrollo y personalización significativa.

Las pruebas preliminares evidencian una transmisión de datos fiable, una recuperación sólida frente a errores y un control efectivo de dispositivos, además de funcionalidades de automatización que disminuyen el consumo energético. Futuros trabajos incluirán la mejora de los algoritmos de automatización, la optimización de la interfaz web, el aumento de la compatibilidad con otros dispositivos del hogar y la validación del rendimiento en entornos reales.

Abstract

The growing demand for IoT solutions has driven advancements in home automation, focusing on improving comfort, security, and energy efficiency. This undergraduate thesis develops an automation system based on ESP32 microcontrollers, offering a more flexible alternative to wired standards such as KNX, particularly in installations where additional wiring is unavailable and cost reduction is a priority.

This project was motivated by the goal of democratizing access to automation technology by leveraging ESP32 microcontrollers, known for their low cost, versatility, and wireless communication capabilities.

The aim is to design a modular and scalable system that optimizes energy consumption and provides seamless interaction with home devices through a web interface for easy control and configuration.

The proposed system employs a multi-node architecture consisting of SensorNodes, a MasterDevice, and RoomNodes. SensorNodes, equipped with SHT31 sensors, collect temperature and humidity data and transmit it to the MasterDevice using the ESP-NOW protocol, ensuring low-latency communication. The MasterDevice, running FreeRTOS, aggregates the data, stores historical measurements, and hosts a web server for real-time visualization and system configuration. The integration of WebSockets enables instant updates on the web interface, enhancing user experience.

RoomNodes manage air conditioning and lighting by using infrared signals at 36.7 kHz and 433 MHz radio frequency modules, respectively. Additionally, LD2410C sensors are included for presence detection, and a TSL2591 sensor measures ambient light, implementing algorithms that adjust lighting intensity and automatically switch off devices in the absence of occupants. FreeRTOS facilitates concurrent task management, ensuring efficiency and robustness against failures.

This project serves as a cost-effective and efficient alternative for DIY enthusiasts and small-scale home automation setups, allowing the modernization of traditional devices with minimal investment, albeit requiring significant development and customization.

Preliminary testing demonstrates reliable data transmission, solid error recovery, and effective device control, along with automation functionalities that reduce energy consumption. Future work will focus on improving automation algorithms, optimizing the web interface, increasing compatibility with other home devices, and validating performance in real-world environments.

ÍNDICE

Resumen.....	i
Abstract	ii
ÍNDICE	i
ÍNDICE DE TABLAS.....	iv
ÍNDICE DE FIGURAS.....	v
LISTA DE ACRÓNIMOS Y ABREVIATURAS	vi
1 Motivación	1
2 Objetivos	1
2.1 Objetivo General.....	1
2.2 Objetivos Específicos	1
3 Introducción	2
3.1 Evolución y Tendencias en la Automatización del Hogar	2
3.2 Limitaciones de las Soluciones Domóticas Inalámbricas	3
3.3 Microcontroladores ESP32: una Alternativa Eficiente y Accesible.....	3
3.4 Problemas y Retos en la Domótica Residencial	4
3.5 Solución del Sistema Propuesto.....	4
4 Desarrollo.....	5
4.1 Arquitectura del Sistema.....	5
4.1.1 Descripción de la Arquitectura Multinodo	6
4.1.2 Características Principales del Diseño.....	8
4.2 Comunicación	9
4.2.1 Protocolos de Comunicación entre Nodos Considerados.....	9
4.2.2 Protocolos de Comunicación Utilizados.....	10
4.2.3 Comunicación entre Nodos	12
4.2.4 Seguridad en la Comunicación entre Nodos.....	15
4.2.5 Comunicación con los Dispositivos del Hogar.....	16
4.3 Hardware.....	22

4.3.1	Selección de Microcontroladores	22
4.3.2	Componentes del Sistema.....	24
4.3.3	Conexión y Cableado	30
4.4	Software	32
4.4.1	Entorno de Desarrollo.....	32
4.4.2	Librerías Empleadas	33
4.4.3	Estructura del Código	35
4.4.4	Flujo del Programa	38
4.4.5	Funcionalidades Implementadas	42
4.4.6	Concurrencia y Gestión de Tareas con FreeRTOS	48
4.5	Interfaz Web.....	52
4.5.1	Justificación de la Arquitectura Elegida	52
4.5.2	Fundamentos de un Servidor Web en ESP32	54
4.5.3	Interacción Cliente-Servidor: HTTP y WebSockets	55
4.5.4	Almacenamiento y Rol de los Archivos Estáticos en LittleFS	57
4.6	Pruebas y Validación.....	57
4.6.1	Entorno de Pruebas y Metodología	57
4.6.2	Validación de Funcionalidades Principales	59
5	Resultados	61
5.1	Cumplimiento de Objetivos	61
5.2	Comparación de Costes	61
5.3	Limitaciones del Sistema	62
5.3.1	Seguridad en la Comunicación	62
5.3.2	Accesibilidad	62
5.3.3	Usabilidad.....	62
5.3.4	Portabilidad	63
5.3.5	Escalabilidad	63
5.3.6	Rendimiento y Fiabilidad	63
6	Conclusiones	63

6.1	Contexto.....	63
6.2	Principales Conclusiones e Implicaciones	64
6.3	Recomendaciones y Trabajos Futuros	65
REFERENCIAS		67
ANEXOS.....		69
Anexo I: Estructura de la Trama ESP-NOW.....		69
Anexo II: Cálculo de la Duración de la Batería		70
Anexo III: Justificación Duración 200ms de Ciclo de Actividad.....		71
Anexo IV: Pines Disponibles en el ESP32		71
Anexo V: Tabla de conexiones del RoomNode y SensorNode		73
Anexo VI: LittleFS.....		74
Características Principales.....		74
Ventajas frente a SPIFFS.....		75
Anexo VII: Cálculo de Costes por Nodo.....		75
SensorNode		75
RoomNode		76
MasterDevice		76
Anexo VIII: Montaje Físico Entorno de Pruebas		76

ÍNDICE DE TABLAS

Tabla 1. Tabla de Tipos de Mensaje	13
Tabla 2. Pinout del módulo LD2410C.....	25
Tabla 3. Pinout del Transmisor RF	26
Tabla 4. Pinout Transmisor RF	27
Tabla 5. Pinout Batería Lipo	29
Tabla 6. Especificaciones SHT31	30
Tabla 7. Pinout SHT31	30
Tabla 8. Tareas Concurrentes del MasterDevice	50
Tabla 9. Estructura General de una Trama ESP-NOW.....	69
Tabla 10. Pines Disponibles en el ESP32.....	73
Tabla 11. Conexiones RoomNode.....	74
Tabla 12. Conexiones SensorNode.....	74
Tabla 13. Coste de Componentes SensorNode.....	76
Tabla 14. Coste de Componentes RoomNode.....	76
Tabla 15. Coste de Componentes MasterDevice.....	76

ÍNDICE DE FIGURAS

Figura 1. Diagrama del Sistema	5
Figura 2. Plano de distribución del sistema objetivo.....	6
Figura 3. Unidad de Aire Acondicionado Panasonic CS-PZ15VKE.....	17
Figura 4. Ventilador de Techo con Luz INSPIRE Siroco	17
Figura 5. RTL-SDR Blog V4	19
Figura 6. Captura de pantalla SDRSharp	20
Figura 7. Captura URH: Señal Completa Correspondiente a Botón de Encendido y Apagado ..	20
Figura 8. Captura URH: Ampliación Señal Principal Botón Encendido/Apagado	21
Figura 9. ESP32-S3	23
Figura 10. ESP32 Dev Kit C V4	23
Figura 11. Wemos LOLIN D32.....	24
Figura 12. Sensor de Presencia LD2410C.....	24
Figura 13. FS1000A: Transmisor RF 433MHz.....	26
Figura 14. Módulo Emisor IR Dual	27
Figura 15. Sensor de Luz TSL2591.....	28
Figura 16. Batería LiPo	29
Figura 17. Sensor SHT31	29
Figura 18. Diagrama Esquemático del RoomNode.....	31
Figura 19. Diagrama Esquemática del SensorNode.....	31
Figura 20. Diagrama de Flujo SensorNode	39
Figura 21. Diagrama de Flujo RoomNode	40
Figura 22. Diagrama de Flujo MasterDevice	41
Figura 23. Captura de la Interfaz Web.....	46
Figura 24. Captura Interfaz Web: Historial de Temperatura y Humedad.	46
Figura 25. Diagrama Comunicación HTTP vs WebSocket	55
Figura 26. Plano Entorno de Pruebas	58
Figura 27. Captura de Pantalla de la Consola de Arduino IDE.	71
Figura 28. SensorNode en la Cocina alimentado por batería.	77
Figura 29. SensorNode en la Habitación 3 alimentado por cable.	77
Figura 30. MasterDevice en la Habitación 1.	78
Figura 31. SensorNode y RoomNode en la Habitación 1.	78
Figura 32. Esquema de Conexión RoomNode Pruebas	79
Figura 33. Buck Converter 24-12V a 5V	79

LISTA DE ACRÓNIMOS Y ABREVIATURAS

AA	Aire Acondicionado.
RF	Radiofrecuencia.
IR	Infrarrojos
KNX	EIB KONNEX.
NTP	<i>Network Time Protocol</i>
ASK	<i>Amplitude Shift Keying</i>
OOK	<i>On-Off Keying</i>
DIY	<i>Do-It-Yourself</i>
IoT	<i>Internet of Things</i>
MAC	<i>Media Access Control</i>
FMCW	<i>Frequency-Modulated Continuous Wave radar</i>
ISM	<i>Industrial, Scientific, Medical</i>
ISR	<i>Interrupt Service Routine</i>
RTC	<i>Real-Time-Clock</i>
PCB	<i>Printed Circuit Boards</i>
SMD	<i>Surface-Mount Devices</i>
SPIFFS	<i>SPI Flash File System</i>
SRAM	<i>Static Random Access Memory</i>
GPIO	<i>General Purpose Input/Output</i>
LiPo	<i>Lithium Polymer</i>
I2C	<i>Inter-Integrated Circuit</i>
UART	<i>Universal Asynchronous Receiver-Transmitter</i>

1 Motivación

La creciente demanda de soluciones domóticas refleja una transformación en la forma en que las personas interactúan con su hogar. En un mundo donde la sostenibilidad y la eficiencia energética son prioridades, contar con sistemas que optimicen recursos, incrementen la comodidad y promuevan la seguridad del hogar resulta crucial. Sin embargo, muchas de las tecnologías actuales, como los sistemas KNX, presentan barreras económicas, complejidad de instalación y una limitada flexibilidad [1] que dificultan su adopción generalizada.

Este proyecto nace con la motivación de democratizar el acceso a la tecnología de automatización, aprovechando microcontroladores ESP32 reconocidos por su bajo costo, versatilidad y capacidad de comunicación inalámbrica. También se busca explorar la posibilidad de integrar módulos de infrarrojos y radiofrecuencia para modernizar dispositivos tradicionales sin la necesidad de reemplazarlos, ofreciendo una alternativa innovadora y sostenible frente a sistemas convencionales.

El interés por desarrollar este sistema también responde a la necesidad de abordar desafíos técnicos como la interoperabilidad entre dispositivos, la robustez frente a fallos, la ejecución de algoritmos en paralelo y la facilidad de escalabilidad para futuras expansiones. Este enfoque busca no solo reducir la barrera económica de los sistemas actuales, sino también ofrecer una experiencia de usuario personalizada e intuitiva.

2 Objetivos

2.1 Objetivo General

Diseñar e implementar un sistema de automatización del hogar basado en microcontroladores ESP32, que sea modular, escalable, eficiente en términos energéticos y de bajo costo, brindando una interacción fluida mediante una interfaz web en tiempo real.

2.2 Objetivos Específicos

- **Diseño de Arquitectura Multinodo:** Desarrollar una estructura que integre nodos de sensores de temperatura y humedad (SensorNodes), nodos encargados del control de los dispositivos (luces y AA) de las habitaciones (RoomNodes) y un dispositivo maestro (MasterDevice).
- **Optimización de la Comunicación:** Implementar el protocolo ESP-NOW para la comunicación entre dispositivos, garantizando una comunicación inalámbrica eficiente,

de baja latencia y confiable. Debido a su ligereza, es especialmente beneficioso para el contexto del proyecto, dónde no se esperan intercambiar grandes tramas de datos.

- **Automatización Inteligente:** Incorporar algoritmos de automatización en los RoomNodes para gestionar la iluminación y climatización basados en sensores de presencia y mediciones de luminosidad. También será necesario integrar señales IR y RF para convertir dispositivos tradicionales como luces y aires acondicionados, convirtiéndolos en “inteligentes” con una inversión mínima.
- **Robustez y Recuperación de Errores:** Utilizar FreeRTOS para la gestión eficiente de tareas concurrentes y asegurar la correcta recuperación del sistema ante fallos.
- **Visualización y Configuración:** Diseñar una interfaz web accesible que permita a los usuarios visualizar datos históricos y en tiempo real, además de configurar parámetros del sistema.
- **Eficiencia Energética:** Optimizar la alimentación a baterías de los SensorNodes, encargados de medir temperatura y humedad, de manera que puedan colocarse de forma independiente en ubicaciones alejadas de tomas eléctricas. Para ello, se implementan modos de bajo consumo (*Deep Sleep*) y se aprovecha el protocolo ESP-NOW, que por su ligereza reduce al mínimo el tiempo que el SensorNode permanece despierto. Además, se incorporan algoritmos que reduzcan el uso innecesario de dispositivos, como el apagado automático cuando no se detecta presencia, con el fin de evitar consumos innecesarios en los aparatos controlados.
- **Validación y Escalabilidad:** Realizar pruebas en entornos reales para validar la fiabilidad del sistema y diseñar un sistema modular que permita agregar habitaciones y dispositivos adicionales sin modificaciones significativas.
- **Sostenibilidad:** Reutilizar dispositivos existentes como las luces y unidades de aire acondicionado, otorgándoles una segunda vida “inteligente”, minimizando la huella ambiental al evitar reemplazos completos de hardware más moderno.

3 Introducción

3.1 Evolución y Tendencias en la Automatización del Hogar

La automatización del hogar ha ganado un protagonismo significativo en la última década, impulsada por los avances en tecnologías de la información, el Internet de las cosas (IoT) y la necesidad creciente de optimizar la eficiencia energética. Estas tecnologías permiten a los usuarios monitorear y automatizar dispositivos cotidianos, como luces, calefacción y electrodomésticos, mejorando así la calidad de vida y optimizando el consumo de recursos.

Actualmente, las soluciones domóticas se dividen en dos grandes categorías:

- **Sistemas basados en bus de cableado**, como KNX[2], que destacan por su robustez y fiabilidad [3]. Aunque existen variantes inalámbricas (KNX RF [4]), lo habitual es que requieran instalaciones cableadas especializadas, lo que puede resultar costoso y complejo [1]. Estas soluciones están orientadas principalmente a grandes proyectos y entornos profesionales, donde los altos costes de implementación y mantenimiento son asumibles.
- **Sistemas inalámbricos** basados en tecnologías IoT, han crecido exponencialmente gracias a su flexibilidad, sencilla instalación y menor coste de despliegue. Estas soluciones suelen basarse exclusivamente en la comunicación sin cables, empleando protocolos como Wi-Fi, Bluetooth, Zigbee u otros [5], eliminando la necesidad de cableado.

3.2 Limitaciones de las Soluciones Domóticas Inalámbricas

Los sistemas cableados tradicionales no se presentan como la opción más adecuada para sistemas domóticos pequeños debido a sus costos elevados y la complejidad de su instalación. No obstante, las soluciones inalámbricas no están exentas de limitaciones. Muchos dispositivos dependen de una conexión a Internet estable y plataformas externas, lo que puede comprometer la fiabilidad del sistema. Además, algunos sistemas comerciales, como Philips Hue [6], requieren hubs adicionales, lo que incrementa el coste y dificulta la interoperabilidad con dispositivos de otros fabricantes. Por otro lado, el coste de integración de sistemas comerciales sigue siendo una barrera para usuarios con presupuestos ajustados.

3.3 Microcontroladores ESP32: una Alternativa Eficiente y Accesible

En este contexto, la llegada de microcontroladores económicos y versátiles, como el ESP32, ha transformado la automatización del hogar. Gracias a su conectividad inalámbrica dual (Wi-Fi y Bluetooth), bajo consumo energético y su compatibilidad con módulos, el ESP32 permite desarrollar **soluciones personalizadas, modulares y escalables**. Además, el uso de protocolos como ESP-NOW [7] facilita la comunicación eficiente entre dispositivos sin depender de una infraestructura compleja ni de conexión continua a Internet.

Este enfoque representa una alternativa eficiente y económica para entusiastas del **DIY(Do-It-Yourself)** y pequeñas implementaciones domóticas, ya que permite convertir dispositivos tradicionales en inteligentes con una inversión mínima. Sin embargo, su aplicación requiere un desarrollo y personalización significativa, lo que lo hace más adecuado para proyectos personales y de pequeña escala que para soluciones comerciales o industriales a gran escala.

3.4 Problemas y Retos en la Domótica Residencial

A pesar de los avances en automatización del hogar, los sistemas actuales enfrentan varios desafíos, especialmente en entornos residenciales donde se prioriza la sencillez, el bajo coste y la eficiencia energética. Soluciones tradicionales como KNX, si bien son robustas y ampliamente utilizadas en viviendas de alto standing, pueden resultar poco rentables para proyectos de menor escala debido a sus elevados costes de instalación y la complejidad de su implementación [8]. Por otro lado, muchos sistemas IoT inalámbricos comerciales dependen de plataformas en la nube, lo que introduce problemas de fiabilidad, latencia y seguridad.

El problema central abordado en este proyecto es la falta de una solución económica, modular y eficiente que permita a los usuarios convertir dispositivos tradicionales (luces, AA) en dispositivos inteligentes, sin depender de infraestructuras complejas ni inversiones significativas.

Se requiere, por tanto, un sistema que:

- Permita monitorizar y controlar dispositivos domésticos de forma inalámbrica.
- Sea económico y accesible para usuarios con presupuestos reducidos.
- Optimice el consumo energético mediante algoritmos de automatización.
- Ofrezca flexibilidad y escalabilidad, facilitando la incorporación de nuevos dispositivos y habitaciones.
- Sea independiente de plataformas externas, reduciendo la dependencia de la conexión a Internet.

3.5 Solución del Sistema Propuesto

La solución propuesta se basa en una arquitectura multinodo modular que integra microcontroladores ESP32, optimizando la eficiencia energética y la interacción con dispositivos tradicionales del hogar.

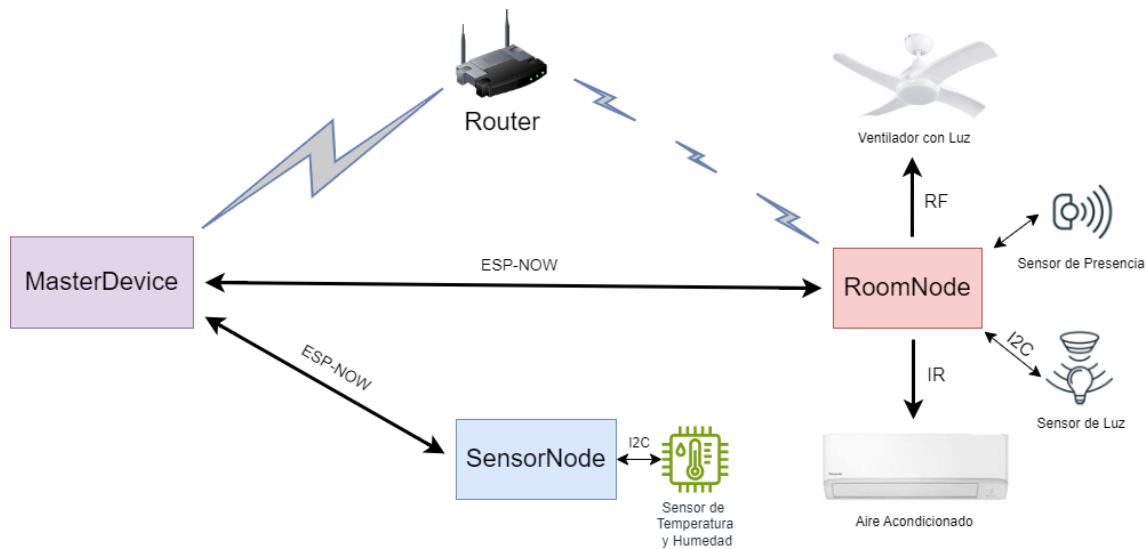


Figura 1. Diagrama del Sistema

Los componentes principales del sistema son:

- **SensorNodes:** dispositivos de bajo consumo equipados con sensores de temperatura y humedad, que envían datos al nodo central mediante el protocolo ESP-NOW, optimizando la eficiencia energética.
- **MasterDevice:** nodo central encargado de agrupar la información, almacenar datos históricos y ofrecer una interfaz web en tiempo real a los usuarios.
- **RoomNodes:** dispositivos que interactúan mediante módulos RF e IR con la iluminación y el AA, convirtiendo dispositivos tradicionales en inteligentes. Estos nodos integran sensores de presencia (LD2410C) y luminosidad (TSL2591) para implementar algoritmos de automatización que optimizan el consumo energético.

Con este enfoque, el proyecto ofrece una solución accesible y escalable para la automatización del hogar, permitiendo a los usuarios modernizar sus dispositivos existentes con una inversión mínima, y mejorando así la eficiencia energética y la comodidad de la vivienda.

4 Desarrollo

4.1 Arquitectura del Sistema

La arquitectura del sistema desarrollado sigue un enfoque multinodo, diseñado para garantizar flexibilidad, modularidad y una distinción clara de tareas entre componentes del sistema. Esta organización responde a los objetivos planteados al proporcionar una solución, accesible, escalable y eficiente para convertir dispositivos tradicionales en “inteligentes”.

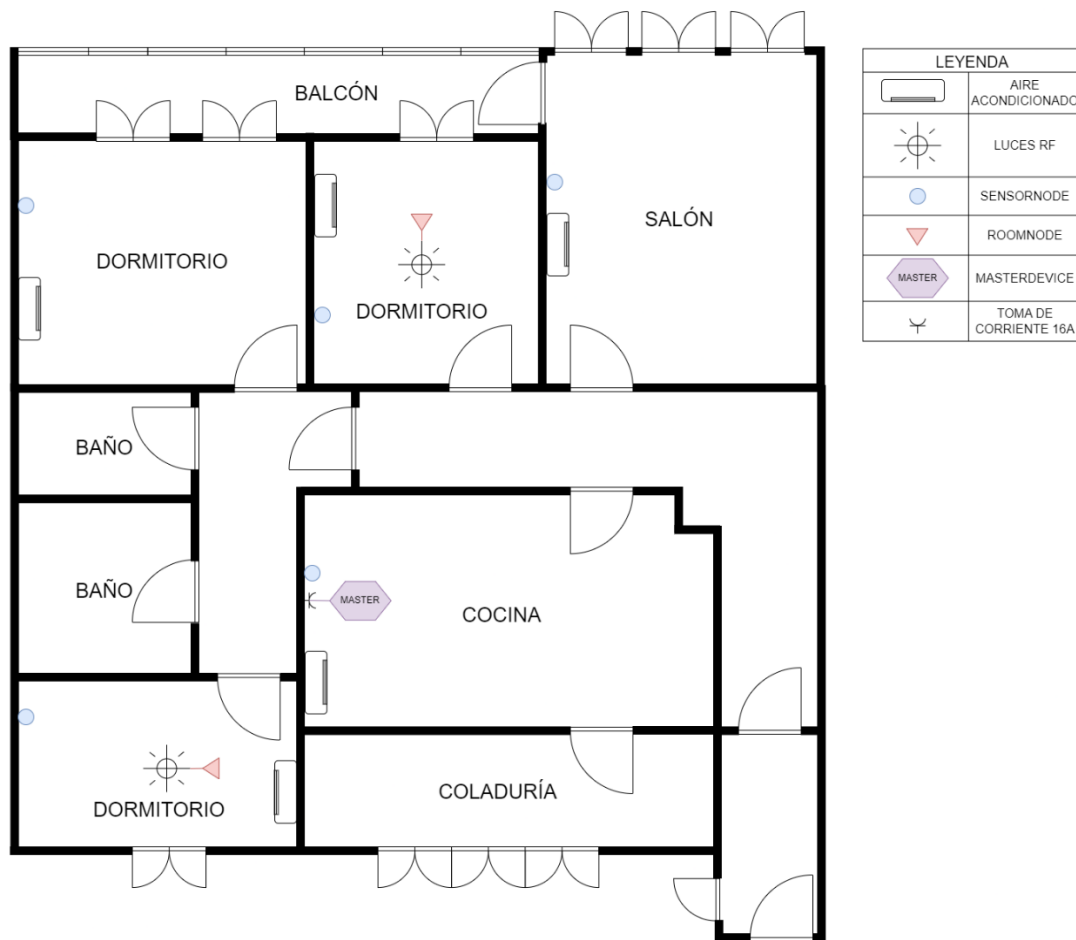


Figura 2. Plano de distribución del sistema objetivo

4.1.1 Descripción de la Arquitectura Multinodo

La arquitectura propuesta se basa en tres tipos de nodos principales: SensorNodes, MasterDevice y RoomNodes. Cada uno de ellos cumple un rol específico y se comunican de manera coordinada para recopilar, procesar información o ejercer acciones de control sobre dispositivos domésticos. A continuación, se describe brevemente su función, hardware, conexión y ubicación.

4.1.1.1 SensorNodes

Los SensorNodes tienen como función **recopilar datos ambientales**, como la temperatura y humedad, mediante sensores de bajo consumo (SHT31). Para lograrlo, se han empleado placas LOLIN D32 (Figura 11. Wemos LOLIN D32) optimizados **para reducir el consumo energético** cuando el microcontrolador se encuentra en *Deep Sleep* (modo de sueño profundo), permitiendo que las baterías duren el mayor tiempo posible. La periodicidad con la que se despierta el SensorNode es configurable a través de la interfaz web del MasterDevice.

En términos de comunicación, estos nodos usan el protocolo ESP-NOW para transmitir información directamente al MasterDevice, y el protocolo I2C para interactuar con el sensor SHT31 (Figura 17. Sensor SHT31). Se distribuyen de forma estratégica en distintas habitaciones,

evitando áreas con fuentes de calor o frío extremos (e.g., cercanas a sistemas de aire acondicionado o ventanas) que podrían distorsionar las mediciones ambientales.

4.1.1.2 RoomNodes

Los RoomNodes están enfocados en el **control de dispositivos** específicos (luces y aire acondicionado) en cada habitación, posibilitando automatizaciones basadas en la detección de presencia y la medición de luminosidad.

El hardware principal consiste en un ESP32 Dev Kit C V4 (Figura 10. ESP32 Dev Kit C V423) y módulos que incluyen un emisor IR (de 940 nm) para el aire acondicionado, un transmisor RF (ASK/OOK a 433.92 MHz) para el control de las luces, un LD2410C (radar de ondas milimétricas a 24 GHz) como detector de presencia y un sensor TSL2591 (Figura 15. Sensor de Luz TSL2591) para medir la luz ambiental.

Estos nodos reciben órdenes e informan de cambios al MasterDevice vía ESP-NOW y se conectan de forma periódica al Wi-Fi para sincronizar la hora con NTP.

Se ubican preferiblemente en el techo, lo que facilita la línea de visión con los dispositivos IR y RF, y se alimentan en la misma toma eléctrica que las luces para un montaje sencillo.

4.1.1.3 MasterDevice

El MasterDevice centraliza los datos recogidos por los SensorNodes y RoomNode, a los que les comunica las peticiones realizadas por el usuario. También opera como servidor web, ofreciendo una interfaz para que los usuarios consulten y configuren el sistema. La interfaz web presenta:

- **Datos ambientales** en tiempo real de los SensorNodes (temperatura y humedad).
- Un **historial** de datos ambientales de cada habitación con gráficos.
- **Configuración del período de sueño** de los nodos.
- **Encendido y apagado** de luces.
- Definición de franjas horarias de **luz cálida o fría**

Este dispositivo recibe datos de los SensorNodes y RoomNodes a través de ESP-NOW y, al mismo tiempo, se mantiene conectado a la red Wi-Fi permanentemente para proporcionar el servidor web y sincronizar la hora con servidores NTP, garantizando la precisión del tiempo en la información mostrada en la interfaz web.

Se instala en un punto central de la vivienda, alimentado por la red eléctrica para asegurar un funcionamiento continuo y un alcance óptimo hacia los demás nodos.

4.1.2 Características Principales del Diseño

El diseño del sistema ha sido seleccionado para cumplir con los objetivos del proyecto, ofreciendo las siguientes ventajas:

Eficiencia Energética

Los SensorNodes aprovechan el modo *Deep Sleep* de los ESP32 para disminuir drásticamente su consumo, prolongando la vida de las baterías. A la vez, los RoomNodes gestionan la iluminación y el aire acondicionado de forma automatizada, basándose en sensores de presencia y luminosidad.

Modularidad y Escalabilidad

La naturaleza multinodo de la arquitectura favorece la incorporación o eliminación de unidades (SensorNodes o RoomNodes) conforme varíen las necesidades del hogar. De este modo, se garantiza la expansión del sistema sin perturbar el rendimiento de los nodos existentes.

Independencia de Infraestructura

La comunicación entre nodos mediante ESP-NOW no depende de plataformas en la nube, a diferencia soluciones como AWS IoT, un servicio ofrecido por Amazon Web Services que permite gestionar dispositivos IoT a través de la nube para monitorización, análisis y control remoto. Al prescindir de estas plataformas, el sistema mejora la privacidad y reduce puntos de falla asociados con servicios externos o conexiones a Internet.

Aunque el MasterDevice está conectado al router local para ofrecer acceso a la interfaz web y sincronizar la hora mediante NTP, el sistema puede seguir funcionando parcialmente incluso en ausencia de conexión a Internet, ya que los RoomNodes y SensorNodes no dependen de esta para operar.

Sin embargo, es importante mencionar que si se pierde la conexión a Internet durante un periodo largo de tiempo, el reloj interno podría desviarse al no actualizarse con NTP, lo que afectaría la precisión de los horarios programados.

Coste Asequible

Se aprovechan módulos económicos y se reutilizan dispositivos existentes (luces y aires acondicionados), reduciendo el gasto global. Esto hace la solución viable para proyectos de tipo DIY y entornos residenciales donde se busca una automatización con inversiones mínimas.

Robustez y Fiabilidad

El diseño multinodo especializa cada componente en una función concreta, minimizando conflictos y colisiones. Al separar las tareas, se facilita la localización de errores y se evita que un problema en un nodo afecte a todo el sistema.

4.2 Comunicación

4.2.1 Protocolos de Comunicación entre Nodos Considerados

En la etapa inicial de diseño del proyecto se evaluaron diferentes tecnologías y protocolos de comunicación para satisfacer diferentes requerimientos, como bajo consumo, independencia de infraestructura y posibilidad de conexión a Internet. Las principales ventajas y desventajas que se valoraron de cada uno de los protocolos destacados fueron:

Wi-Fi tradicional

El protocolo Wi-Fi, basado en la pila de protocolos 802.11, es ampliamente utilizado en entornos domésticos. El soporte nativo de Wi-Fi en el ESP32 facilita la implementación de servidores web y la sincronización con Internet.

- **Ventajas:**
 - **Protocolo ampliamente extendido**, con abundante documentación y soporte[9]
 - Compatible con ESP32 y sencillo de implementar.
 - **Conectividad a Internet**: imprescindible para funcionalidades como alojar un servidor web y sincronizar la hora mediante NTP.
- **Desventajas:**
 - **Consumo energético elevado**[10], poco adecuado para dispositivos alimentados por batería.
 - **Dependencia** de un **router local**.

ESP-NOW

ESP-NOW es un protocolo propietario de Espressif, fabricante de los procesadores ESP32 y similares. Ese protocolo está diseñado para comunicaciones ligeras *peer-to-peer* de baja latencia entre dispositivos ESP32 sin necesidad de un router.

- **Ventajas:**
 - **Baja latencia y menor consumo** que Wi-Fi. [8].
 - **No requiere** infraestructura de **router**.
 - Ideal para intercambios de datos entre microcontroladores ESP32 **sin conexión** continua a **Internet**.
 - **Mayor alcance** que el Wi-Fi[11].
- **Desventajas:**
 - **Número limitado de peers**[12].
 - **Sin confirmación nativa** de recepción.

Zigbee

Protocolo de red en malla muy utilizado en IoT, enfocado al bajo consumo y a la comunicación de corto alcance. Zigbee permite a los dispositivos reenviar paquetes de datos, creando redes autoorganizadas.

- **Ventajas:**

- **Bajo consumo**, especialmente en redes con alta concentración de dispositivos.
- **Estándar IoT** muy común, presente en muchos dispositivos comerciales.
- El microcontrolador ESP32-C6 es compatible con el protocolo Zigbee.

- **Desventajas:**

- Requiere un **coordinador Zigbee** y, en el caso del ESP32-C6, un firmware específico y desarrollo en ESP-IDF.
- Mayor **complejidad** de configuración
- **Falta de adopción nativa** en otros módulos **ESP32**.

RF 433 MHz

Comunicación por radiofrecuencia en banda ISM (433.92 MHz), muy popular en dispositivos de control remoto simples (e.g. luces, persianas).

- **Ventajas:**

- **Simple** y bajo coste.
- **Bajo consumo**
- Posibilidad de aprovechar el módulo RF ya incorporado para las luces en los RoomNodes.

- **Desventajas:**

- **Unidireccional**, poca seguridad y sin confirmación nativa de recepción.
- **No escalable** para grandes volúmenes de datos o conexiones bidireccionales complejas.

4.2.2 Protocolos de Comunicación Utilizados

Tras evaluar diversas tecnologías (Wi-Fi, ESP-NOW, Zigbee, RF a 433 MHz y servicios en la nube), se decidió combinar ESP-NOW y Wi-Fi como protocolos principales. Esta elección responde a la necesidad de lograr un equilibrio entre eficiencia energética, flexibilidad de implementación y simplicidad de configuración, sin sacrificar la autonomía y de acuerdo con el enfoque del proyecto.

4.2.2.1 Elección de ESP-NOW y Wi-Fi

ESP-NOW aporta **baja latencia y bajo consumo**, que resulta ideal para la comunicación de datos internos entre MasterDevice, SensorNodes y RoomNodes. Gracias a su comunicación directa con

el receptor y la ausencia de una pila TCP/IP completa, los SensorNodes se mantienen despiertos el menor tiempo posible, aprovechando al máximo el ahorro energético del modo *Deep Sleep*.

Por otro lado, **Wi-Fi** resulta necesario para que el MasterDevice se conecte al router local y aloje la **interfaz web**, además de sincronizar la hora mediante servidores NTP que requieren acceso a Internet. Cabe mencionar que esta conexión con el router podría establecerse mediante cable Ethernet en lugar de Wi-Fi, sin embargo, en la mayoría de entornos domésticos, la infraestructura más habitual o conveniente sigue siendo la inalámbrica.

La combinación de ambos protocolos permite una **arquitectura híbrida**: ESP-NOW cubre la comunicación interna eficiente y de bajo consumo, mientras que Wi-Fi satisface las funcionalidades que exigen conectividad o servicios más complejos. Esta configuración, además, reduce la dependencia de la nube, salvaguarda la privacidad de la información y mejora la robustez del sistema al permitir que el sistema siga funcionando parcialmente de forma local incluso sin acceso a Internet.

4.2.2.2 Por qué no se centraliza todo en Wi-Fi

Aunque la solución más sencilla parecería ser usar Wi-Fi en todos los nodos, esto incrementaría notablemente el **consumo energético de los SensorNodes**. Esto se debe a que, para transmitir datos, los nodos deberían establecer una conexión con la red Wi-Fi, lo cual requiere un tiempo considerable debido a la complejidad del protocolo. Además, aunque Wi-Fi es capaz de transmitir datos a una velocidad mucho mayor que ESP-NOW, su pila TCP/IP añade una sobrecarga innecesaria cuando solo se necesitan transmitir pequeños volúmenes de datos, como las mediciones de los sensores.

De esta forma, optar por un protocolo más ligero y directo como ESP-NOW para la comunicación interna permite optimizar el tiempo que los SensorNodes permanecen activos, logrando una comunicación eficiente y reduciendo el consumo energético al mínimo. Este enfoque prolonga significativamente la duración de las baterías, lo que resulta crítico en dispositivos que dependen de una alimentación autónoma.

4.2.2.3 Zigbee y por qué se descarta

Si bien Zigbee es muy eficiente y de bajo consumo en redes malladas con múltiples dispositivos, su implementación requeriría un coordinador y, en el caso de ESP32-C6, un firmware específico con funcionalidades Zigbee limitadas. Esto obligaría a usar dicho modelo de microcontrolador para todos los nodos, eliminando la flexibilidad de elegir otros ESP32 mejor adaptados a requisitos como *Deep Sleep* extremo, integración de múltiples módulos o mayores capacidades de memoria. Además, instalar un *hub* adicional para gestionar Zigbee engrosaría la infraestructura y no aportaría ventajas reales en el hogar actual. Por el contrario, si en un futuro se previese una residencia con gran cantidad de dispositivos IoT Zigbee, podría ser recomendable usar

microcontroladores con Zigbee nativo (e.g., nRF52840, EFR32 o ESP32-C6) para lograr una integración plena y optimización de la comunicación y consumo del sistema.

4.2.2.4 RF a 433 MHz y sus limitaciones

En el proyecto, los RoomNodes requieren de módulos RF a 433 MHz para controlar luces ya existentes. Sin embargo, extender este protocolo a los demás nodos resultaría ineficiente, al carecer estos de tales módulos y exigir de su integración. Además, la mayoría de los módulos RF a 433 MHz son **unidireccionales** y no contemplan seguridad o confirmación de recepción, **disminuyendo la robustez** del sistema, especialmente cuando se deben comunicar múltiples dispositivos a la vez. Si bien cabría la posibilidad de diseñar un protocolo propio ligero con una modulación robusta y de bajo consumo para los SensorNodes, surgirían problemas de seguridad y fiabilidad, junto con una complejidad añadida en el desarrollo.

4.2.2.5 Servicios en la Nube y su falta de conveniencia

También se estudió la posibilidad de subordinar la comunicación en la nube, y aunque servicios como AWS IoT, Google Cloud IoT o Azure IoT ofrecen gran escalabilidad y potentes herramientas de análisis, requerirían una conexión Wi-Fi constante en todos los nodos, elevando sustancialmente el consumo y aumentando la dependencia de Internet. El escenario local y de pequeño alcance de este proyecto hace que las ventajas de la nube **no compensen los costes y la complejidad** de configuración. Además, la privacidad del sistema se vería comprometida al subir los datos a servidores externos.

4.2.3 Comunicación entre Nodos

4.2.3.1 Estructura del Mensaje

En el sistema diseñado, la comunicación interna entre MasterDevice, SensorNodes y RoomNodes se realiza mediante el protocolo ESP-NOW. Dentro de ESP-NOW, cada trama se transporta como un bloque de bytes, detalles adicionales sobre la estructura de una trama ESP-NOW pueden consultarse en el Anexo I: Estructura de la Trama ESP-NOW. En este proyecto, se ha definido una estructura de mensaje propia dentro del contenido de la trama que consta de:

- **Primer byte:** Especifica el MessageType, que indica la función o categoría de la trama (e.g., TEMP_HUMID, NEW_SLEEP_PERIOD, ACK, etc.).
- **Bytes restantes (*payload*):** Contienen la información específica de la trama, que puede ser, e.g., los valores de temperatura y humedad, o los parámetros para una nueva programación de horario.

La definición de los MessageType y las estructuras de cada payload se encuentran en el archivo common.h, archivo común entre todos los nodos del proyecto.

Este sistema de comunicación se ha incorporado para dar flexibilidad y eficiencia al proyecto, puesto que cada tipo de mensaje lleva únicamente la información esencial (e.g., valores de temperatura, horario de luces, etc.), minimizando el tamaño de la trama y, por ende, el consumo y procesamiento del mensaje.

4.2.3.2 Tipos de Mensaje del Sistema

En la siguiente tabla se describen los tipos de mensajes que se intercambian entre nodos, junto con el emisor, destinatario, funcional principal y la estructura del *payload*.

Mensaje	Identificador	Emisor	Destinatario	Función	Estructura de Datos
JOIN_SENSOR	0x00	SensorNode	MasterDevice	MasterDevice registra un nuevo SensorNode	uint8_t room_id : Identificación de la habitación. uint32_t sleep_period_ms : Duración del <i>Deep Sleep</i> .
JOIN_ROOM	0x01	RoomNode	MasterDevice	MasterDevice registra un nuevo RoomNode	uint8_t room_id : Identificador de la habitación. Time warm : Tiempo del día en el que se activa el modo de luz caliente. Time cold : Tiempo del día en el que se activa el modo de luz fría.
ACK	0x02	Cualquiera	Cualquiera	Confirma recepción del mensaje	MessageType acked_msg : Indica que mensaje se confirmó.
TEMP_HUMID	0x03	SensorNode	MasterDevice	Reporta temperatura y humedad	uint32_t room_id : Identificador de la habitación. float temp : Temperatura medida. float humid : Humedad medida.
NEW_SLEEP_PERIOD	0x04	MasterDevice	SensorNode	Informa a SensorNode del nuevo periodo de sueño configurado por el usuario	uint32_t new_period_ms : Nueva duración del <i>Deep Sleep</i> configurada por el usuario.
NEW_SCHEDULE	0x05	MasterDevice	RoomNode	Informa de la nueva programación de iluminación configurada por el usuario.	Time cold : Nuevo tiempo del día en el que se activa el modo de luz fría. Time warm : Nuevo tiempo del día en el que se activa el modo de luz caliente.
HEARTBEAT	0x06	RoomNode	MasterDevice	Indica que RoomNode sigue activo.	Uin8_t room_id : Identificador de la habitación.

Tabla 1. Tabla de Tipos de Mensaje

4.2.3.3 Registro y Sincronización de Nodos

Para lograr una arquitectura flexible y escalable, el proyecto implementa un **registro dinámico** de los nodos. En lugar de fijar por adelantado todas las direcciones MAC de SensorNodes y RoomNodes, se han diseñado dos mensajes especiales: JOIN_SENSOR y JOIN_ROOM.

El mensaje JOIN_SENSOR es enviado por los SensorNodes en su primer ciclo de operación y contiene la información necesaria ('room_id', 'sleep_period_ms') por el MasterDevice para registrar el dispositivo. Mientras que el mensaje JOIN_ROOM es emitido por cada RoomNode al iniciar e incluye campos como 'room_id', 'Time warm', 'Time cold' (horarios iniciales para modos de iluminación) y 'lights_on' (estado actual de las luces).

Esta información es utilizada por el **MasterDevice** para **registrar** el nodo en su **base de datos** y enlazarlo como *peer*, para posteriormente poder comunicarles los cambios en la configuración que haya realizado el usuario.

Este mecanismo de registro dinámico donde la única dirección MAC fija del sistema es la del MasterDevice, prioriza la maleabilidad y flexibilidad del sistema, permitiendo añadir y eliminar nodos con facilidad.

4.2.3.4 Mecanismo de Escaneo de Canales

Debido a que los microcontroladores ESP32 pueden **operar únicamente en un canal Wi-Fi** al mismo tiempo, de los 13 canales disponibles, para comunicaciones ESP-NOW y Wi-Fi, ha sido necesario implementar un **algoritmo** para establecer **la comunicación inicial** entre nodos. Esto se debe a que el MasterDevice debe estar conectado permanentemente al Wi-Fi debido a sus funciones como servidor web, lo que solo le permite recibir mensajes ESP-NOW mediante el canal Wi-Fi que le asigne el router, de este modo se implementó un proceso de escaneo de canales Wi-Fi en los SensorNodes y RoomNodes cuando tratan de unirse al sistema ('JOIN_SENSOR' o 'JOIN_ROOM').

Este mecanismo consiste en enviar el respectivo mensaje de unión ('JOIN_SENSOR' o 'JOIN_ROOM') en cada canal de Wi-Fi, hasta recibir confirmación por parte del MasterDevice.

Este método permite no solo detectar el canal en el que opera el MasterDevice al inicio de la ejecución del nodo, sino también tratar de restablecer la comunicación en caso de que el MasterDevice se haya reiniciado inesperadamente o desconectado momentáneamente.

A diferencia del RoomNode, cuando SensorNode es incapaz de establecer conexión con el MasterDevice después de haber escaneado todos los canales Wi-Fi disponibles, este entra en modo de descanso profundo hasta que se reinicie manualmente, por motivos de ahorro de batería.

4.2.3.5 Mecanismo Heartbeat entre RoomNode y MasterDevice

Una de las funcionalidades clave para asegurar la disponibilidad de los RoomNodes es el mecanismo de *heartbeat*. Este mecanismo sirve para confirmar periódicamente que un RoomNode sigue activo y en línea.

Cada RoomNode envía un mensaje HEARTBEAT en intervalos regulares. El mensaje incluye el campo 'room_id' para que el MasterDevice identifique de qué nodo proviene. El MasterDevice

mantiene un registro de la última vez que se recibió el *heartbeat* de cada RoomNode y envía un mensaje de confirmación ACK cuando lo recibe. De esta forma, si un RoomNode no recibe la confirmación de mensaje antes de un tiempo determinado, interpreta que el MasterDevice se ha desconectado o reiniciado e intenta unirse a la red (JOIN_ROOM) de nuevo.

4.2.3.6 Borrado de Nodos Inactivos

Dentro del contexto del mecanismo de *heartbeat* entre el RoomNode y MasterDeive, si transcurre un tiempo mayor que un límite configurado (e.g., 1 minuto mayor al periodo de envío) sin recibir un nuevo *heartbeat*, el MasterDevice asume que el RoomNode está desconectado o inoperativo y lo borra de su registro.

De forma similar ocurre con el SensorNode, cuando el MasterDevice detecta un retraso significativo en el envío de la trama TEMP_HUMID, borra al respectivo SensorNode del su registro.

En un futuro el sistema puede alertar al usuario de estas desconexiones a través de la interfaz web o notificaciones por Telegram.

4.2.3.7 Mecanismo de Confirmación (ACK)

Para los mensajes con mayor importancia, como aquellos relacionados con configuraciones críticas o comandos específicos, se ha implementado un mecanismo de confirmación (ACK).

Este sistema asegura que el emisor del mensaje recibe una confirmación explícita de que el receptor ha procesado la información. Mientras que, en caso de no recibir un ACK, el mensaje se retransmite automáticamente hasta un número máximo de intentos.

4.2.4 Seguridad en la Comunicación entre Nodos

La seguridad en la comunicación es un elemento clave en cualquier sistema de IoT, ya que garantiza la integridad, confidencialidad y disponibilidad de los datos. En el presente proyecto, se han priorizado la eficiencia energética y la flexibilidad de la arquitectura, razones por las cuales no se ha habilitado la encriptación en las tramas ESP-NOW.

4.2.4.1 Razones para la Omisión de Encriptación en ESP-NOW

- **Registro Dinámico de Nodos:** El cifrado requiere que los *peers* conozcan previamente las MAC y claves de cada dispositivo, dificultando la incorporación “sobre la marcha” de nuevos SensorNodes y RoomNodes.
- **Complejidad y Consumo:** La gestión de claves y autenticación mutua para la encriptación incrementaría la complejidad y podría elevar el consumo energético, oponiéndose a la filosofía de bajo consumo en nodos alimentados por batería.

- **Baja Criticidad de las Funciones:** Dado que actualmente solo se controlan aspectos como luces y mediciones ambientales, no se consideran datos o acciones críticas para la seguridad del hogar.

4.2.4.2 Riesgos Asociados

En el diseño actual del sistema, el **riesgo** de que un dispositivo no autorizado se incorpore a la red es **limitado** y de difícil ejecución. Esto se debe a que un atacante necesitaría obtener varios elementos clave, como la dirección MAC del MasterDevice, la estructura interna de los mensajes y proximidad a la red local. Estas barreras técnicas hacen que la probabilidad de un ataque exitoso sea baja.

Además, las funciones que gestiona el sistema, como el control de luces y la recopilación de datos ambientales, no son críticas para la seguridad del hogar. Por lo tanto, incluso en el improbable caso de que un nodo malicioso lograra infiltrarse, su impacto sería mínimo y no comprometería datos sensibles ni funcionalidades de alto riesgo.

4.2.4.3 Perspectivas Futuras

Si en futuras implementaciones el sistema se expande para incluir **dispositivos de alta relevancia**, como cerraduras electrónicas o alarmas, será esencial adoptar **medidas de seguridad** más avanzadas. Una de estas medidas sería habilitar el cifrado en ESP-NOW, acompañado de un sistema de autenticación para garantizar que solo los *peers* confiables puedan comunicarse en la red. La incorporación del cifrado permitiría proteger la integridad y confidencialidad de las tramas, asegurando que los datos intercambiados no sean interceptados e imposibilitando obtener las direcciones MAC de los nodos por esta vía.

Aunque en este proyecto inicial se ha **priorizado la escalabilidad y facilidad de despliegue**, evitando el cifrado para permitir un registro dinámico y flexible de nodos, no se descarta la implementación de medidas de seguridad más estrictas en el futuro.

4.2.5 Comunicación con los Dispositivos del Hogar

En esta sección se describen los mecanismos de comunicación utilizados para interactuar con los dispositivos del hogar, concretamente el control de luces remotas a 433 MHz y la emisión de señales infrarrojas para el aire acondicionado. Se dispone en el hogar de:

- 5x Unidades de aire acondicionado modelo CS-PZ15VKE de marca Panasonic repartidos en múltiples habitaciones.



Figura 3. Unidad de Aire Acondicionado Panasonic CS-PZ15VKE

- 2x Ventiladores de techo con luz en dos dormitorios.



Figura 4. Ventilador de Techo con Luz INSPIRE Siroco

4.2.5.1 Comunicación Infrarroja (IR)

Los equipos de aire acondicionado, televisores y otros dispositivos de entretenimiento emplean mayoritariamente señales infrarrojas para su control. Estas señales se generan modulando una portadora de frecuencia típicamente entre 36 kHz y 40 kHz en ráfagas de pulsos que representan las distintas órdenes.

Para controlar las unidades de aire acondicionado, es necesario conocer la secuencia de pulsos específica de cada marca/modelo. Estas secuencias de datos suelen contener el estado completo del sistema, lo que simplifica el control al no requerir seguimiento del estado actual. Existen **librerías especializadas** que abstraen el envío de señales IR para una amplia variedad de fabricantes, facilitando la integración y replicación de las órdenes. En este proyecto se emplea la librería IRremoteESP8266 [13], que permite interactuar, entre muchos otros fabricantes, con los dispositivos Panasonic ya instalados en el hogar.

A pesar de las ventajas, la comunicación infrarroja presenta limitaciones como la dependencia de la orientación del emisor y las restricciones en distancia, que a pesar de usar dos diodos emisores en este proyecto, es importante tener en cuenta y asegurar una línea de visión clara y distancia

cercana con el aire acondicionado. Por ello se ha posicionado el RoomNode en el techo y a no más de 2 metros del dispositivo.

4.2.5.2 Comunicación RF a 433 MHz

Para el control de las luces remotas instaladas en los dormitorios del hogar (Figura 4. Ventilador de Techo con Luz INSPIRE Siroco) se deben **replicar** las **señales** de radiofrecuencia que emite el **mando remoto original** de este modelo específico mediante un módulo transmisor RF conectado al RoomNode. Esta clase de mandos suelen utilizar modulación por amplitud (ASK) en la banda ISM de 433 MHz, empleando específicamente On-Off Keying (OOK), una variante sencilla en la que la amplitud de la portadora solo adopta dos valores: presencia o ausencia de señal.

La banda de 433.92 MHz, asignada en la Unión Europea para uso sin licencia[14], es popular para dispositivos domésticos como luces, enchufes inteligentes y sistemas de alarma. Su simplicidad y flexibilidad han llevado a su adopción masiva en el ámbito del hogar.

Cabe destacar que, aunque esta banda facilita la interoperabilidad entre dispositivos, es **susceptible a interferencias** por otros equipos que operen en la misma frecuencia. Para garantizar una operación fiable y escalable, se ha tenido en cuenta la posibilidad de interferencias y se han aplicado medidas de verificación de cambios mediante sensores de luminosidad y mecanismos de reintentos.

Con fines de replicar la señal de radiofrecuencia del mando original de las luces, se ha capturado la señal a clonar mediante una radio definida por software (Figura 5. RTL-SDR Blog V4), empleando herramientas como SDRSharp y Universal Radio Hacker para grabar y decodificar los datos necesarios para su correcta reproducción.

4.2.5.3 Clonación de las Señales RF de las Luces

Para que el RoomNode controle las luces remotas a 433 MHz, ha sido necesario clonar las señales RF emitidas por el mando original. El objetivo de esta clonación es reproducir la modulación y los patrones de bits de cada botón, de modo que las luces “interpreten” la señal como si proviniera de su propio mando.

Herramientas utilizadas

- RTL-SDR Blog V4: Un receptor de radio definida por software (SDR) conectado a una antena que mejora la calidad y la intensidad de la señal recibida.



Figura 5. RTL-SDR Blog V4

- **SDRSharp:** Software que permite visualizar el espectro de la señal captada por el SDR, verificar su frecuencia de operación con exactitud y estimar parámetros como ancho de banda, niveles de potencia y posibles interferencias.
- **Universal Radio Hacker (URH):** Herramienta para capturar, grabar y analizar señales de RF. Facilita la decodificación binaria y la identificación de patrones de pulsos (longitudes, pausas, repeticiones, etc.).

Captura y análisis inicial de la señal

En primer lugar, se conectó el RTL-SDR Blog V4 a un ordenador y a una antena externa para mejorar la recepción. A través de SDRSharp, se sintonizó la frecuencia de 433.92 MHz y se confirmó que al pulsar cualquier botón del mando original de las luces aparecía en pantalla una señal, que se repetía mientras se mantuviese el botón presionado.

También se verificó que no utilizaba modulación FSK, ya que de haber sido así, se habrían observado varios picos de transmisión separados en el espectro, en lugar de un único pico centralizado como se muestra en la Figura 7. En esta etapa se ajustaron valores como el ancho de banda (100kHz) y la ganancia (~30), con el fin de obtener una representación clara en la herramienta Universal Radio Hacker.

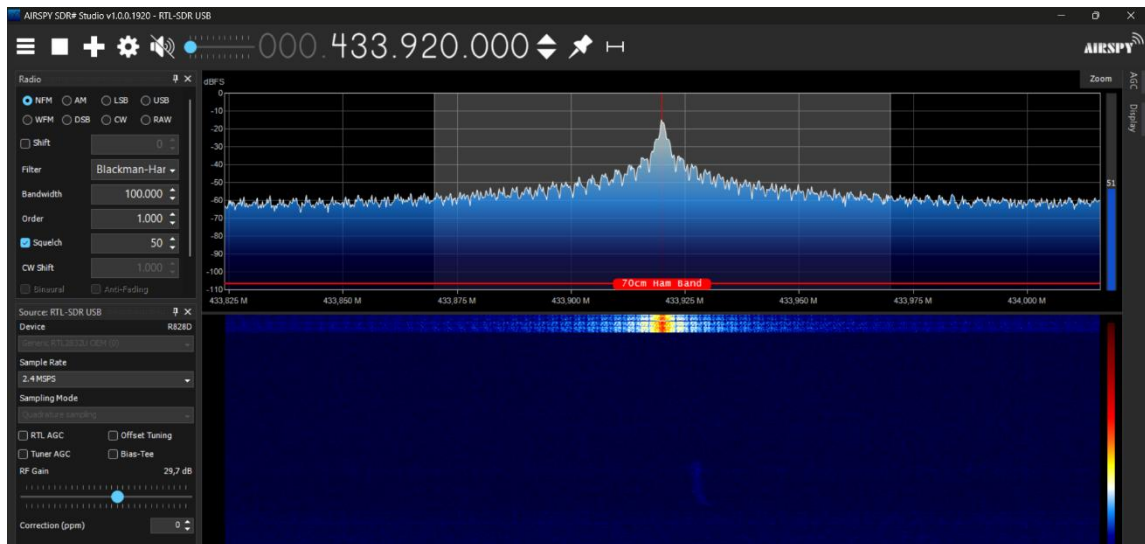


Figura 6. Captura de pantalla SDRSharp

A continuación, se empleó **Universal Radio Hacker (URH)** para grabar la señal capturada empleando los valores obtenidos mediante SDRSharp. La interfaz de URH permitió visualizar que cada comando del mando remoto se componía de una secuencia de pulsos y pausas repetida varias veces como se puede observar en la Figura 7. Es posible diferenciar dos señales distintas en una misma trama, la principal y las de repetición, dichas repeticiones se transmitían cada 8.850 ms, lo que indica cómo el transmisor RF original envía la misma trama varias veces para asegurar la recepción por parte de la luz remota.



Figura 7. Captura URH: Señal Completa Correspondiente a Botón de Encendido y Apagado

Decodificación de pulsos y reproducción en el RoomNode

Se observó que la señal se componía mayoritariamente de pulsos y pausas de 350 μ s o 1.06ms (aprox. 3 veces 350ns), conformando combinaciones de “corto” y “largo” que representan bits como se puede observar en la Figura 8.

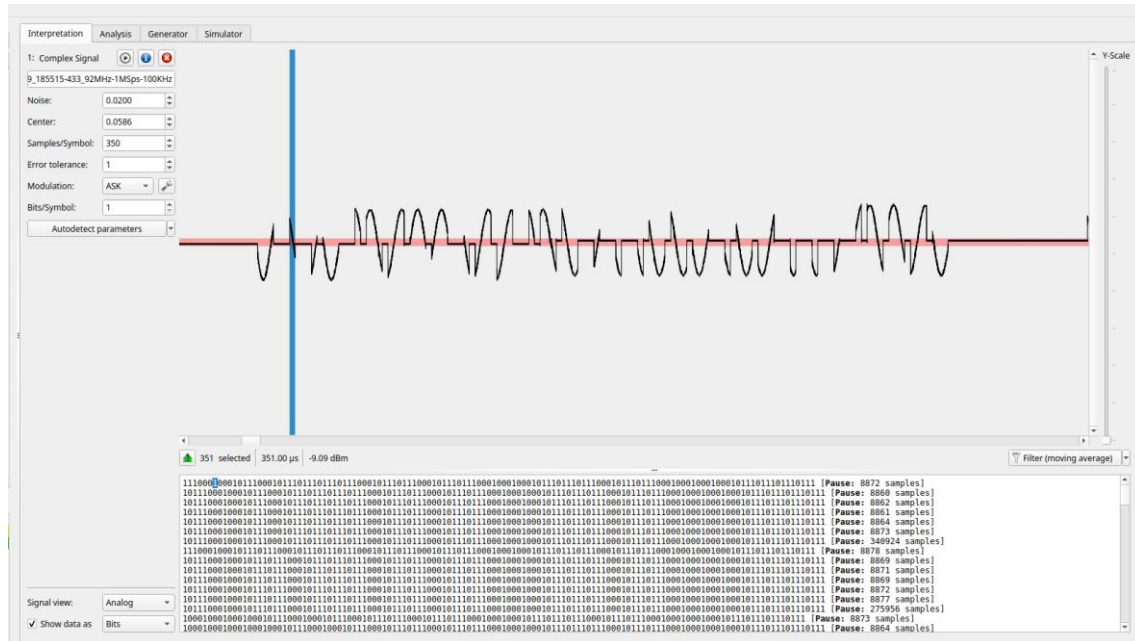


Figura 8. Captura URH: Ampliación Señal Principal Botón Encendido/Apagado

De forma resumida:

- Una combinación de “pausa corta + pulso largo” probablemente se interpretase como ‘1’,
- Mientras que “pausa larga + pulso corto” podría equivaler a ‘0’.

Después de decodificar la señal mediante dígitos de duración de 350ns se pudo observar que algunas señales tenían un preámbulo de un único ‘1’ mientras que otras arrancaban con tres ‘1’ consecutivos (‘111’). Además, para generar la repetición de la señal se introducía al inicio un patrón adicional diferente según el caso:

- Si la trama principal inicia por un solo '1', la repetición se antecede por ‘1000’, seguida de la trama principal.
- En cambio, si la trama principal comienza por tres '1' consecutivos (111), la repetición se inicia con ‘01’ concatenado con la trama principal.

Si bien la razón exacta de este comportamiento no es esencial para la clonación, se estima que la variación en esos bits iniciales sirve para distinguir la primera transmisión de las subsecuentes repeticiones y para la detección y recuperación de errores en el lado del receptor. Lo cual podría resultar útil en un futuro, si se incorpora la funcionalidad de interactuar con las luces con el mando remoto (además de con la interfaz web).

Tras la identificación de la secuencia de cada señal, URH permitió descomponer la trama en un array binario, donde cada dígito se corresponde con una duración de 350 μ s, indicando en cada instante si se está transmitiendo portadora (“1”) o se mantiene apagada (“0”).

Con la secuencia binaria de cada comando, se programó el RoomNode para transmitir o apagar el módulo RF durante 350ns según lo indicase cada bit.

4.3 Hardware

4.3.1 Selección de Microcontroladores

En este proyecto se ha decidido emplear microcontroladores Espressif ESP32 debido a su variedad de productos, soporte nativo para Wi-Fi y ESP-NOW, y un enfoque marcado en la eficiencia energética con una relación calidad-precio muy competitiva. Estas características permiten adaptar cada dispositivo a los requisitos específicos de cada nodo.

4.3.1.1 MasterDevice: ESP32-S3

El ESP32-S3 ha sido seleccionado para el MasterDevice debido a sus **capacidades avanzadas** que lo hacen adecuado para la coordinación centralizada del sistema. Este modelo ofrece un procesador dual-core, con una velocidad de reloj de hasta 240MHz, idóneo para gestionar tareas complejas simultáneamente como la sincronización NTP, la ejecución de una interfaz web y la comunicación eficiente con otros nodos mediante el protocolo ESP-NOW.

Adicionalmente, el ESP32-S3 cuenta con una memoria SRAM de 512 KB, esta elevada capacidad permite:

- Ejecutar un mayor número de tareas de FreeRTOS simultáneamente, ya que hay suficiente memoria para asignar stacks individuales a cada tarea.
- Crear tareas con stacks más grandes, necesarias para operaciones que impliquen el manejo de datos voluminosos o cálculos complejos.
- Asegurarse de tener suficiente memoria reservada para otros procesos críticos, como la comunicación Wi-Fi o el manejo de buffers, sin agotar los recursos del sistema.

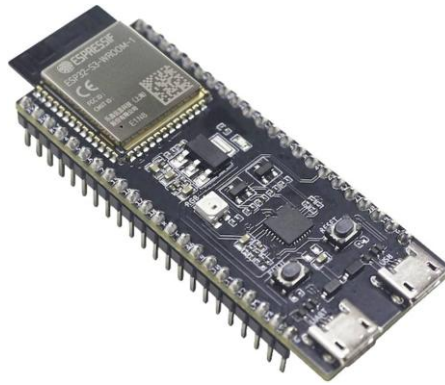


Figura 9. ESP32-S3

4.3.1.2 RoomNode: ESP32 Dev Kit C V4

El ESP32 Dev Kit C V4 ha sido seleccionado para los RoomNodes debido a su **versatilidad**, extensa documentación y equilibrio entre rendimiento y coste. Este microcontrolador soporta la ejecución de FreeRTOS, lo que permite gestionar tareas en paralelo de manera eficiente. Además, cuenta con una amplia gama de GPIOs y múltiples interfaces de comunicación, como UART, SPI, I2C e I2S, lo que facilita la integración de sensores, emisores IR, transmisores RF y otros módulos. Estas características también ofrecen flexibilidad para incorporar nuevas funcionalidades en el futuro, adaptándose a las necesidades cambiantes del proyecto.

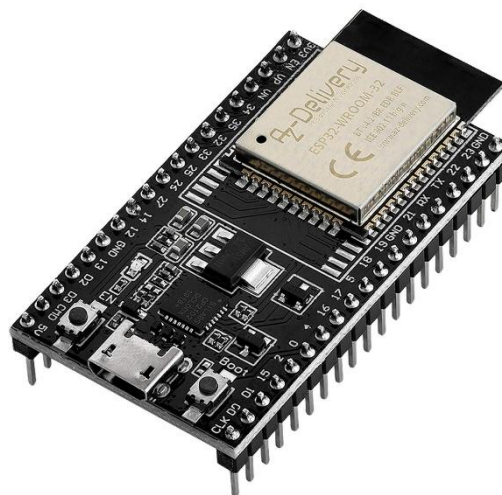


Figura 10. ESP32 Dev Kit C V4

4.3.1.3 SensorNode: LOLIN D32

Para los SensorNodes, se ha seleccionado el LOLIN D32 V1.0.0, un microcontrolador optimizado para aplicaciones de bajo consumo. Este modelo destaca por su excelente **eficiencia energética**, gracias a la eficiencia logrando un consumo de tan solo 70 μA en modo *Deep Sleep* [15], [16]. Esta característica es crucial para nodos alimentados por batería, maximizando su autonomía y reduciendo la frecuencia de recargas o sustituciones.

La placa LOLIN D32 es compatible con la alimentación por batería, mediante su conector JST o su pin BAT. El regulador de carga incorporado permite cargar la batería mediante el puerto MicroUSB. Adicionalmente, dispone de 2 interfaces I2C, que permiten la comunicación con el módulo sensor SHT31.

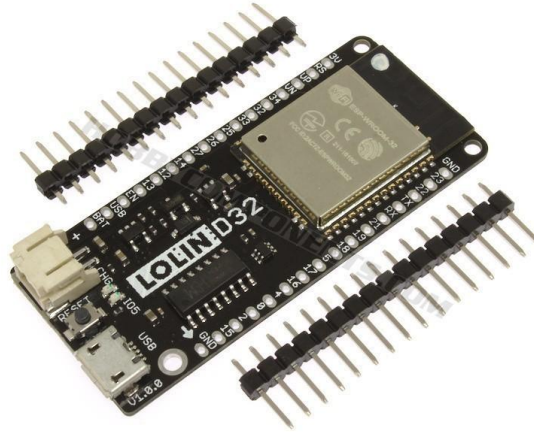


Figura 11. Wemos LOLIN D32

4.3.2 Componentes del Sistema

A fin de complementar las limitaciones de los microcontroladores y proporcionar las funcionalidades requeridas por cada nodo (detección de presencia, control de luces RF, emisión infrarroja, etc.), se han incorporado varios módulos de hardware. A continuación, se describen brevemente las características más relevantes de cada uno.

4.3.2.1 Radar LD2410C (24 GHz)

Para la detección de presencia se utiliza el sensor HLK-LD2410C[17], un radar de modulación FMCW (24 GHz) capaz de reconocer movimientos, posturas estáticas y micromovimientos humanos en un rango de hasta 5 metros. Esto resulta fundamental para automatizar acciones en las habitaciones, como la activación o el apagado de luces y aire acondicionado.

Gracias a la tecnología FMCW, el sensor aprovecha el desplazamiento Doppler para medir con relativa precisión la distancia hasta el objetivo.

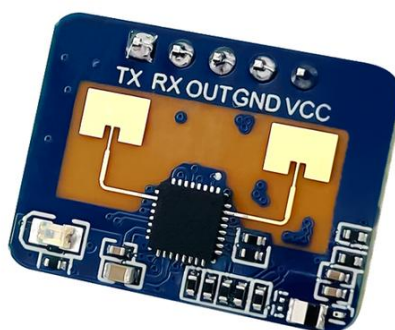


Figura 12. Sensor de Presencia LD2410C

Tiene dimensiones reducidas (16 x 22mm) y funciona correctamente tanto en techos como en paredes, siempre que la antena del módulo esté orientada hacia la zona a cubrir. Además, ofrece configuración por UART para ajustar parámetros como la sensibilidad o el tiempo de “no-one duration” (el lapso de ausencia antes de reportar “sin presencia”), lo que incrementa la fiabilidad al determinar que una persona ya no se encuentra en el área.

En este proyecto no se emplea la función de medición de distancia ni se realizan ajustes adicionales al sensor, dado que la configuración predeterminada (un “no-one duration” de 5 segundos y una sensibilidad del 50%) resulta suficiente para la detección de presencia en la habitación.

Su principal inconveniente es la alimentación a **5V** y requerimiento de una capacidad de corriente mayor a 200mA, que impide su alimentación directa con el ESP32 y provoca la necesidad de aumentar el tamaño del proyecto incorporando un regulador de corriente capaz de suministrar la cantidad de potencia requerida.

El módulo tiene los pines descritos en la siguiente tabla:

Pin	Nombre	Función
1	UART_Tx	Transmisión de datos serie.
2	UART_Rx	Recepción de datos serie.
3	OUT	Salida de estado de presencia (alto si detecta, bajo si no).
4	GND	Conexión a tierra.
5	VCC	Entrada de alimentación (5V).

Tabla 2. Pinout del módulo LD2410C

Elección del Sensor de Presencia

La detección de presencia es un desafío común en la domótica, sobre todo cuando la persona permanece sentada o quieta durante períodos prolongados. Los sensores PIR (Passive Infrared) suelen emplearse en aplicaciones donde solo se requiere detectar movimiento puntual, dado que pueden fallar a la hora de mantener encendida una luz si el usuario permanece casi inmóvil (e.g., trabajando en un escritorio). Otras alternativas, como la detección Bluetooth, son útiles cuando el usuario porta constantemente un móvil o reloj inteligente, pero no resulta viable en este proyecto, puesto que no se garantiza que el usuario en todo momento.

En el mercado existen soluciones que combinan varios métodos de sensorización (PIR, radar, ultrasonido, Bluetooth, entre otros) y algoritmos de fusión de datos para determinar la presencia con mayor fiabilidad. Sin embargo, estos enfoques suelen aumentar la complejidad y el coste del sistema.

El **LD2410C** se seleccionó por su capacidad para abordar este problema de manera **eficiente y asequible**, sin complejidad excesiva. Su capacidad para detectar micromovimientos y ocupación inmóvil lo hace ideal para las necesidades específicas de este proyecto.

4.3.2.2 FS1000A Transmisor RF (433 MHz)

Los RoomNodes incorporan un transmisor RF[18] de 433 MHz para las **luces remotas** del dormitorio. Este módulo utiliza modulación por amplitud (ASK), concretamente modulación por apagado y encendido (OOK), que es un caso particular de ASK dónde la amplitud de la señal toma dos únicos valores, apagado y encendido. De esta forma, este módulo permite enviar señales de control, replicando las órdenes del mando remoto original de las luces.



Figura 13. FS1000A: Transmisor RF 433MHz

El transmisor FS1000A opera en la banda ISM de 433.92 MHz, frecuentemente empleada en dispositivos de hogar (enchufes inteligentes, interruptores, etc.), garantizando compatibilidad y potencial para futuras expansiones. Funciona enviando pulsos desde el microcontrolador al pin DATA, modulando la señal para que sea interpretada correctamente por los receptores originales.

Este módulo puede alimentarse con voltajes entre 3 V y 12 V, aunque el alcance varía según la tensión de alimentación y el tipo de antena. En el presente proyecto, se alimenta a 3.3V del ESP32, pues la distancia a cubrir es corta y no se requieren grandes potencias de transmisión.

El módulo RF tiene los pines descritos en la siguiente tabla:

Pin	Función
DATA	Entrada de datos que modulan la señal de transmisión ASK.
GND	Conexión a tierra.
VCC	Alimentación (3-12V)

Tabla 3. Pinout del Transmisor RF

4.3.2.3 Emisor Infrarrojo Doble

Para controlar los equipos de aire acondicionado, que funcionan con mandos a distancia infrarrojos, se emplea un emisor IR con dos diodos emisores (longitud de onda de 940 nm). Al

disponer de **dos diodos**, se **mejora la cobertura y la fiabilidad** al enviar la señal IR, abarcando un ángulo mayor.

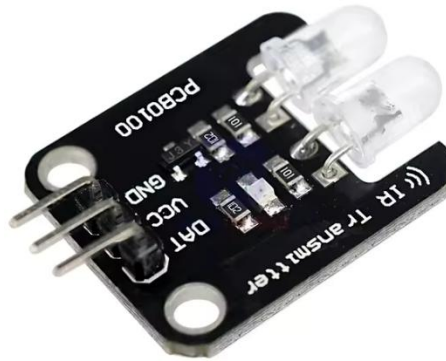


Figura 14. Módulo Emisor IR Dual

El principio de funcionamiento es similar al del transmisor RF: el microcontrolador genera en el pin DATA trenes de pulsos modulados a la frecuencia apropiada (en nuestro caso, 36.7kHz) que luego se emiten en infrarrojo, replicando así las señales del mando original.

El módulo IR cuenta con los siguientes pines:

Pin	Función
DATA	Señal de entrada para emitir luz infrarroja.
GND	Conexión a tierra.
VCC	Alimentación (5V)

Tabla 4. Pinout Transmisor RF

4.3.2.4 Sensor de Luz TSL2591

Para medir con mayor **precisión** la **luminosidad** ambiental, en lugar de un LDR que ofrece una respuesta más lenta y menos precisa, se utiliza el sensor TSL2591[19]. Este dispositivo es un sensor de luz digital de alta sensibilidad que se comunica mediante el bus **I2C**, ofreciendo lecturas de hasta 88,000 lux, abarcando así un amplio rango de condiciones lumínicas.



Figura 15. Sensor de Luz TSL2591

El TSL2591 integra un fotodiodo de infrarrojo y uno de espectro total, calculando la intensidad de luz ambiente a partir de ambos canales. También incluye compensaciones y ajustes internos que proporcionan mediciones muy estables, facilitando la calibración y la lectura directa de lux sin requerir complejos cálculos.

Además de verificar si la emisión de luz ha aumentado o disminuido correctamente tras los comandos RF, el TSL2591 posibilita la implementación de lógicas de control de brillo y ajustes automáticos en función de la iluminación del entorno.

El sensor TSL2591 presenta los siguientes pines:

Pin	Función
VIN	Alimentación (3V)
GND	Conexión a tierra.
SCL	Línea de reloj del bus I2C
SDA	Línea de datos del bus I2C
INT	Opcional: Pin de interrupción.
3V0	Pin auxiliar: Salida de voltaje

4.3.2.5 Baterías LiPo (3.7 V, 1000 mAh)

En los SensorNodes se emplean baterías de polímero de litio (LiPo) por su excelente relación entre densidad de energía, peso y tamaño. Para este proyecto, se ha seleccionado una capacidad de 1000 mAh, cuyas dimensiones son similares al microcontrolador LOLIN D32, microcontrolador empleado para los SensorNodes. Tiene un voltaje nominal de 3.7 V, que es el voltaje soportado por el LOLIN D32 para la alimentación a través del pin.



Figura 16. Batería LiPo

La recarga de las baterías se puede realizar mediante el puerto MicroUSB de la ESP32, gracias a el regulador de carga (LTH7) que incorpora la placa LOLIN D32, encargado de regula la corriente y el voltaje enviados a la batería para cargarla de manera segura y eficiente.

En condiciones estándar de consumo para este SensorNode, las estimaciones sugieren que la batería puede ofrecer hasta **452 días de autonomía**, tal como se detalla en el Anexo II: Cálculo de la Duración de la Batería, donde se describen los cálculos y supuestos realizados para llegar a esta cifra aproximada.

Pin	Función
B+	Terminal positivo de la batería.
B-	Terminal negativo de la batería.

Tabla 5. Pinout Batería Lipo

4.3.2.6 Sensor de Temperatura y Humedad SHT31

Para la medición de temperatura y humedad, se incorpora un sensor digital SHT31[20]. Este dispositivo comunica con el microcontrolador mediante la interfaz **I2C** y es capaz de registrar temperatura y humedad de forma **estable** y con **bajo consumo**. El SHT31 integra en su interior celdas capacitivas y resistivas, calibradas de fábrica, ofreciendo datos con precisiones típicas de $\pm 0.3\text{ }^{\circ}\text{C}$ en temperatura y $\pm 2\text{ }\%$ en humedad.

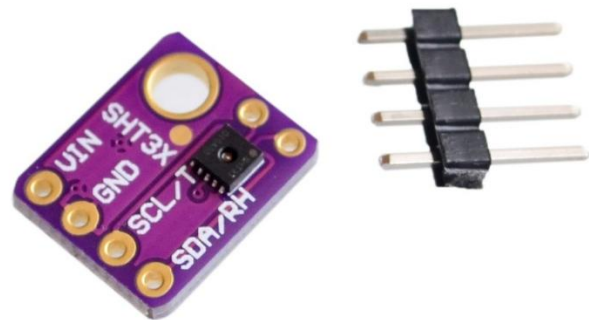


Figura 17. Sensor SHT31

Sus características se muestran de manera resumida en la siguiente tabla:

Parámetro	Valor
Temperatura	
Rango de operación	-40 °C a 125 °C
Precisión típica	± 0.3 °C*
Humedad relativa	
Rango de operación	0 % a 100 %
Precisión típica	± 2 %*
Consumo de corriente	
Durante la medición	600 μ A típicos
Duración de la medición	2.5 ms (baja repetibilidad),
En reposo	0.2 μ A
Alimentación	
Voltaje de operación	2.15 V a 5.5 V

**Valores típicos de precisión, pueden variar dependiendo del rango de medida.*

Tabla 6. Especificaciones SHT31

El módulo SHT31 presenta los siguientes pines:

Pin	Función
VIN	Entrada de Alimentación (3.3V)
GND	Referencia a Tierra.
SDA	Línea de Datos del bus I2C.
SCL	Línea de Reloj del bus I2C.

Tabla 7. Pinout SHT31

4.3.3 Conexión y Cableado

En las siguientes figuras se ilustran los esquemas de conexión del RoomNode y del SensorNode, ambos realizados con el software Fritzing. En el caso del MasterDevice, su conexión es mínima: se trata de un ESP32-S3 (Figura 9) que se alimenta a 5V, sin requerir módulos adicionales.

A la hora de la selección de pines del microcontrolador ESP32 se ha tenido en cuenta que algunos de ellos no se deben utilizar o presentan comportamientos especiales (e.g., pines reservados para

Wi-Fi, arranque, flasheo o con funcionalidad únicamente de entrada), para conocer más detalles ver Anexo IV: Pines Disponibles en el ESP32.

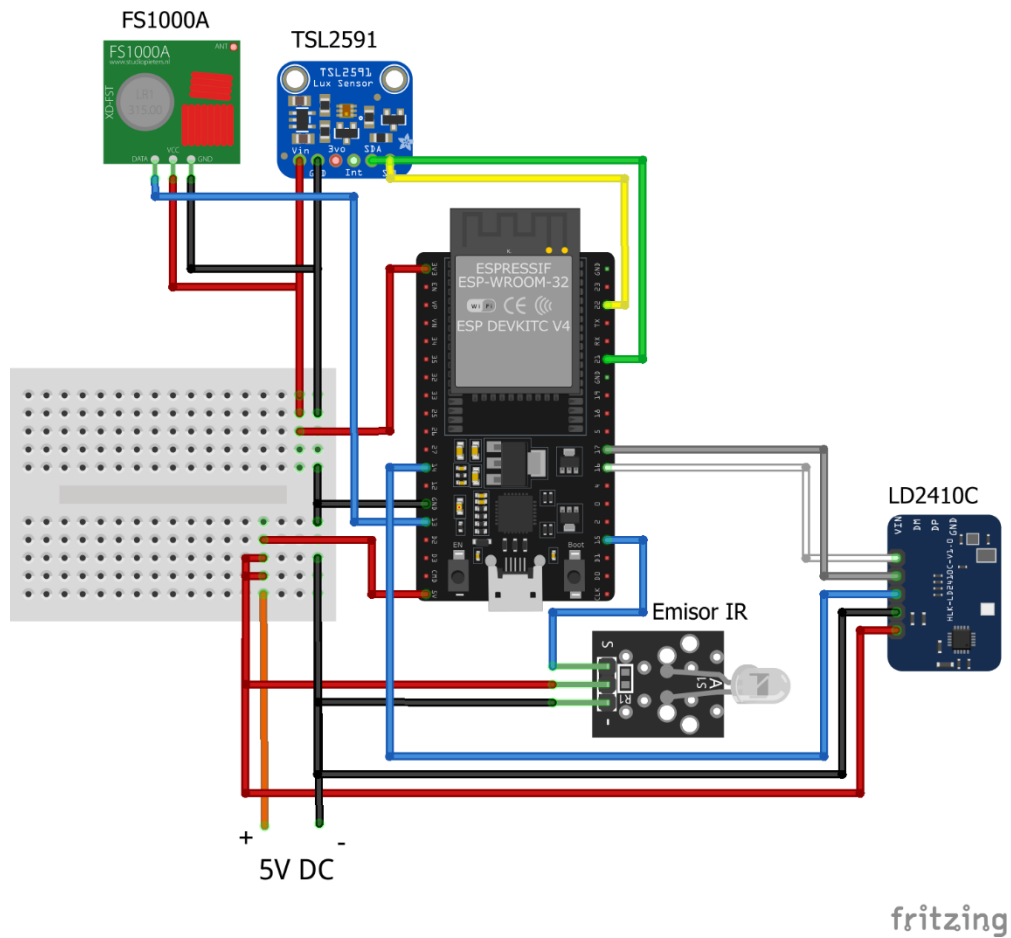


Figura 18. Diagrama Esquemático del RoomNode

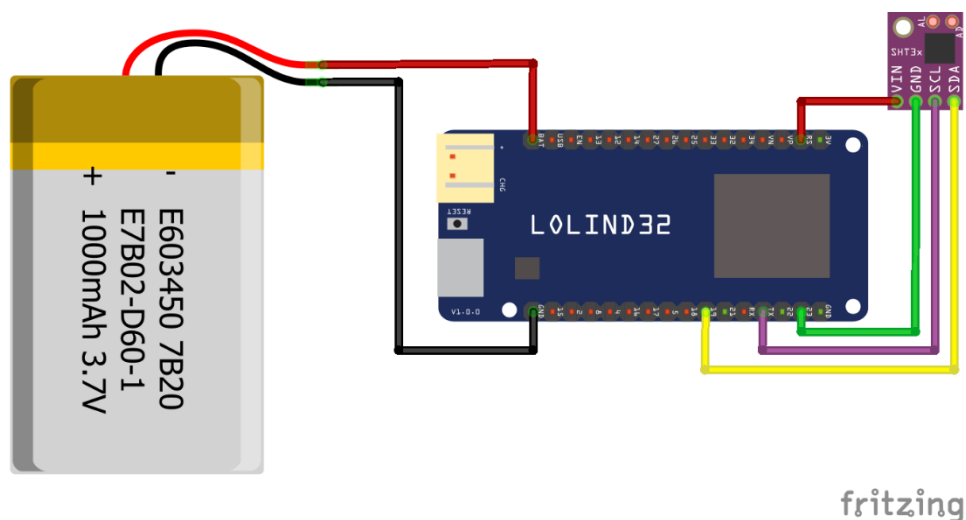


Figura 19. Diagrama Esquemática del SensorNode.

Para detalles completos sobre las conexiones realizadas en el RoomNode y el SensorNode, incluyendo los pines específicos asignados para cada módulo, consultar el Anexo V: Tabla de conexiones del RoomNode y SensorNode.

4.4 Software

4.4.1 Entorno de Desarrollo

Para llevar a cabo el desarrollo del software se han empleado varios entornos y herramientas que han permitido, por un lado, una configuración óptima de los microcontroladores y, por otro, un control preciso de la evolución del código.

4.4.1.1 IDEs Utilizados

Visual Studio Code + PlatformIO

Se ha empleado el conocido IDE Visual Studio Code junto a la extensión de PlatformIO para el desarrollo del proyecto principal. PlatformIO proporciona una integración nativa con el ecosistema Arduino/ESP32, lo que facilita tanto la instalación como la actualización de las librerías necesarias, además de simplificar la compilación y la carga del firmware en los diferentes modelos de microcontroladores.

Dentro de PlatformIO, se definieron entornos específicos para cada uno de los nodos diseñados en el sistema (SensorNode, RoomNode y MasterDevice). Esta segmentación permitió gestionar de forma independiente dependencias, configuraciones y parámetros de compilación para cada nodo en un mismo repositorio, dotando no solo de claridad y orden sino también ofreciendo la compartición de módulos y archivos de configuración.

Arduino IDE

El Arduino IDE fue utilizado durante el comienzo del proyecto, pero debido a su falta de soporte de modularidad se decidió desarrollar el código en un entorno más avanzado, como es VSCode. Durante el resto del desarrollo se utilizó de forma complementaria, fundamentalmente para la rápida ejecución de ejemplos de librerías y la validación inicial de pequeños sketches de prueba.

Resultó particularmente útil en experimentos de **medición de tiempos**, ya que el monitor serie de Arduino IDE incluye una marca de tiempo que facilita el registro aproximado de eventos en el orden de milisegundos.

Además, sirvió como alternativa de respaldo cuando era necesario verificar la compatibilidad del código con ciertas librerías o para realizar comprobaciones puntuales cuando surgían dudas acerca de la configuración en PlatformIO.

4.4.1.2 Control de Versiones con Git y GitHub

La gestión del código se centralizó en un repositorio público de GitHub (<https://github.com/luismorenogv/efficient-home-automation-esp32>), donde se puede observar el camino recorrido durante el desarrollo del proyecto. El uso de Git y GitHub ofreció la ventaja de

llevar un registro histórico de los cambios y la posibilidad de restaurar estados anteriores cuando surgían errores o se requería evaluar versiones previas.

En las primeras etapas del desarrollo, se creaba una rama separada para cada funcionalidad (e.g., feature/START, sync_v1, sync_v2). Este esquema de trabajo garantiza aislar los cambios y probarlos sin afectar la rama principal del proyecto. Sin embargo, a medida que el proyecto evolucionó se estimó que utilizar esta metodología resultaba demasiado detallada para un proyecto personal.

Por ello, se optó por una estructura de ramas más simple, integrada por dos ramas principales:

- **main:** Contiene la versión estable del código. Solo se fusionan los cambios a esta rama cuando han sido probados y validados con éxito.
- **develop:** Orientada al desarrollo. En esta rama se introducen y prueban nuevas características antes de verificar su estabilidad para incorporarlas a main.

Este flujo de trabajo equilibró la necesidad de mantener un código confiable (en main) con la agilidad de desarrollar y validar nuevas funcionalidades de forma iterativa (en develop).

4.4.2 Librerías Empleadas

El proyecto hace uso de una combinación de librerías estándar (propias del entorno Arduino y de la plataforma ESP32) y librerías específicas para manejar componentes de hardware, servicios web y protocolos de comunicación. A continuación, se describen las principales librerías y la funcionalidad que aportan al sistema.

4.4.2.1 Librerías estándar de Arduino y ESP32

- **<Arduino.h>:** Provee las definiciones básicas y funciones esenciales del entorno Arduino (como digitalWrite, pinMode, etc.). Empleada para un control sencillo e intuitivo sobre los pines del microcontrolador.
- **<WiFi.h>:** Permite configurar y controlar la interfaz Wi-Fi en los ESP32. Incluye métodos para conectarse a redes, desconectarse, verificar el estado de la conexión y obtener información (e.g., dirección IP, calidad de la señal). Para este proyecto, se utiliza mayormente en el MasterDevice para ofrecer la interfaz web y sincronizar la hora mediante NTP.
- **<esp_wifi.h>:** Ofrece acceso a las funcionalidades de bajo nivel de la interfaz Wi-Fi del chip ESP32. Se emplea, por ejemplo, para fijar un canal Wi-Fi específico o gestionar parámetros avanzados que no están disponibles en la librería de alto nivel (WiFi.h). Esto resulta esencial en el RoomNode y SensorNode cuando se configura el canal de comunicación para ESP-NOW, ajustándolo al utilizado por el MasterDevice.

- **<esp_now.h>**: Implementa el protocolo ESP-NOW, ofreciendo funciones y definiciones esenciales para la correcta inicialización, envío y recepción de mensajes.
- **<time.h>**: Proporciona mecanismos para gestionar la fecha y la hora. Esta librería es fundamental para la sincronización con servidores NTP, permitiendo que los nodos dispongan de una referencia horaria precisa.

4.4.2.2 Librerías Específicas de Sensores y Hardware

- **<Adafruit_TSL2591.h>**: Adafruit_TSL2591 [21] ofrece soporte para el sensor de luz TSL2591, empleado por el RoomNode.
- **<Adafruit_SHT31.h>**: Adafruit_SHT31 [22] permite interactuar con el sensor de temperatura y humedad SHT31, que se comunica por medio de I2C.
- **<IRremoteESP8266.h>**: Esta librería [13] se usa para enviar y decodificar señales de infrarrojo, ofrece una gran variedad de protocolos propietarios de aires acondicionados. El RoomNode se basa en ella para enviar una trama IR que replican la señal de apagado del mando a distancia del equipo de climatización. La librería ofrece el conjunto completo de comandos disponibles en el mando a distancia original, que abre las puertas a futuras implementaciones más avanzadas del aire acondicionado.
- **<MyLD2410.h>**: MyLD2410 [23] es una librería diseñada específicamente para interactuar con el sensor de presencia LD2410 y sus distintas versiones. Este sensor, empleado en el RoomNode, utiliza comunicación UART para configurar parámetros avanzados y detectar cambios en el estado de presencia en tiempo real. La librería abstrae la comunicación UART con el sensor y provee de una interfaz sencilla para la interacción con este.

4.4.2.3 Librerías de comunicación web

- **<ESPAsyncWebServer.h>**: Esta librería [24] implementa un servidor web asíncrono en el MasterDevice. La principal ventaja de la asincronía radica en que el ESP32 puede atender múltiples peticiones sin bloquearse: si se recibe una solicitud HTTP (o WebSocket) mientras se está procesando otra, el microcontrolador no queda inactivo hasta terminar la anterior, sino que gestiona todo mediante eventos en paralelo. El resultado es una experiencia más fluida para el usuario al interactuar con la interfaz web cuando hay múltiples personas conectadas al servidor.
- **<ArduinoJson.h>**: Facilita la serialización y deserialización de datos en formato JSON. Este formato se utiliza para intercambiar datos entre la interfaz web (*front-end*) y el MasterDevice(*back-end*) en tiempo real.

4.4.3 Estructura del Código

Para administrar de forma organizada los diferentes módulos (MasterDevice, RoomNode y SensorNode), así como las librerías comunes y archivos de configuración, se ha adoptado una estructura de proyecto basada en PlatformIO y en la separación clara de funciones en módulos. Esto permite mantener la coherencia y escalabilidad del código a medida que el sistema crece. A continuación, se presenta una descripción general de la estructura de directorios y la lógica que sigue cada uno de ellos:

4.4.3.1 Archivo de Configuración de Proyecto

- **platformio.ini:** Contiene la configuración principal del proyecto en PlatformIO, especificando:
 - La plataforma y el framework (Arduino) usados para compilar y vincular el proyecto.
 - Las distintos entornos (e.g., env:room, env:master, env:sensor), cada uno correspondiente a un modelo de microcontrolador específico.
 - Las librerías que deben incluirse y las banderas de compilación (build_flags), empleadas para realizar definiciones en tiempo de compilación y especificar directorios a incluir en dicho proceso.

4.4.3.2 Carpeta data/

Contiene los archivos estáticos para la interfaz web (index.html, script.js, style.css y favicon.png). Estos se cargan en el sistema de archivos LittleFS (Ver más detalles en Anexo VI: LittleFS), y son servidos por el MasterDevice cuando un usuario accede a la interfaz web.

4.4.3.3 Carpeta config/

Esta carpeta contiene archivos esenciales para la configuración global del sistema y utilidades comunes. A continuación, se detallan los archivos más relevantes:

- **config.h:** Define constantes globales que incluyen pines asignados a sensores y módulos, periodicidad de las tareas concurrentes o parámetros de compilación. También define la bandera de compilación ENABLE_DEBUGGING, que permite activar o desactivar la depuración en todo el sistema.
- **Logger.h:** Proporciona macros de registro de mensajes (LOG_INFO, LOG_WARNING, LOG_ERROR), utilizadas para imprimir información de diagnóstico en el puerto serie durante el desarrollo y las pruebas. Está diseñado de forma que si ENABLE_DEBUGGING no está definido, las macros se convierten en instrucciones vacías, mejorando el rendimiento durante la operación.

4.4.3.4 Carpeta include/

Estructurada en subdirectorios que agrupan las clases y cabeceras según el nodo o la lógica que implementen:

- **Common/**: Contiene cabeceras compartidas entre todos los nodos, como:
 - **common.h**: Define enumeraciones y estructuras de datos globales (e.g., **MessageType**, **IncomingMsg**, **AllMessages**), asegurando la compatibilidad de la comunicación ESP-NOW.
 - **CommunicationsBase.h**, **NTPClient.h**, etc.: Declaración de clases que se usan en dos o más tipos de nodo (**SensorNode**, **MasterDevice** o **RoomNode**).
- **MasterDevice/**: Contiene archivos de cabecera para la declaración de clases empleadas en el **MasterDevice**: **MasterController.h**, **DataManager.h**, **MasterCommunications.h**, **WebServer.h** y **WebSockets.h**.
- **RoomNode/**: Contiene archivos de cabecera para la declaración de clases empleadas en el **MasterDevice**: **RoomNode.h**, **Lights.h**, **LD2410.h**, **RoomCommunications.h** y **AirConditioner.h**.
- **SensorNode/**: Contiene archivos de cabecera para la declaración de clases empleadas en el **MasterDevice**: **ESPNowHandler.h**, **PowerManager.h**, **SHT31Sensor.h**, **SensorNode.h**

4.4.3.5 Carpeta src/

Aquí se ubican las implementaciones (.cpp) correspondientes a las cabeceras de la carpeta include/. Se subdivide en varios subdirectorios:

Subdirectorio Common/

- **CommunicationsBase.cpp**: Este módulo proporciona una clase base para las comunicaciones mediante ESP-NOW. Define métodos comunes a todos los nodos, como la inicialización de ESP-NOW, el envío y la recepción de mensajes, y la gestión de errores. Es heredado y extendido por los módulos de comunicación específicos de cada nodo (e.g., **MasterCommunications**, **RoomCommunications**, **ESPNowHandler**).
- **NTPClient.cpp**: Implementa una clase para gestionar la sincronización de hora mediante servidores NTP (Network Time Protocol).

Subdirectorio MasterDevice/

- **MasterController.cpp**: Este módulo implementa la clase principal del **MasterDevice**, encargada de orquestar todas las tareas y coordinar los diferentes módulos. Sus funciones principales son inicializar y configurar los distintos módulos, además de crear y coordinar las tareas concurrentes mediante FreeRTOS.
- **DataManager.cpp**: Encargado de gestionar el almacenamiento y procesamiento de la información recibida de los nodos de forma segura implementando semáforos.

- **MasterCommunications.cpp:** Implementa la lógica de comunicación del MasterDevice con los nodos SensorNodes y RoomNodes. Deriva de la clase CommunicationsBase, en base a la cual modifica y añade métodos, ajustando las necesidades de comunicación al MasterDevice.
- **WebServer.cpp:** Proporciona la lógica para manejar el servidor web del MasterDevice. Configura y aloja un servidor HTTP basado en la librería ESPAsyncWebServer, además de gestionar los archivos estáticos (HTML, CSS, JavaScript) y recursos enviados al navegador.
- **WebSockets.cpp:** Implementa la funcionalidad de comunicación en tiempo real entre el MasterDevice y el navegador del usuario mediante WebSockets. Se encarga de recibir y procesar las peticiones enviadas por el usuario desde el frontend.

Subdirectorío RoomNode/

- **RoomNode.cpp:** Este módulo implementa la clase principal del RoomNode, encargada de integrar y coordinar los diferentes módulos en la habitación. Inicializa los módulos asociados al nodo y ejecuta las tareas concurrentes relacionadas con la automatización de luces y aire acondicionado mediante FreeRTOS.
- **Lights.cpp:** Contiene la lógica para controlar las luces RF de la habitación. Implementa métodos para enviar señales RF a través del módulo FS1000A. Adicionalmente, se encarga de manejar el sensor de luz TSL2591, obteniendo mediciones precisas de luminosidad del entorno. En estas mediciones de luminosidad se basan mecanismos de validación y reintento que incluye también el módulo, para garantizar el éxito de los comandos enviados.
- **LD2410.cpp:** Implementa la comunicación y configuración del sensor de radar LD2410. Incluye métodos para inicializar el sensor y configura la lectura del sensor mediante interrupción.
- **RoomCommunications.cpp:** Este módulo se encarga de gestionar la comunicación del RoomNode con el MasterDevice. Deriva de la clase CommunicationsBase, adaptando y extendiendo su funcionalidad para ajustarse a las necesidades del nodo.
- **AirConditioner.cpp:** Contiene la lógica para enviar comandos IR al aire acondicionado basándose en la librería IRremoteESP8266.

Subdirectorío SensorNode/

- **SensorNode.cpp:** Este módulo implementa la clase principal del SensorNode. Se encarga de inicializar y coordinar los módulos del nodo, gestionando la toma de datos ambientales, la comunicación con el MasterDevice y la transición entre estados de bajo consumo (*Deep Sleep*) y operación.

- **SHT31Sensor.cpp**: Implementa la lógica para interactuar con el sensor de temperatura y humedad SHT31 basado en la librería Adafruit_SHT31. Proporciona métodos para inicializar el sensor, realizar mediciones y gestionar posibles errores durante las lecturas.
- **PowerManager.cpp**: Gestiona las funciones y variables relacionadas con el modo de bajo consumo (*Deep Sleep*) del SensorNode.
- **ESPNowHandler.cpp**: Este módulo extiende la clase CommunicationsBase para adaptarse a las necesidades específicas del SensorNode. A diferencia del RoomNode y MasterDevice, que delegan el procesamiento de los mensajes recibidos a una tarea concurrente, ESPNowHandler gestiona la recepción de mensajes en su propia ISR.

4.4.4 Flujo del Programa

En este apartado se provee de diagramas de flujo para los diferentes nodos del sistema (SensorNode, RoomNode y MasterDevice). Estos diagramas ilustran de forma esquemática mediante frases descriptivas cómo cada uno de los nodos realiza sus funciones esenciales. De este modo, se proporciona un contexto visual que complementa las descripciones presentadas en los apartados siguientes.

A continuación, se presentan los diagramas de flujo de cada nodo:

4.4.4.1 SensorNode

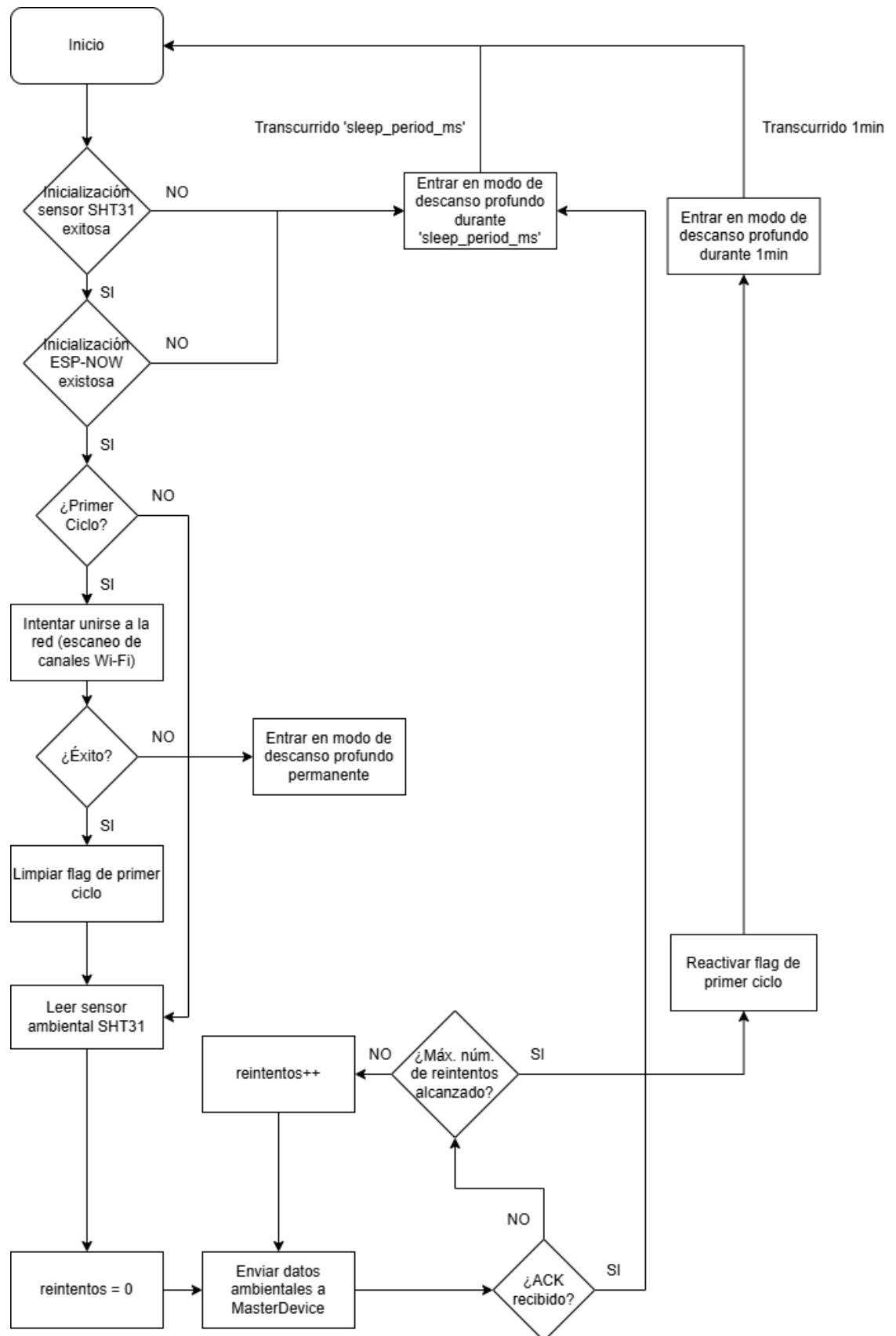


Figura 20. Diagrama de Flujo SensorNode

4.4.4.2 RoomNode

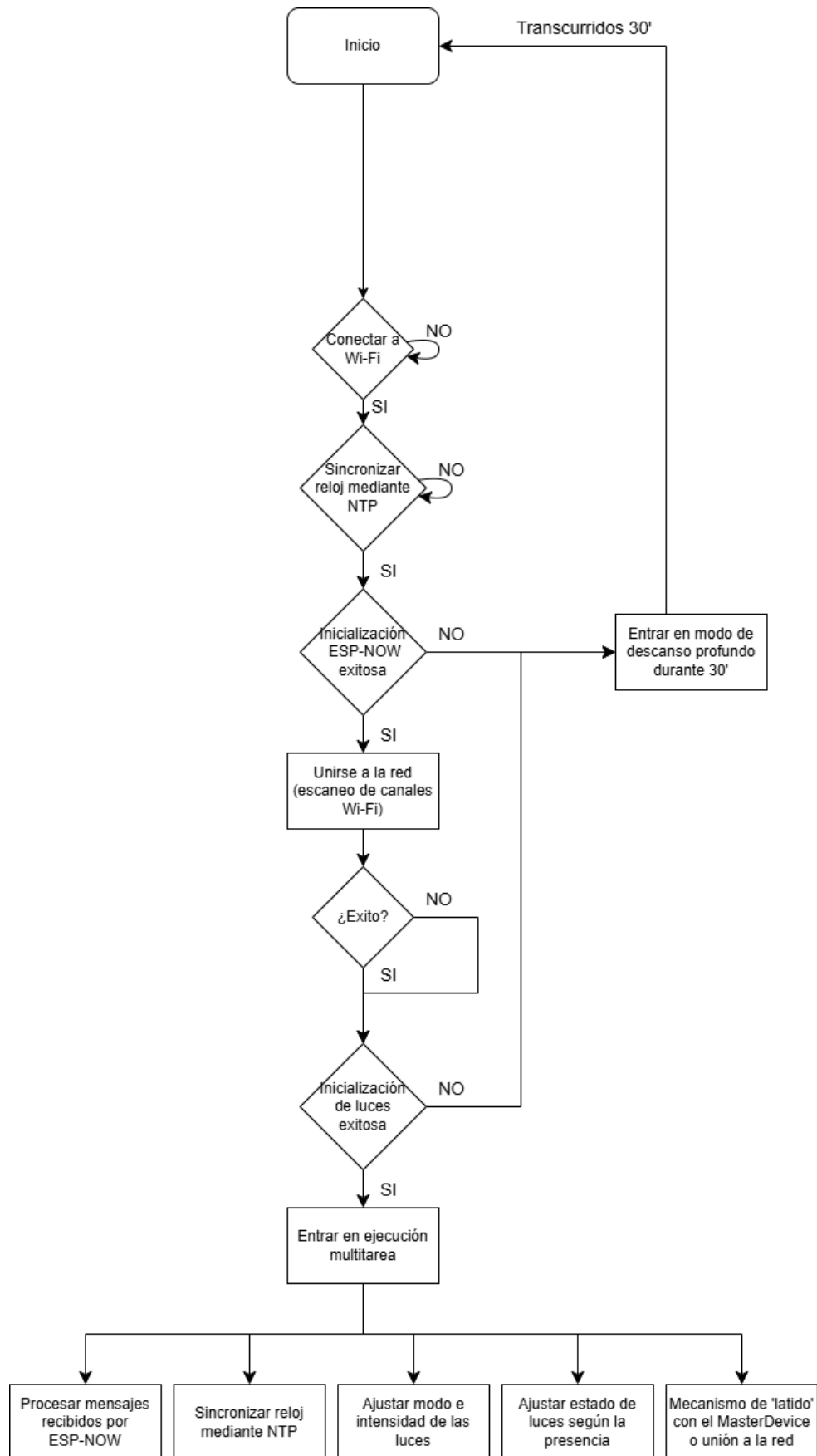


Figura 21. Diagrama de Flujo RoomNode

4.4.4.3 MasterDevice

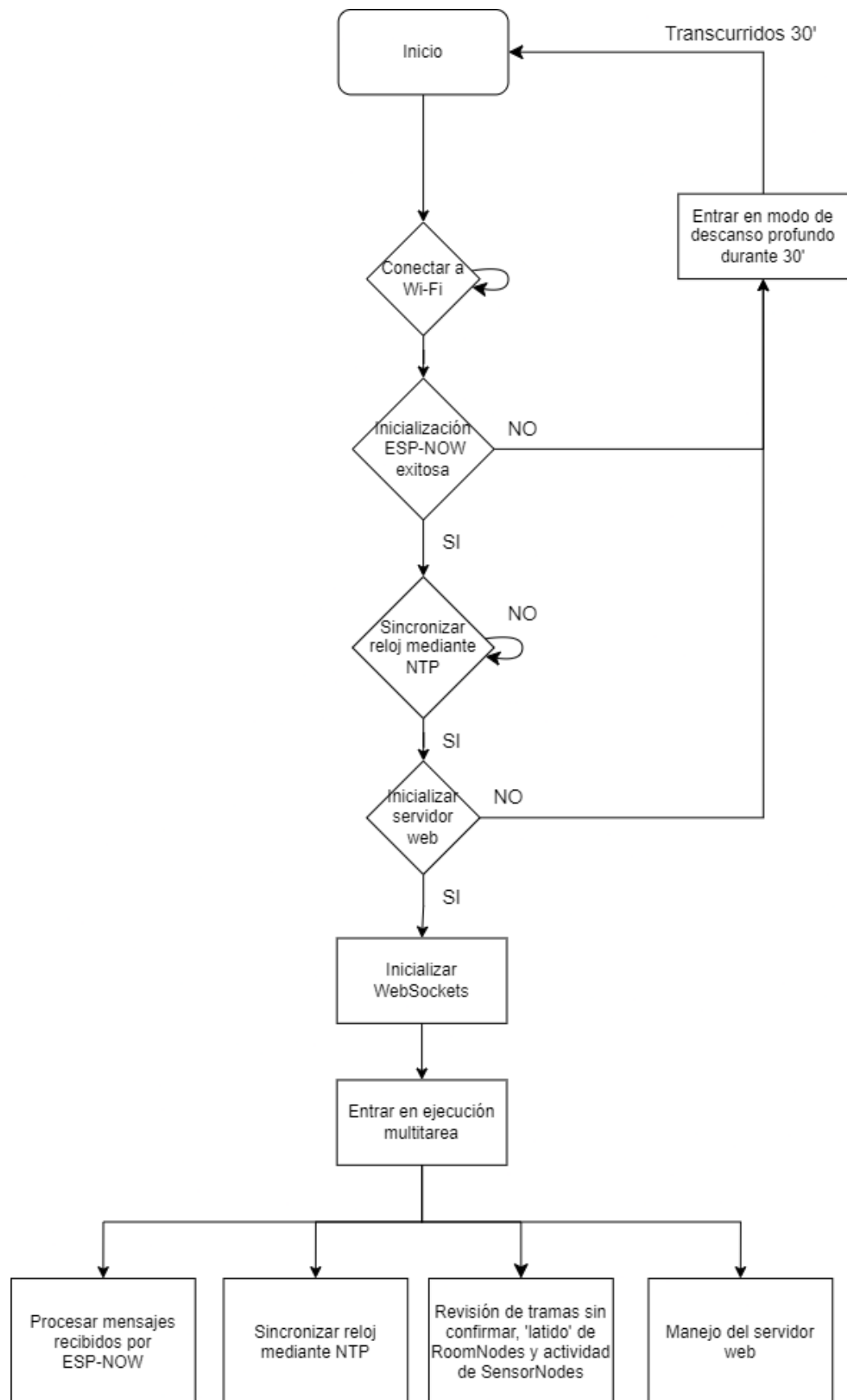


Figura 22. Diagrama de Flujo MasterDevice

4.4.5 Funcionalidades Implementadas

El sistema de automatización desarrollado integra una serie de funcionalidades diseñadas para optimizar la eficiencia energética, el monitoreo ambiental y el control domótico. A continuación, se describen las principales características implementadas:

4.4.5.1 Sincronización Temporal Mediante NTP

La sincronización precisa del tiempo es un aspecto crucial en sistemas de automatización del hogar. Los microcontroladores ESP32 cuentan con un reloj interno que, aunque adecuado para tareas básicas, tiende a desviarse con el tiempo debido a variaciones en la frecuencia de su oscilador. Estas desviaciones pueden afectar negativamente a la ejecución de eventos programados, como los cambios en la iluminación del RoomNode. Para evitar este problema, se utiliza el protocolo Network Time Protocol (NTP) en el MasterDevice y RoomNode, que permite sincronizar los relojes del sistema con servidores de tiempo de alta precisión disponibles en Internet.

Funcionamiento de la Sincronización NTP

Para poder sincronizar el reloj mediante un servidor NTP es necesario que el nodo se encuentre conectado a internet. Una vez conectado, el nodo envía una solicitud a un servidor NTP (e.g., pool.ntp.org) para obtener la hora UTC. A lo que el servidor NTP responde con un paquete que contiene la hora exacta. Después de recibir la respuesta, el nodo ajusta su reloj interno al tiempo proporcionado, asegurando precisión a nivel de segundos.

Particularidades en el RoomNode

El RoomNode implementa medidas específicas para garantizar que el **proceso de sincronización NTP no interfiera con** las capacidades de comunicación **ESP-NOW**, evitando posibles fallos de sincronización o pérdida de datos.

De esta forma el RoomNode mantiene en Wi-Fi desconectado la mayoría del tiempo y solo se reconecta cuando va a sincronizar la hora con el servidor NTP.

Gestión de Conflictos de Canales

Cuando el RoomNode se conecta al Wi-Fi para sincronizarse con el servidor NTP, el router puede asignarle un **canal Wi-Fi diferente al del MasterDevice**. Si durante esta operación el RoomNode intentara enviar mensajes ESP-NOW en el canal incorrecto (ya que el MasterDevice está registrado en un canal diferente), dichos mensajes fallarían.

Uso de Mutex para Bloqueo Temporal

Para evitar estos problemas, el RoomNode emplea un **mutex** (Véase Mecanismos para Tareas Concurrentes) que bloquea temporalmente el acceso a las funciones de radio mientras está conectado al Wi-Fi. Este bloqueo garantiza que no se realicen intentos de comunicación ESP-

NOW hasta que se complete el proceso de sincronización y el nodo regrese al canal correcto. Una vez finalizada la sincronización NTP, el RoomNode libera el mutex, permitiendo reanudar las operaciones normales de ESP-NOW en el canal correspondiente al MasterDevice.

Este diseño asegura que la sincronización del tiempo no comprometa las funcionalidades de comunicación, manteniendo tanto la precisión temporal como la integridad del sistema.

4.4.5.2 Control Domótico de Habitaciones

En cada habitación, un RoomNode se encarga de regular tanto la iluminación como el aire acondicionado, basándose en la detección de presencia, la medición de la luz ambiente y la verificación continua del estado real de las luces. A continuación, se describen los aspectos más relevantes de este control domótico.

Comandos Disponibles para las Luces

En el código se han definido distintos comandos para controlar la iluminación:

- **ON / OFF:** Emplean la misma señal RF y se distinguen por la lógica de verificación con el sensor de luz. ON se considera exitoso si el nivel de lux aumenta, OFF si disminuye.
- **MORE_LIGHT / LESS_LIGHT:** Permiten ajustar gradualmente la intensidad lumínica, incrementándola o reduciéndola hasta alcanzar valores en torno a unos umbrales predefinidos.
- **BLUE / YELLOW:** Cambian la tonalidad de la luz entre modos “fríos” o “cálidos”. Como no se mide con certeza el cambio en la iluminación (pues cambia el color, no la intensidad), el RoomNode asume directamente el éxito de la orden.

Mecanismo de Verificación en el Control de Luces

La iluminación se controla mediante señales RF en la banda de 433 MHz, enviadas al receptor integrado en las luces. Esta comunicación es unidireccional, sin confirmación directa por parte de las bombillas. Para solucionar esta falta de realimentación, el sistema implementa dos estrategias:

- **Lectura de luz ambiente:** El RoomNode dispone de un sensor de luz TSL2591 para verificar si el comando enviado (encender, apagar, aumentar brillo, etc.) ha surtido efecto. Tras emitir la señal RF, el sistema mide la variación en los valores de iluminación (lux). Según la magnitud de ese cambio, concluye si el comando fue exitoso (CommandResult::POSITIVE), fallido (NEGATIVE) o indeterminado (UNCLEAR).
- **Reintento y reversión del comando:** Dado que encender y apagar las luces utilizan la misma señal RF (ON/OFF), es crítico confirmar si realmente se han encendido o apagado. Si la variación en la iluminación no corresponde con lo esperado (por ejemplo, se envía “encender” y el nivel de lux disminuye en lugar de aumentar), el resultado se clasifica como NEGATIVE y el sistema revierte su estado interno, indicando que las luces siguen

en el estado anterior. Además, el RoomNode reenvía la orden para intentar corregir la discrepancia.

Debido a que este proyecto busca reutilizar dispositivos de iluminación RF ya existentes sin capacidad de retroalimentación nativa, resultó imprescindible diseñar un mecanismo de verificación que confirmase el correcto encendido, apagado o ajuste de la iluminación, a pesar de no ser la solución ideal.

No Utilización del Mando Remoto

Es necesario prescindir del uso de este mando remoto de las luces ya que este generaría conflictos con la automatización implementada y provocaría que el sistema perdiera la noción real del estado de las luces, lo que derivaría en inconsistencias de control

Detección de Presencia y Control Automático

Cada RoomNode integra un sensor de presencia radar (LD2410C) que emite un estado digital (alto o bajo) según detecte o no la presencia de personas en la habitación. Este sensor se configura bajo interrupción por cambio de valor, cuya ISR añade el valor del sensor actualizado a una cola de mensajes (presenceQueue).

Mediante una **tarea concurrente** (presenceTask), el RoomNode lee este estado a través de la cola de mensajes presenceQueue y toma decisiones de encendido o apagado de las luces y del aire acondicionado (AC).

Adicionalmente, el RoomNode tiene una tarea concurrente (LightsControlTask) que se encarga de asegurarse de que el **modo de iluminación** (luz caliente o fría) sea el adecuado y ajustar la **intensidad** de la iluminación dentro de un rango adecuado.

Encendido y apagado automático y manual

El RoomNode aplica en la tarea presenceTask las siguientes normas:

- **Encendido automático:** Cuando se detecta presencia, el RoomNode ordena encender las luces si estas se encuentran apagadas y si detecta que no hay suficiente luz natural. No se enciende el aire acondicionado. En el caso de no encender las luces a causa de suficiente luz natural se revisa periódicamente si la luminosidad disminuye hasta un cierto umbral considerado como oscuro, para encender las luces acordeamente dentro de la tarea lightsControlTask.
- **Apagado automático:** Al no detectarse presencia durante el tiempo configurado, las luces se apagan. Adicionalmente, se envía un comando para apagar el aire acondicionado, en caso de que el usuario lo haya encendido manualmente, recordar que la señal IR incluye el nuevo estado completo del dispositivo, por lo que desconocer el estado actual no generará conflicto.

A pesar de que esta regla se cumple la mayoría de las veces existen excepciones si el usuario decide realizar el **apagado manual** de las luces a través de la interfaz web. Cuando esto ocurre el RoomNode apaga las luces y desactiva la lógica de automatización para no reencenderlas hasta que se vuelvan a encender desde la interfaz web.

Esta lógica ha sido implementada especialmente diseñada para dormitorios, permitiendo al usuario apagar las luces manualmente para las horas de descanso.

Ajuste Automático del Brillo y Modo de Iluminación

El RoomNode aplica también reglas relacionadas con el ajuste del estado de las luces en la tarea LightsControlTask, estas reglas consisten en:

- **Automatización según Hora:** El RoomNode configura modos “caliente” (YELLOW) o “frío” (BLUE) basándose en franjas horarias definidas por el usuario (e.g., luz fría por la mañana, luz cálida al anochecer). Cada vez que se encienden las luces, se verifica si corresponde un modo u otro según la hora.
- **Ajuste Dinámico de Brillo:** Si las luces están encendidas, se evalúa la iluminación ambiente y, en caso de estar por debajo o por encima de umbrales (DARK_THRESHOLD y BRIGHT_THRESHOLD), se emite el comando de MORE_LIGHT o LESS_LIGHT.

4.4.5.3 Monitorización Ambiental

Los SensorNodes están equipados con un sensor de temperatura y humedad SHT31, capaz de proporcionar mediciones precisas del entorno. Estos valores se miden de manera periódica y se envían al MasterDevice por medio de ESP-NOW, donde se almacenan y se muestran en la interfaz web.

4.4.5.4 Interfaz Web Interactiva

Para centralizar la gestión del sistema y ofrecer un control intuitivo para los usuarios, el MasterDevice aloja una interfaz web asíncrona que permite:

- **Visualizar datos ambientales** (temperatura y humedad) de los SensorNodes en tiempo real.
- Visualizar un **historial** de los datos ambientales de cada habitación en forma de gráfico.
- **Configurar el periodo de sueño** de los nodos.
- **Encender o apagar** las luces en las distintas habitaciones.
- Modificar las **franjas horarias de luz “cálida” y “fría”**.

Para conectarse a la interfaz web, es necesario que el usuario conozca la dirección IP del MasterDevice, que puede fijarse mediante la configuración del router.

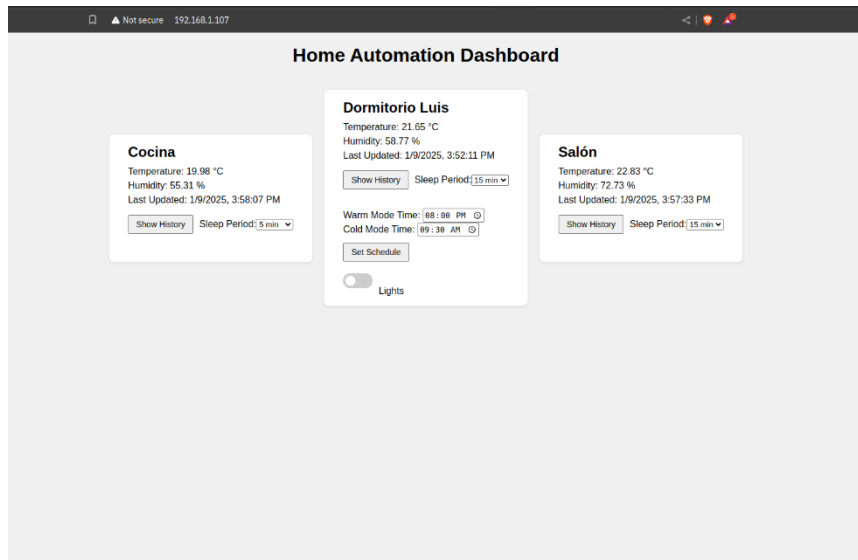


Figura 23. Captura de la Interfaz Web

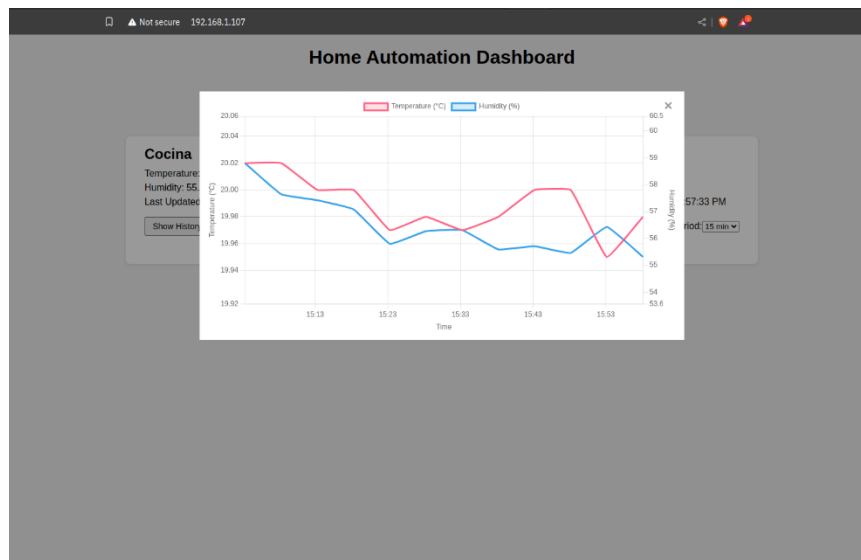


Figura 24. Captura Interfaz Web: Historial de Temperatura y Humedad.

El servidor web, implementado con la librería ESPAsyncWebServer, opera por defecto sobre cualquiera de los dos núcleos del ESP32-S3. Esto permite que las tareas de manejo de eventos del servidor se distribuyan dinámicamente según la carga del sistema. Esta configuración garantiza una respuesta rápida incluso en escenarios de alta carga. Para más detalles sobre el diseño y funcionamiento de la interfaz, véase la Sección 4.5 (Interfaz Web).

4.4.5.5 Gestión Energética Avanzada

Los SensorNodes han sido diseñados con una estrategia de gestión energética que maximiza la duración de sus baterías, proporcionando una autonomía de hasta 452 días (Anexo II). Para ello, se han implementado dos medidas principales:

- **Modo de Bajo Consumo (*Deep Sleep*):** Los nodos permanecen en modo de reposo la mayor parte del tiempo, despertándose únicamente para realizar mediciones ambientales y transmitir los datos al MasterDevice. Este enfoque minimiza el consumo energético, dado que el microcontrolador entra en un estado de bajo consumo mientras no está activo.
- **Optimización del Uso de Wi-Fi:** Para reducir aún más el tiempo de actividad, los nodos emplean ESP-NOW para la transmisión de datos. Este protocolo permite una comunicación rápida y eficiente, reduciendo al mínimo el tiempo activo del módulo Wi-Fi.

Personalización del Periodo de Despertar

A través de la interfaz web, los usuarios pueden configurar el intervalo de tiempo que un SensorNode permanece en modo *Deep Sleep* antes de despertarse. Dado que el SensorNode podría estar en *Deep Sleep* en el momento en que el MasterDevice recibe la nueva configuración, el sistema está diseñado para enviar esta información al nodo la próxima vez que este transmita datos:

- **Cambio en el Periodo de Despertar:** El usuario ajusta el periodo de *Deep Sleep* en la interfaz web, y el MasterDevice recibe y almacena esta configuración.
- **Comunicación al SensorNode:** Cuando el SensorNode envía su próxima trama TEMP_HUMID, el MasterDevice le envía la nueva configuración de tiempo de *Deep Sleep* como respuesta.
- **Actualización del Periodo:** El SensorNode interpreta este mensaje como una orden para actualizar su configuración interna y lo confirma al MasterDevice mediante un ACK.

Este enfoque asegura que los cambios en la configuración de los nodos sean aplicados de manera eficiente, sin interrumpir la rutina de bajo consumo que garantiza una mayor autonomía de las baterías.

Uso de la Memoria RTC

Los SensorNodes emplean la memoria RTC para conservar información crítica entre ciclos de *Deep Sleep*. Esta memoria permite almacenar datos que deben mantenerse disponibles tras el reinicio del microcontrolador al despertar del modo de bajo consumo. En este proyecto, la memoria RTC se utiliza para dos propósitos principales:

- **Almacenamiento del Periodo de *Deep Sleep*:** El tiempo de reposo configurado por el usuario a través de la interfaz web se guarda en la memoria RTC ('sleep_period_ms'). Esto permite que el SensorNode utilice el valor actualizado tras cada ciclo, incluso después de entrar en *Deep Sleep*, sin necesidad de reconfigurarlo.
- **Control del Primer Ciclo de Ejecución:** Un flag ('first_cycle') almacenado en la memoria RTC permite al SensorNode determinar si se encuentra en su primer ciclo de

ejecución tras un reinicio. Este indicador es esencial para decidir el comportamiento del nodo:

- Si es el primer ciclo, el SensorNode debe enviar un mensaje JOIN_SENSOR al MasterDevice para registrarse en el sistema.
- En los ciclos siguientes, puede proceder directamente a medir y transmitir datos ambientales (TEMP_HUMID) sin realizar el proceso de unión.
- **Monitorización del Canal Wi-Fi del MasterDevice:** También se guarda en la memoria RTC el canal Wi-Fi en el que se ha establecido conexión con el MasterDevice (channel). Esto permite comunicarse con el MasterDevice en ciclos posteriores sin necesidad de realizar un escaneo de canales cada vez.

4.4.6 Concurrencia y Gestión de Tareas con FreeRTOS

La plataforma ESP32 ofrece la posibilidad de ejecutar un sistema operativo en tiempo real (RTOS) basado en FreeRTOS. Con este enfoque, el MasterDevice y RoomNode pueden ejecutar distintas funciones de forma concurrente.

Además, el uso de FreeRTOS facilita el diseño modular de la aplicación al aislar cada responsabilidad en una tarea y especialmente la escalabilidad del proyecto, permitiendo añadir fácilmente tramas encargadas de otras funcionalidades.

4.4.6.1 Aspectos Generales de FreeRTOS en ESP32

Las tareas en FreeRTOS[25] son funciones creadas por el usuario que se ejecutan de manera **concurrente**. Cada tarea tiene una prioridad asignada, y el sistema siempre intenta ejecutar la tarea de mayor prioridad que esté "lista para ejecutarse".

FreeRTOS permite simular una ejecución “paralela” de tareas[26], pero esto puede llegar a ser cierto únicamente cuando dos tareas se ejecutan en distintos núcleos de la CPU. La mayoría de las veces, esta concurrencia es una ilusión conseguida mediante una **planificación apropiativa** (“preemptive”). Esto significa que cuando una tarea de mayor prioridad está “lista para la ejecución”, interrumpirá la ejecución de una tarea de menor prioridad para ejecutarse. Si varias tareas tienen la misma prioridad, FreeRTOS aplica un esquema de "time-slicing" para compartir el tiempo de CPU entre ellas.

Cuando el sistema cambia entre tareas, guarda el estado completo de la tarea actual (como registros del procesador y puntero de pila) y restaura el estado de la tarea siguiente. Esto asegura que cada tarea pueda reanudar su ejecución como si no hubiera sido interrumpida.

Este enfoque introduce un dilema en la implementación de un conjunto de tareas concurrentes, ya que el tiempo de CPU es limitado y pueden existir tareas con "deadlines" críticos. En estos casos, es necesario un análisis de viabilidad del sistema para asegurar que las prioridades asignadas

permiten cumplir esos "deadlines". En este proyecto, debido a que el tiempo de ejecución disponible es suficiente para repartir entre las tareas creadas y no existen tareas con "deadlines" críticos, no ha sido necesario realizar dicho análisis.

Mecanismos para Tareas Concurrentes

FreeRTOS proporciona mecanismos como colas de mensajes, semáforos, mutexes y eventos para permitir el acceso seguro a memoria y periféricos, así como también la comunicación y sincronización entre tareas. En este proyecto se han empleado mutex y colas de mensajes:

- **Colas de Mensajes:** Las colas permiten que las tareas se comuniquen entre sí o con las interrupciones de forma segura. Cada mensaje se almacena en la cola y se entrega a la tarea correspondiente según el orden de llegada (FIFO). Esto facilita la transferencia de información entre tareas sin que estas interactúen directamente. En este proyecto, se utilizan para enviar datos como mensajes recibidos por ESP-NOW o estados de sensores en sus respectivas ISRs.
- **Mutex:** Los mutex son utilizados para proteger el acceso a recursos compartidos, como periféricos o variables globales. Solo una tarea puede "bloquear" un mutex a la vez, lo que garantiza que no haya interferencia entre tareas concurrentes. En este proyecto, los mutex se utilizan para garantizar que las tareas no accedan simultáneamente a la memoria o recursos críticos como los módulos de radio o transmisor RF.
- **Semáforo Binario:** Los semáforos binarios permiten la sincronización entre tareas o entre interrupciones y tareas, asegurando que una tarea espere hasta que ocurra un evento específico.

4.4.6.2 Concurrencia en el MasterDevice

El MasterDevice aprovecha los dos núcleos de la CPU del ESP32 para garantizar un funcionamiento eficiente y fluido. El núcleo 1 se reserva para todas las tareas relacionadas con la lógica de la aplicación, mientras que el núcleo 0 se utiliza principalmente para las tareas internas del microcontrolador, como la gestión del stack WiFi y las capacidades de radio.

Por defecto, el servidor web asíncrono (ESPAsyncWebServer) puedan ejecutarse en cualquier núcleo según lo decida FreeRTOS. Esto proporciona flexibilidad y equilibrio en la carga del sistema.

A continuación, se detalla la gestión de las tareas concurrentes y los mecanismos de sincronización empleados:

Tarea	Núcleo	Prioridad	Activación	Función Principal
espNowTask	1	2	Por cola de mensajes	Espera y procesa tramas ESP-NOW recibidas en la cola de mensajes espNowQueue.

NTPSyncTask	1	1	Periódica (300seg)	Sincroniza periódicamente el reloj del MasterDevice con servidores NTP.
updateCheck Task	1	1	Periódica (1seg)	Verifica y reenvía actualizaciones pendientes que no han sido confirmadas (e.g. cambios en el periodo de <i>Deep Sleep</i> o nuevos horarios de iluminación). También comprueba <i>heartbeats</i> y el estado de los SensorNodes, para verificar la continuidad de actividad en los nodos.

Tabla 8. Tareas Concurrentes del MasterDevice

Para coordinar estas tareas y evitar conflictos en el acceso a recursos compartidos, el MasterDevice emplea:

- **Colas de Mensajes (espnowQueue):** Facilitan la comunicación entre la ISR de ESP-NOW y la tarea espNowTask. Las tramas recibidas se encapsulan en estructuras y se envían a la cola mediante xQueueSendFromISR. Esto permite que la ISR sea lo más breve posible y que el procesamiento completo ocurra en el contexto de la tarea.
- **Mutex (sensorMutex, controlMutex):** Protegen los datos compartidos en el DataManager: sensorMutex protege los datos relacionados con el SensorNode, mientras que controlMutex se encarga de los datos del RoomNode.

4.4.6.3 Concurrencia en el RoomNode

El RoomNode utiliza también las capacidades multitarea de FreeRTOS, pero, en este caso se ejecutan todas las tareas en un mismo nodo. A continuación, se presenta la descripción de las tareas concurrentes implementadas:

Tarea	Prioridad	Activación	Función Principal
espNowTask	3	Por cola de mensajes (espNowQueue)	Procesa tramas ESP-NOW recibidas desde espNowQueue, gestionando la comunicación con el MasterDevice.
presenceTask	3	Por cola de mensajes (presenceQueue)	Gestiona eventos de presencia detectados y enviados a presenceQueue por el radar LD2410. Toma decisiones sobre encendido y apagado de las luces y aire acondicionado.
lightControlTask	2	Periódica	Ajusta automáticamente el brillo y el modo de iluminación según los niveles de luz ambiente y

		(1 seg)	la hora del día. Si las luces están apagadas, enciende las luces si detecta presencia y que no hay suficiente luz ambiental.
NTPSyncTask	1	Periódica (300 seg)	Sincroniza periódicamente el reloj del RoomNode con servidores NTP.
heartbeatTask	2	Periódica (120 seg)	Envía una trama HEARTBEAT al MasterDevice. En caso de perder la conexión con el MasterDevice trata de reconectarse al sistema.
lightsToggleTask		Por cola de mensajes (lightsToggleQueue)	Procesa las peticiones de cambio de estado (ON/OFF) del maestro.

Para evitar conflictos y garantizar la coherencia en el acceso a los recursos compartidos, el RoomNode emplea:

- **Colas de Mensajes (espnowQueue, presenceQueue, lightsToggleQueue):**
 - espnowQueue: Recibe tramas ESP-NOW desde la ISR, que luego son procesadas por espNowTask.
 - presenceQueue: Recibe el estado en el sensor de presencia LD2410C cuando se detecta un cambio en la ISR. Al igual que espnowQueue, permite reducir al mínimo el tiempo de ejecución de la ISR.
 - lightsToggleQueue: Almacena el nuevo estado (ON/OFF) de las luces ordenado por el master.
- **Mutex (transmitterMutex, radioMutex, isOnMutex, lightSensorMutex):**
 - transmitterMutex: Protege el acceso a el transmisor RF.
 - radioMutex: Asegura que no se envíen tramas ESP-NOW mientras el Wi-Fi se encuentre concetado.
 - isOnMutex: Protege la lectura y escritura del estado de las luces.
 - lightSensorMutex: Asegura que dos tareas no interactúen al mismo tiempo con el sensor de luz TSL2591.
- **Semáforo Binario (ackSemaphore)**
 - Coordina la recepción de mensajes de confirmación (ACK) enviados por el MasterDevice. Permite que el RoomNode espere de forma bloqueante por el

ACK tras enviar un mensaje, asegurando que solo se proceda cuando se recibe la confirmación correspondiente.

4.5 Interfaz Web

4.5.1 Justificación de la Arquitectura Elegida

A la hora de decidir cómo alojar la interfaz web y centralizar la gestión de los datos, se consideraron tres opciones principales: la integración con Home Assistant, la implementación de un servidor local en la propia ESP32 (MasterDevice) y, por último, el uso de plataformas en la nube, como AWS. A continuación, se exponen las principales razones que han motivado la elección final del alojamiento en la propia ESP32:

4.5.1.1 Home Assistant

Home Assistant[27] es una plataforma de automatización del hogar de código abierto con amplia adopción en la comunidad DIY. Ofrece una interfaz web intuitiva, integración con una gran variedad de fabricantes y protocolos, y un gran ecosistema de complementos.

- **Ventajas:**
 - La sencillez para **unificar dispositivos de distintos fabricantes** en una misma interfaz.
 - Posibilidad de automatizar múltiples aspectos del hogar sin necesidad de programar “desde cero”, gracias a su **comunidad** y a la abundancia de **integraciones**.
 - Permite un **control avanzado** y la capacidad de aprovechar un sistema de notificaciones, scripts e históricos más extensos.
- **Inconvenientes:**
 - **Coste de hardware adicional:** Se requiere una Raspberry Pi o una máquina virtual dedicada para alojar Home Assistant, lo que encarece el sistema.
 - **Dependencia del Wi-Fi:** De no usarse un “MasterDevice” con ESP-NOW, los SensorNodes tendrían que comunicarse vía Wi-Fi para enviar las tramas a Home Assistant, lo cual incrementaría significativamente el consumo y acortaría la autonomía de las baterías.
 - **Valor añadido limitado:** El sistema propuesto no contempla una gran expansión con múltiples dispositivos, por lo que la versatilidad extra que aporta Home Assistant no compensa su coste en términos de hardware, mantenimiento y consumo.

4.5.1.2 Implementación en la propia ESP32 (MasterDevice)

Consiste en utilizar un dispositivo ESP32-S3 como servidor local, encargado de centralizar la recepción de datos (mediante ESP-NOW), así como de exponer una interfaz web asíncrona para su configuración y monitoreo.

- **Ventajas:**
 - **Eficiencia energética:** Al mantener la comunicación entre nodos mediante ESP-NOW, se minimiza el tiempo activo de los SensorNodes y se prolonga la autonomía de sus baterías.
 - **Bajo coste y simplicidad:** No se requiere hardware adicional, como una Raspberry Pi.
 - **Privacidad y control total:** Al no depender de la nube ni de plataformas externas, el control y almacenamiento de datos permanecen en la red local.
 - **Escalabilidad razonable:** Para un número reducido o medio de dispositivos, el ESP32 puede manejar sin problemas la interfaz web y la comunicación interna.
 - **Personalización:** Permite diseñar la interfaz web al gusto y necesidades del usuario.
- **Inconvenientes:**
 - **Limitaciones de hardware** en caso de alta concurrencia o de requerir potentes gráficos en la interfaz.
 - Requiere un mayor nivel de **programación y conocimientos de servidor web** y ESP32, en contraste con la simplicidad de instalación que proporciona Home Assistant.

4.5.1.3 Servicios en la nube (AWS o similares)

Apoyarse en servicios cloud, como AWS, para almacenar y procesar datos, además de proveer la interfaz de administración.

- **Ventajas:**
 - **Escalabilidad casi ilimitada**, útil si se prevé un crecimiento elevado del sistema.
 - **Acceso remoto** fácil y sofisticadas herramientas de **análisis y visualización**.
- **Inconvenientes en el proyecto:**
 - **Coste periódico:** A pesar de ofrecer AWS IoT precios adaptativos[28], puede generar un coste adicional a largo plazo, especialmente si se escala el sistema.
 - **Privacidad:** Los datos se almacenan en servidores externos, lo que conlleva mayor complejidad en términos de protección de datos y dependencia de Internet.

- **Sobredimensionamiento:** El proyecto no requiere análisis masivo de datos ni escalado global, por lo que la nube no ofrece un beneficio significativo que compense sus inconvenientes.

4.5.1.4 Conclusión

Después de valorar las alternativas, se ha optado por alojar la interfaz web y la lógica de control en la propia **ESP32-S3 (MasterDevice)**. Esta solución equilibra de forma óptima los costes, el consumo energético y la funcionalidad. Por un lado, se mantiene la eficiencia y sencillez de la comunicación con los SensorNodes mediante ESP-NOW, y por otro, se dispone de una interfaz web local que no depende de servicios externos ni genera gastos adicionales. Además, se protege la privacidad de los datos al residir exclusivamente en la red local, evitándose la transmisión continua a plataformas en la nube.

4.5.2 Fundamentos de un Servidor Web en ESP32

Los **servidores web** tradicionales están diseñados para ejecutarse en sistemas con **recursos sustanciales**, como ordenadores con sistemas operativos completos. Estos servidores pueden gestionar grandes volúmenes de tráfico concurrente, alojar páginas web complejas e incluso ejecutar aplicaciones dinámicas desarrolladas en tecnologías como Angular o Node.js.

Cuando se traslada esta funcionalidad a un microcontrolador ESP32, surgen desafíos importantes debido a sus **limitaciones de hardware**. La memoria y la capacidad de procesamiento disponibles en la ESP32 son considerablemente inferiores a las de un ordenador convencional, lo que hace necesario optimizar tanto la gestión de recursos como la lógica de funcionamiento del servidor web.

A pesar de estas restricciones, el ESP32 es lo **suficientemente potente** como para generar y servir **contenido web básico** de manera eficiente, siempre que se empleen herramientas adecuadas. Una de las más destacadas es la librería **ESPAsyncWebServer**, diseñada específicamente para entornos con capacidades limitadas, como los microcontroladores.

4.5.2.1 AsyncESP32WebServer: Concurrencia y Eficiencia

La librería **ESPAsyncWebServer** introduce un modelo de programación **asíncrona**, ideal para manejar la concurrencia y optimizar los recursos en un servidor web basado en ESP32. Este enfoque ofrece varias ventajas clave:

- **Manejo de Múltiples Conexiones Concurrentes:** En lugar de procesar cada petición de forma secuencial y bloquear recursos hasta concluir la respuesta, el servidor asíncrono puede atender múltiples solicitudes en paralelo.
- **Evitación de Bloqueos:** En servidores sin capacidades asíncronas (e.g., basados en el modelo Arduino tradicional), es común que la ejecución principal se detenga cuando se

gestionan peticiones de gran tamaño. Con AsyncESP32WebServer, cada operación de lectura y escritura de la red se divide en fragmentos gestionados por el loop principal de la librería, permitiendo que el microcontrolador continúe con otras tareas en segundo plano.

- **Gestión Simplificada de Solicitudes HTTP:** Esta librería proporciona métodos para definir rutas y manejar diferentes tipos de peticiones (GET, POST, etc.), facilitando la implementación de la lógica del servidor.
- **Compatibilidad con WebSockets:** Además de las tradicionales solicitudes HTTP, AsyncESP32WebServer ofrece soporte nativo para conexiones WebSocket. Esto habilita la comunicación bidireccional entre la ESP32 y el navegador del usuario, fundamental para notificar actualizaciones en tiempo real sin recargar la página.

En conclusión, la adopción de un enfoque asíncrono en el servidor web de la ESP32 se alinea perfectamente con las limitaciones de hardware del microcontrolador y las demandas específicas de aplicaciones domóticas ligeras. Este modelo permite aprovechar al máximo los recursos disponibles sin comprometer la capacidad de respuesta ni la estabilidad del sistema, garantizando una experiencia fluida para los usuarios y un consumo eficiente de recursos.

4.5.3 Interacción Cliente-Servidor: HTTP y WebSockets

La comunicación entre cliente y servidor en aplicaciones web puede implementarse mediante diversas técnicas, cada una con ventajas e inconvenientes según los requisitos del sistema. En este proyecto se han comparado dos enfoques principales: el método HTTP tradicional y el método implementado, WebSockets.

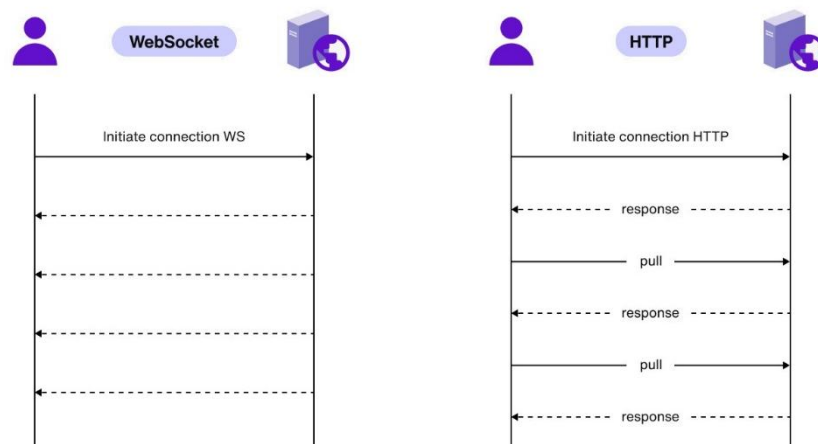


Figura 25. Diagrama Comunicación HTTP vs WebSocket

4.5.3.1 HTTP Tradicional

El protocolo HTTP[29], [30] opera en un modelo de **solicitud-respuesta**, donde el cliente envía peticiones individuales al servidor, que responde con los datos solicitados. Este enfoque resulta

adecuado para aplicaciones con actualizaciones poco frecuentes o no críticas. Sin embargo, en sistemas que requieren actualizaciones constantes, como este proyecto, el método tradicional presenta limitaciones significativas. Para obtener actualizaciones en tiempo real mediante HTTP tradicional, sería necesario recurrir al envío de solicitudes periódicas (*polling*), lo que incrementaría la latencia, la carga en la red y el consumo de recursos del servidor.

Además, el carácter *stateless* de HTTP, donde cada petición es independiente, dificulta mantener un flujo continuo de información. Aunque en el proyecto actual este aspecto no supone un problema inmediato, podría resultar ineficiente si se ampliara la interfaz web con múltiples páginas o funcionalidades más complejas.

4.5.3.2 WebSockets

Los WebSockets, por otro lado, ofrecen una solución más eficiente al permitir **conexiones persistentes y bidireccionales entre cliente y servidor**. Una vez establecida la conexión inicial mediante un *handshake* HTTP, el canal permanece abierto, lo que facilita el intercambio continuo de datos sin necesidad de repetir solicitudes.

Este enfoque permite que tanto el cliente como el servidor envíen datos en tiempo real, optimizando la comunicación para sistemas dinámicos como el planteado en este proyecto. La reducción de latencia y la disminución del tráfico de red hacen de los WebSockets una opción ideal para gestionar lecturas de sensores y eventos de manera inmediata.

4.5.3.3 Razones para la Elección

En este proyecto, la implementación de WebSockets se ha considerado esencial debido a su capacidad para manejar **actualizaciones en tiempo real** de manera eficiente. Uno de los casos de uso más destacados es la notificación instantánea de cambios en el estado de los sensores o dispositivos conectados al sistema, lo cual resulta crucial para garantizar una experiencia de usuario fluida y reactiva.

Otro caso relevante es la posibilidad de integrar en un futuro una **pantalla fija** en el hogar, conectada de manera permanente al servidor. Esta pantalla actuaría como un **panel de control** donde el usuario podría visualizar de forma continua el estado de los sensores y dispositivos. Gracias a los WebSockets, este panel podría recibir actualizaciones en tiempo real sin incrementar la carga del servidor ni del microcontrolador ESP32, asegurando una operación eficiente.

4.5.3.4 Costes y Limitaciones

El principal coste asociado al uso de WebSockets es el **consumo de memoria**, que limita el número de conexiones simultáneas. No obstante, debido al contexto del proyecto, donde se espera un número reducido de usuarios concurrentes, esta limitación no supone un problema significativo.

A pesar de que los WebSockets implican ciertos retos, como un mayor consumo de memoria en el servidor y una mayor complejidad en su implementación, estos han sido gestionados con éxito. La librería AsyncWebServer simplifica la configuración de WebSockets, mientras que se ha optimizado el uso de los recursos del ESP32.

4.5.4 Almacenamiento y Rol de los Archivos Estáticos en LittleFS

Los archivos estáticos, como HTML, CSS, JavaScript y otros recursos como el favicon (icono para el sitio web), son esenciales para la construcción de la interfaz web. Estos elementos se almacenan en el sistema de archivos LittleFS dentro de la memoria **flash** del ESP32. Este sistema, además de beneficiar en el desarrollo del código, permite alojar y servir dichos archivos rápidamente directamente desde el microcontrolador, proporcionando al navegador del usuario el contenido necesario para una experiencia interactiva y visualmente atractiva.

El archivo **HTML** define la estructura básica de la página web, **CSS** estiliza los elementos visuales para garantizar una presentación coherente y profesional y por último **JavaScript** maneja la lógica del lado del cliente, como las interacciones dinámicas y las actualizaciones de datos en tiempo real. Estos recursos estáticos se sirven desde el directorio `data/` del repositorio cargado en LittleFS durante la etapa de compilación.

La elección de LittleFS como sistema de archivos asegura robustez y eficiencia, factores críticos en el contexto de sistemas empujados. Para más detalles sobre LittleFS consultar el Anexo VI.

4.6 Pruebas y Validación

4.6.1 Entorno de Pruebas y Metodología

El sistema está diseñado para soportar múltiples SensorNodes y varios RoomNodes. No obstante, el conjunto de pruebas se ha realizado sobre un entorno de pruebas reducido que incluye:

- Un MasterDevice (ESP32-S3).
- Tres SensorNodes (Lolin D32).
- Un RoomNode (ESP32 Dev Kit C V4).

Esta configuración, representada en la siguiente figura, ha resultado suficiente para validar las funcionalidades clave, ya que se han cubierto todos los aspectos principales: intercambio de datos mediante ESP-NOW, control de dispositivos del hogar y monitoreo mediante la interfaz web.

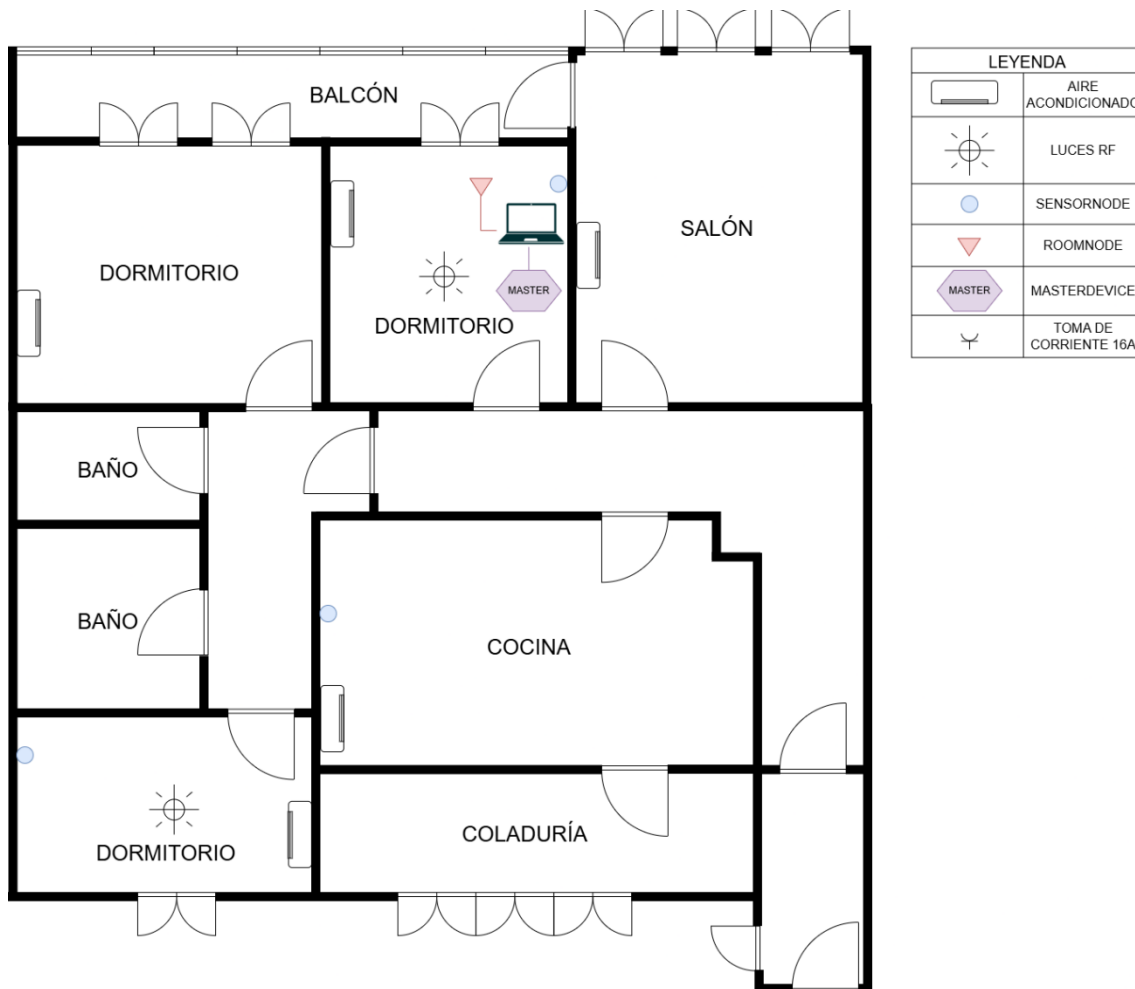


Figura 26. Plano Entorno de Pruebas

El despliegue del entorno de pruebas se ha llevado a cabo en una vivienda real, para ver más detalles sobre el montaje físico consultar Anexo VIII: Montaje Físico Entorno de Pruebas. Se mantuvo el RoomNode y MasterDevice alimentados y conectados al ordenador para la verificación y control de funcionamiento. Se repartieron 3 SensorNodes en distintas habitaciones, uno de ellos se colocó en un dormitorio relativamente alejado, con el fin de comprobar el alcance del protocolo ESP-NOW a través de paredes y pasillos.

Se ha grabado un pequeño video (<https://youtu.be/uPGxSjT0MKE>) con el fin de demostrar la implementación de las funcionalidades descritas.

4.6.1.1 Metodología de Validación

La validación se ha realizado de manera incremental y orientada a la integración, siguiendo este procedimiento general en la gran mayoría de funcionalidades:

- **Desarrollo aislado y pruebas preliminares:** Antes de integrar cada funcionalidad en el proyecto principal, se han elaborado códigos de prueba específicos para módulos de

radiofrecuencia (433 MHz), emisión IR, detección por radar (LD2410C), etc. De esta forma, se confirmó su correcto funcionamiento individual.

- **Integración progresiva:** Una vez verificado el correcto funcionamiento en las pruebas aisladas se integraron en el proyecto principal. Allí se llevaron a cabo pruebas para verificar que no surgieran conflictos con el resto del sistema. Una vez validado el funcionamiento, se fusionaron los cambios en la rama principal (main).
- **Validación en entorno doméstico:** Cada vez que se completaba una funcionalidad (e.g., control de luces, envío de datos al MasterDevice, o sincronización NTP), se actualizaba el software de los distintos nodos del entorno de pruebas correspondiente y se llevaban a cabo pruebas reales. Esta práctica permitió detectar y corregir posibles problemas futuros.

4.6.2 Validación de Funcionalidades Principales

Se realizaron multitud de pruebas durante el desarrollo, pero las más relevantes han sido:

4.6.2.1 Control de Luces RF

Se comprobó que el RoomNode podía **encender y apagar** las luces a 433 MHz **de manera fiable** mediante el transmisor FS1000A, replicando la señal del mando original. Para ello, se enviaron instrucciones ON y OFF desde la interfaz web y se monitorizaron los valores de iluminación captados por el sensor TSL2591, así como el resto de los mensajes en el puerto serie. Se validó que el microcontrolador llevará un **control correcto del estado** de las luces, y que pausará y reanudará su la automatización de acuerdo con lo que el usuario ordenara desde la página web.

4.6.2.2 Automatización Basada en Presencia (LD2410)

En pruebas preliminares se alimentó el sensor mediante el pin VIN del ESP32, pero debido a **limitaciones de amperaje** la detección de presencia no funcionaba correctamente, ya que, según detalla el *datasheet*, el sensor requiere de una alimentación con capacidad de administrar más de 200mA, estándares que el ESP32 no podía cumplir, y menos con el número de módulos conectados al RoomNode. Más tarde se alimentó el sensor mediante una fuente de alimentación externa a 5V y 500mA y la detección de presencia funcionó como indicaba el fabricante, demostrando que los problemas previos eran debidos a una alimentación inadecuada.

Para validar la detección de presencia inmóvil, se realizaron pruebas en las que una persona se mantuvo sentada o en posición estática durante un intervalo prolongado, y en las que se verificó que el sensor cumplía su función de forma excelente.

El LD2410C demostró su capacidad de detectar micromovimientos y a mantener las luces encendidas incluso mientras la persona se tumbaba en la cama o se mantenía sentada en el escritorio durante tiempos prolongados. También se verificó el apagado automático transcurrido

no-one duration (por defecto 5 segundos) al salir de la habitación, que el sistema cumplió tal cómo se había configurado con un error no mayor a 2 segundos.

4.6.2.3 Respuesta ante Reinicios y Cortes de Energía

Con el fin de confirmar la robustez del sistema frente a fallos inesperados, se desconectaron tanto los RoomNodes y SensorNodes como el MasterDevice en distintos momentos. En todos los casos se observó el comportamiento correcto:

Al **desconectar** el **MasterDevice**, los SensorNodes y RoomNodes detectaron que no respondían respuesta por parte del maestro y reanudaron correctamente su funcionamiento gracias a los mensajes JOIN_ROOM y JOIN_SENSOR, escaneando canales Wi-Fi en caso de que el MasterDevice estuviera inactivo o se hubiese reiniciado.

Asimismo, cuando se **desconectaba** un **RoomNode** o **SensorNode** el MasterDevice eliminaba automáticamente de su registro a los nodos inactivos transcurrido un periodo sin recepción de *heartbeats* (RoomNode) o tramas TEMP_HUMID (SensorNode).

4.6.2.4 Alcance de la Comunicación ESP-NOW

Se probó la comunicación ESP-NOW con uno de los SensorNodes ubicado en la **habitación más alejada** de la vivienda, con varias paredes y puertas intermedias. Durante el periodo de pruebas, todas las tramas TEMP_HUMID llegaron al MasterDevice de forma exitosa, evidenciando que el alcance y la calidad de la señal eran suficientes bajo las condiciones de la vivienda.

4.6.2.5 Interfaz Web y Manejo Concurrente

La interfaz web se sometió a pruebas de carga moderada, conectando **simultáneamente** hasta **cuatro dispositivos** para verificar su capacidad de respuesta. No se observaron retardo ni bloqueos significativos mientras se interactuaba con el sistema, y las funcionalidades de encendido/apagado de luces, modificación de periodos de sueño y actualización de datos de sensores siguieron funcionando de forma fluida.

A pesar de esto, es importante mencionar que, si se realizan peticiones de apagado y encendido de las luces muy seguidas entre sí puede haber un retraso significativo hasta que se aplica la última petición enviada. Esto es debido a que el RoomNode ejecuta las peticiones en cola (No empieza la siguiente hasta que acaba la anterior) y a que tiene que esperar un tiempo para verificar el cambio de estado de las luces mediante la variación de luminosidad.

5 Resultados

5.1 Cumplimiento de Objetivos

Tras el desarrollo e implementación del proyecto, se cumplieron con éxito los objetivos planteados:

- Se creó un sistema domótico **multinodo** basado en microcontroladores ESP32, capaz de monitorear variables ambientales (temperatura, humedad) y controlar luces y aire acondicionado de manera automática.
- Se integraron nodos de **bajo consumo** (SensorNodes), lo que prolonga la autonomía cuando son alimentados por baterías.
- Se **gestionó la iluminación y climatización** a través de sensores de presencia y luminosidad, reduciendo así el uso innecesario de recursos.
- El MasterDevice proporciona **una interfaz web local** que permite la interacción en tiempo real con todos los elementos del sistema, así como la configuración de parámetros (periodo de sueño, horarios de luz cálida/fría, etc.), sin dependencia de plataformas en la nube.
- El protocolo ESP-NOW, garantizó **baja latencia** en la comunicación interna y mantuvo el consumo energético en valores competitivos.

Como resultado, se obtuvo un ecosistema domótico inteligente construido sobre dispositivos de hogar tradicionales, donde cada nodo cumple una función independiente y coordinada, satisfaciendo las metas de modularidad, eficiencia y flexibilidad.

5.2 Comparación de Costes

Para evaluar la eficiencia económica del sistema propuesto, se estableció un **escenario específico** que incluye la monitorización de datos ambientales en tres habitaciones mediante SensorNodes, y el control y automatización de las luces en una habitación mediante un RoomNode. Este despliegue requiere un total de tres SensorNodes, un RoomNode y un MasterDevice.

Basado en los costes detallados en el Anexo VII: Cálculo de Costes por Nodo, en el escenario planteado, el coste total estimado para implementar el sistema domótico basado en microcontroladores ESP32 asciende a aproximadamente **56.5€**. Es importante destacar que estos precios se basan en la adquisición de unas pocas unidades de cada componente.

Es complicado realizar una comparación precisa de costes con otras soluciones comerciales, ya que es difícil encontrar dispositivos que compongan exactamente las mismas funcionalidades ofrecidas por el sistema propuesto. A pesar de esto, se estima que soluciones comerciales de gama

media, como las ofrecidas por Philips Hue u otros fabricantes reconocidos, requieren una **inversión inicial significativamente mayor**. En estos casos, tan solo el ‘hub’ de control asciende hasta los 50€, sin contar costes de sensores ni actuadores.

Por otro lado, los sistemas cableados tradicionales, como **KNX**, demandan una inversión **aún más elevada**, con costes que pueden ascender a varios cientos de euros por cada elemento de control. Además de los componentes específicos, estos sistemas requieren una instalación profesional, lo que incrementa notablemente el coste global y la complejidad de la implementación.

5.3 Limitaciones del Sistema

A pesar de los logros alcanzados, el sistema desarrollado presenta ciertas limitaciones que deben ser consideradas para futuras mejoras y aplicaciones más extensas:

5.3.1 Seguridad en la Comunicación

A pesar de los logros alcanzados, el sistema desarrollado presenta ciertas limitaciones que deben ser consideradas para futuras mejoras y aplicaciones más extensas. Una de las principales restricciones radica en la seguridad de la comunicación. El protocolo **ESP-NOW**, empleado para la interacción entre nodos, **no incorpora cifrado nativo**, lo que podría exponer el sistema a riesgos de interceptación o acceso no autorizado.

5.3.2 Accesibilidad

Otra limitación significativa es la accesibilidad del sistema. Al estar alojado en una red local, el acceso a la interfaz web del MasterDevice requiere estar conectado a la misma **red Wi-Fi** y conocer la **dirección IP** del MasterDevice, lo que puede resultar incómodo en comparación con aplicaciones móviles comerciales que ofrecen interfaces más intuitivas y accesibles desde cualquier ubicación.

No obstante, existe la posibilidad de habilitar el acceso remoto a la interfaz web, abriendo un puerto en el router y configurando un servicio de DNS dinámico (e.g., mediante plataformas como No-IP[31]) para gestionar la dirección IP cambiante del proveedor de Internet. Aunque, debido a que se expondría la interfaz a Internet, sería necesario como mínimo implementar un sistema de autenticación y considerar el cifrado del tráfico HTTP.

5.3.3 Usabilidad

Existen limitaciones en la interacción con las luces, ya que la utilización del **mando remoto original** genera **conflictos** con la automatización implementada y provoca que el sistema pierda la noción real del estado de las bombillas. Esta situación deriva en inconsistencias de control, por lo que, en la práctica, se deberá prescindir del mando remoto y limitar la interacción de las luces

la interfaz web. En un futuro se pretende incorporar un receptor RF en el RoomNode que registre los comandos enviados por dicho mando y actualice correctamente el estado interno de la luz.

5.3.4 Portabilidad

La portabilidad del sistema también representa un desafío. Si se quisiera trasladar el sistema a otro hogar, los SensorNodes y el MasterDevice podrían operar sin inconvenientes, pero los RoomNodes dependen de luces que funcionen mediante radiofrecuencia **433 MHz ASK** con **parámetros específicos**. Aunque el control del aire acondicionado es fácilmente adaptable gracias a la compatibilidad de la librería utilizada con una variedad de fabricantes, la necesidad de utilizar luces con características concretas dificulta la replicación del sistema en diferentes entornos sin modificaciones adicionales en el hardware o el software.

Además, las automatizaciones implementadas pueden no satisfacer **las preferencias de todos los usuarios**. Las reglas de encendido y apagado basadas en sensores de presencia y luminosidad están diseñadas para optimizar el consumo energético, pero es posible que algunos usuarios deseen configuraciones más personalizadas que no están contempladas en el sistema actual.

5.3.5 Escalabilidad

En términos de escalabilidad, aunque los microcontroladores ESP32 ofrecen grandes capacidades y, mediante una programación correcta, podrían ofrecer funciones similares a dispositivos de fabricantes domóticos más establecidos, la **ampliación** del sistema con más **nodos** o el aumento de **usuarios** conectados a la interfaz web introduce una carga adicional no contemplada y que afectaría la funcionalidad del sistema.

5.3.6 Rendimiento y Fiabilidad

Finalmente, el rendimiento y la fiabilidad del sistema dependen en gran medida de la calidad de los **componentes** utilizados y de la precisión en la configuración e implementación. Problemas como interferencias en la señal RF, limitaciones de rango en la comunicación ESP-NOW, o inconsistencias en la detección de presencia durante el tiempo podrían afectar la estabilidad y eficacia del sistema en entornos reales. Abordar estos aspectos requeriría una **optimización continua** y posibles mejoras en la selección y calibración de los sensores y módulos empleados, asegurando así una operación más robusta y fiable en diversas condiciones de uso.

6 Conclusiones

6.1 Contexto

La necesidad de sostenibilidad y eficiencia energética en la domótica motivó este trabajo, orientado a democratizar el acceso a la automatización del hogar. Desde el inicio se planteó el uso

de microcontroladores ESP32 para elaborar un sistema modular y escalable que abarcara tanto la monitorización ambiental como el control de dispositivos domésticos (luces y aires acondicionados), reutilizando equipamiento existente gracias a la incorporación de módulos infrarrojos y de radiofrecuencia.

El problema central radicó en la carencia de soluciones económicas y flexibles para modernizar dispositivos tradicionales, sin depender de infraestructuras complejas ni presupuestos elevados. Con esa premisa, se establecieron los siguientes objetivos:

- Diseñar una arquitectura **multinodo** (SensorNodes, MasterDevice y RoomNodes).
- **Optimizar la comunicación** mediante ESP-NOW y, puntualmente, Wi-Fi.
- Implementar **algoritmos de automatización** basados en sensores de presencia y luminosidad.
- Garantizar la **escalabilidad** y **validar el sistema** en un entorno real.
- Favorecer la **eficiencia energética** tanto en nodos de batería como en el control de luces y climatización.

6.2 Principales Conclusiones e Implicaciones

Protocolo de Comunicación y Ahorro Energético

El uso de ESP-NOW como protocolo principal de comunicación interna, con respaldo puntual de Wi-Fi para la interfaz web y la sincronización NTP, demostró ser eficaz para reducir el consumo energético en nodos alimentados por batería. Esta elección simplifica la arquitectura al no requerir routers o plataformas externas para el intercambio de datos.

Arquitectura Multinodo y Flexibilidad

La organización del sistema en SensorNodes (monitoreo de temperatura y humedad), RoomNodes (control de iluminación y climatización) y un MasterDevice central permitió una fácil separación de tareas y una escalabilidad escalonada. Esta modularidad facilita el crecimiento del sistema y su adaptación a distintos entornos residenciales.

Modernización de Dispositivos Tradicionales

La integración de módulos IR y RF sirvió para controlar luces unidades de aire acondicionado de marca Panasonic (u otros) sin sustituir el equipamiento original. Este enfoque ofrece un reaprovechamiento de dispositivos preexistentes, reduciendo la inversión necesaria y disminuyendo la huella ambiental.

Este propósito resultó ser todo un **desafío** ya que fue necesario adaptar el software a unas **características fijadas por hardware**, que no fueron concebidas para su replicación y

automatización. De este modo se tuvieron que implementar **mecanismos de verificación** que aseguraran un sistema robusto.

Automatizaciones y Optimización de Recursos

El sistema permite control automático de iluminación basado en sensores de presencia (LD2410C) y luminosidad (TSL2591), lo que conlleva ahorros de energía al reducir el tiempo de encendido de luces y el uso innecesario de aparatos de climatización.

Aplicaciones Prácticas

Viviendas pequeñas y medianas que requieran automatizaciones con costes mínimos y sin grandes infraestructuras.

- Proyectos de tipo DIY, donde el factor económico y la autonomía local (sin nube) sean prioritarios.

6.3 Recomendaciones y Trabajos Futuros

Mejora en la Accesibilidad

Sería deseable explorar opciones como el alojamiento externo del servidor web o la apertura de un puerto en el router, complementado con servicios de DNS dinámico, para habilitar el acceso remoto sin depender exclusivamente de la red local. Además, el desarrollo de una aplicación móvil complementaria podría mejorar significativamente la experiencia del usuario. Estas medidas, sin embargo, deberían ir acompañadas de una implementación robusta de mecanismos de seguridad para la interfaz web, garantizando la protección de los datos y la privacidad del sistema.

Incorporación de Seguridad y Autenticación

Una próxima evolución podría implementar cifrado en ESP-NOW y otros mecanismos de seguridad, especialmente si se añaden funciones más críticas (e.g., cerraduras o sistemas de alarma). De igual forma, cualquier alojamiento externo del servidor web requeriría protocolos seguros y mecanismos de autenticación para garantizar la protección de datos.

Escalabilidad

La expansión de más nodos o el incremento de usuarios conectados simultáneamente al MasterDevice podría degradar el rendimiento. En casos en los que se prevenga una carga mayor a la estudiada en este proyecto, convendría plantear distribuir la carga en varios MasterDevice o utilizar hardware más potente para la gestión central, asegurando la latencia y fiabilidad del sistema.

Refinamiento de Automatizaciones

Resultaría interesante indagar en reglas avanzadas, como perfiles de usuario, franjas horarias detalladas o algoritmos de optimización basados en aprendizaje automático. Esto podría ajustarse aún más a las preferencias energéticas y de confort de cada hogar.

También se podría incorporar compatibilidad con otras variedades de dispositivos o de distintos fabricantes, aumentando la portabilidad del sistema.

Diseño de Placas PCB

Para reducir el tamaño del sistema, especialmente de los RoomNodes y SensorNodes, sería aconsejable implementar un diseño basado en placas PCB personalizadas que integren todos los componentes necesarios, preferiblemente sustituyéndolos por sus equivalentes SMD.

REFERENCIAS

- [1] J. C. González, «Hogar Domótico (V): KNX», Revista Todo. Accedido: 17 de diciembre de 2024. [En línea]. Disponible en: <https://revistatodo.com/hogar-domotico-v-knx/>
- [2] «7_Qué es KNX.pdf». Accedido: 17 de diciembre de 2024. [En línea]. Disponible en: https://www.knx.es/_data/landingsfiles/7_Qu%C3%A9%20es%20KNX.pdf
- [3] «4 razones por las que utilizar el estándar KNX», IDomótica. Accedido: 17 de diciembre de 2024. [En línea]. Disponible en: <https://iddomotica.com/4-razones-por-las-que-utilizar-el-estandar-knx-2/>
- [4] «Gira KNX RF: Ampliar o completar las instalaciones KNX de modo inalámbrico, rápida y fácilmente». Accedido: 14 de enero de 2025. [En línea]. Disponible en: https://partner.gira.com/es_ES/systeme/knx-system/knx-produkte/knx-rf.html#cms-anchor-intro
- [5] «Domotica: Protocolos», enerxia.net. Accedido: 19 de diciembre de 2024. [En línea]. Disponible en: <https://www.enerxia.net/portal/index.php/i-domo/894-domotica-protocolos>
- [6] «¿Cómo funciona Philips Hue?», Philips Hue ES. Accedido: 19 de diciembre de 2024. [En línea]. Disponible en: <https://www.philips-hue.com/es-es/explore-hue/how-it-works>
- [7] «ESP-NOW - ESP32 - — ESP-IDF Programming Guide v5.3.2 documentation». Accedido: 19 de diciembre de 2024. [En línea]. Disponible en: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_now.html
- [8] «5 razones para usar KNX y 5 razones para no usar KNX - Domótica en Casa». Accedido: 19 de diciembre de 2024. [En línea]. Disponible en: <https://domoticaencasa.es/5-razones-para-usar-knx-y-5-razones-para-no-usar-knx/>
- [9] «Wi-Fi API - - — Arduino ESP32 latest documentation». Accedido: 21 de diciembre de 2024. [En línea]. Disponible en: <https://docs.espressif.com/projects/arduino-esp32/en/latest/api/wifi.html>
- [10] D. Eichhorn, «ESP32 - Ultra-Long Battery Life With ESP-NOW • ThingPulse», ThingPulse. Accedido: 21 de diciembre de 2024. [En línea]. Disponible en: <https://thingpulse.com/esp32-ultra-long-battery-life-with-espnow/>
- [11] D. Eridani, A. F. Rochim, y F. N. Cesara, «Comparative Performance Study of ESP-NOW, Wi-Fi, Bluetooth Protocols based on Range, Transmission Speed, Latency, Energy Usage and Barrier Resistance», en *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)*, Semarangin, Indonesia: IEEE, sep. 2021, pp. 322-328. doi: 10.1109/iSemantic52711.2021.9573246.
- [12] «ESP-NOW - - — ESP-FAQ latest documentation». Accedido: 21 de diciembre de 2024. [En línea]. Disponible en: <https://docs.espressif.com/projects/esp-faq/en/latest/application-solution/esp-now.html#esp-now-allows-pairing-with-a-maximum-of-20-devices-is-there-a-way-to-control-more-devices>
- [13] D. Conran, *crankyoldgit/IRremoteESP8266*. (14 de enero de 2025). C++. Accedido: 14 de enero de 2025. [En línea]. Disponible en: <https://github.com/crankyoldgit/IRremoteESP8266>
- [14] ElevenPaths, «Los 433 MHz y el software libre. Parte 2.», Telefónica Tech. Accedido: 9 de enero de 2025. [En línea]. Disponible en: https://telefonicatech.com/blog/los-433-mhz-y-el-software-libre-parte-2?utm_source=chatgpt.com
- [15] «ESP32 Boards Comparison», Google Docs. Accedido: 25 de diciembre de 2024. [En línea]. Disponible en: https://docs.google.com/spreadsheets/d/1Mu-bNwpnkiNUiM7f2dx8-gPnIAFMibsC2hMIWhIHbPQ/edit?usp=sharing&usp=embed_facebook
- [16] Andreas Spiess, *Prueba de placas ESP32 a batería (Olimex, TinyPICO, EzSBC, TTGO)*, (20 de junio de 2021). Accedido: 25 de diciembre de 2024. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=ajt7vtgKNNM>
- [17] «HLK-LD2410C_datasheet.pdf». Accedido: 19 de enero de 2025. [En línea]. Disponible en: https://naylampmechatronics.com/img/cms/001080/HLK-LD2410C_datasheet.pdf
- [18] alldatasheet.es, «FS1000A PDF». Accedido: 9 de enero de 2025. [En línea]. Disponible en: <http://www.alldatasheet.es/datasheet-pdf/view/1568235/LPRS/FS1000A.html>

- [19]«TSL25911_Datasheet_EN_v1.pdf». Accedido: 9 de enero de 2025. [En línea]. Disponible en: https://cdn-shop.adafruit.com/datasheets/TSL25911_Datasheet_EN_v1.pdf
- [20]«Datasheet_SHT3x_DIS.pdf». Accedido: 9 de enero de 2025. [En línea]. Disponible en: https://sensirion.com/media/documents/213E6A3B/63A5A569/Datasheet_SHT3x_DIS.pdf
- [21]adafruit/Adafruit_TSL2591_Library. (4 de mayo de 2024). C++. Adafruit Industries. Accedido: 14 de enero de 2025. [En línea]. Disponible en: https://github.com/adafruit/Adafruit_TSL2591_Library
- [22]adafruit/Adafruit_SHT31. (11 de enero de 2025). C++. Adafruit Industries. Accedido: 14 de enero de 2025. [En línea]. Disponible en: https://github.com/adafruit/Adafruit_SHT31
- [23]I. Veltchev, *iavorvel/MyLD2410*. (19 de enero de 2025). C++. Accedido: 19 de enero de 2025. [En línea]. Disponible en: <https://github.com/iavorvel/MyLD2410>
- [24]M. N. Dev, *me-no-dev/ESPAsyncWebServer*. (14 de enero de 2025). C++. Accedido: 14 de enero de 2025. [En línea]. Disponible en: <https://github.com/me-no-dev/ESPAsyncWebServer>
- [25]«RTOS Fundamentals - FreeRTOS™». Accedido: 9 de enero de 2025. [En línea]. Disponible en: <https://freertos.org/Documentation/01-FreeRTOS-quick-start/01-Beginners-guide/01-RTOS-fundamentals>
- [26]Warren Gay, *FreeRTOS for ESP32 and Arduino*, 2020.^a ed. Elektor International Media BV, 2020. [En línea]. Disponible en: <https://www.elektor.com/products/freertos-for-esp32-arduino-e-book>
- [27]H. Assistant, «Home Assistant», Home Assistant. Accedido: 9 de enero de 2025. [En línea]. Disponible en: <https://www.home-assistant.io/>
- [28]«Conexión segura de dispositivos IoT – Precios de AWS IoT Core – Amazon Web Services», Amazon Web Services, Inc. Accedido: 5 de enero de 2025. [En línea]. Disponible en: <https://aws.amazon.com/es/iot-core/pricing/>
- [29]«WebSocket vs. HTTP communication protocols: What's the difference for developers?», Sendbird. Accedido: 9 de enero de 2025. [En línea]. Disponible en: <https://sendbird.com/developer/tutorials/websocket-vs-http-communication-protocols>
- [30]«WebSockets vs HTTP: Which to choose for your project in 2024», Ably Realtime. Accedido: 9 de enero de 2025. [En línea]. Disponible en: <https://ably.com/topic/websockets-vs-http>
- [31]«DNS dinámico gratuito - DNS Administrado - Managed Email - Registración de Dominio - No-IP». Accedido: 15 de enero de 2025. [En línea]. Disponible en: <https://www.noip.com/es-MX>
- [32]Espressif Systems., «ESP32 Series Datasheet». Espressif Systems, V4.7. [En línea]. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- [33]L. Llamas, «Qué pines puedo usar en un ESP32», Luis Llamas. Accedido: 2 de enero de 2025. [En línea]. Disponible en: <https://www.luisllamas.es/que-pines-puedo-usar-esp32/>
- [34]E. Guru, «File management on ESP32: SPIFFS and LittleFS compared - Techrm». Accedido: 5 de enero de 2025. [En línea]. Disponible en: <https://www.techrm.com/file-management-on-esp32-spiffs-and-littlefs-compared/>
- [35]R. Bacon, *RalphBacon/203-SPIFFS-vs-LITTLEFS*. (9 de noviembre de 2024). C++. Accedido: 5 de enero de 2025. [En línea]. Disponible en: <https://github.com/RalphBacon/203-SPIFFS-vs-LITTLEFS>

ANEXOS

Anexo I: Estructura de la Trama ESP-NOW

El protocolo ESP-NOW utiliza una estructura de trama basada en el estándar IEEE 802.11, adaptada para optimizar la comunicación directa entre dispositivos sin necesidad de un router o infraestructura adicional. A continuación, se detalla la composición general de una trama ESP-NOW, con explicaciones sobre cada campo y su funcionalidad:

Encabezado MAC	Código de Categoría	Identificador de Organización	Valores Aleatorios	Contenido Específico	FCS
24 bytes	1 byte	3 bytes	4 bytes	7-257 bytes	4 bytes

Tabla 9. Estructura General de una Trama ESP-NOW

- **Encabezado MAC (24 bytes):**

Encabezado estándar de IEEE 802.11. Incluye las direcciones MAC del emisor y del receptor, así como información sobre el tipo y subtipo de la trama. Su función principal es garantizar que los datos lleguen al destino correcto dentro de la red local.

- **Código de Categoría (1 byte):**

Identifica la categoría del mensaje. En ESP-NOW, este campo tiene un valor fijo para distinguirlo de otras tramas IEEE 802.11 estándar.

- **Identificador de Organización (3 bytes):**

Especifica un identificador único asignado al fabricante, en este caso, Espressif Systems. Este campo asegura la compatibilidad y la identificación del protocolo propietario ESP-NOW.

- **Valores Aleatorios (4 bytes):**

Contiene valores generados aleatoriamente para evitar colisiones de datos y garantizar la unicidad de la trama en entornos concurridos.

- **Contenido Específico del Proveedor (7-257 bytes):**

Este es el contenido principal de la trama, donde se transportan los datos definidos por el usuario. En el contexto de este proyecto, se ha implementado una estructura personalizada

- **Secuencia de Comprobación de Trama (FCS) (4 bytes):**

Campo de control que asegura la integridad de la trama mediante la detección de errores en la transmisión. El cálculo se realiza mediante un algoritmo CRC-32.

Anexo II: Cálculo de la Duración de la Batería

El objetivo de este anexo es estimar la autonomía de una batería LiPo de 3.7 V y 1000 mAh, utilizada en el SensorNode. En operación normal, el SensorNode permanece en modo *Deep Sleep* la mayor parte del tiempo y se despierta en intervalos de 15 minutos para realizar una medición de temperatura y humedad, enviar un mensaje ESP-NOW, esperar un ACK y retornar al *Deep Sleep*. La duración total de la actividad en cada despertar se ha medido cerca de 200 ms, valor basado en las observaciones del puerto serie (véase el registro en el Anexo III).

Dado que se establece un ciclo de 15 minutos (900 segundos) y un tiempo activo de 0.2 segundos, significa que, en aproximadamente 1 hora (4 ciclos), el SensorNode permanecerá 0.8 segundos en activo y 3600 segundos en reposo profundo.

En modo activo, se asume un consumo aproximado de 100 mA, basado en las indicaciones de consumo con el módulo RF activo del *datasheet* del ESP32[32], mientras que se consideran insignificantes los consumos de componentes secundarios como el sensor SHT31 y otros elementos de la placa ESP32, como el regulador de voltaje. Mientras que en modo *Deep Sleep*, el módulo completo del microcontrolador requiere alrededor de 70 μ A [15], [16]. Por comodidad de cálculo, se utilizan horas como unidad de tiempo en el cómputo de la capacidad consumida en mAh.

Por tanto, el consumo durante los 0.8 segundos de actividad en una hora se estima en:

$$0.8s \times \frac{1h}{3600s} \times 100mA \approx 0.0222 mAh$$

Mientras que el consumo durante los 3600 segundos restantes de estado de reposo *Deep Sleep* se estiman en:

$$3600s \times \frac{1h}{3600s} \times 0.07mA = 0.07 mAh$$

Por tanto, el consumo del sistema durante una hora (4 ciclos de 15min) es de aproximadamente de $0.0222mAh + 0.07mAh = 0.0922mAh$. Dado que en un día hay 24h el consumo del sistema durante un día es de:

$$0.0922mAh \times 24h \approx 2.2128 mAh/día$$

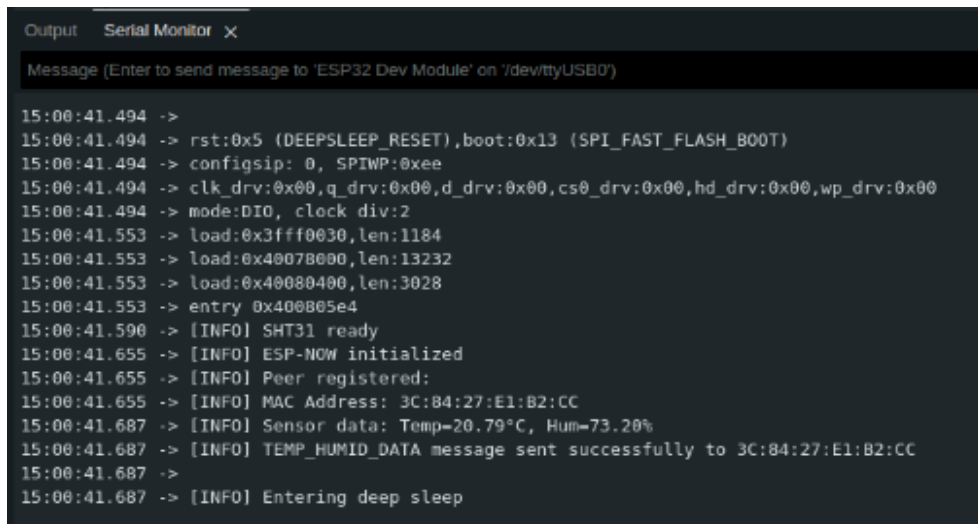
Dado que se dispone de 1000 mAh de capacidad útil, implicaría una duración teórica de alrededor de $1000 mAh / 2.2128 mAh/día \approx 451,92$ días, lo que equivaldría a casi dos años de autonomía.

A pesar de este resultado, es importante tener en cuenta factores como la autodescarga de la batería, las condiciones reales de transmisión, la variación de la temperatura ambiental y posibles reintentos de comunicación, que pueden reducir esta cifra en la práctica.

Anexo III: Justificación Duración 200ms de Ciclo de Actividad

En este anexo se presentan los registros y metodología empleados para estimar la duración aproximada del periodo de actividad del SensorNode en 200ms.

Esta duración está basada en múltiples registros en la consola del Arduino IDE del SensorNode, todos ellos mostrando una duración de la actividad de alrededor 200ms, como se puede observar en la Figura 27:



```
Output Serial Monitor X
Message (Enter to send message to 'ESP32 Dev Module' on '/dev/ttyUSB0')

15:00:41.494 ->
15:00:41.494 -> rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
15:00:41.494 -> config: 0, SPIWP:0xee
15:00:41.494 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
15:00:41.494 -> mode:DIO, clock div:2
15:00:41.553 -> load:0x3fff0030,len:1184
15:00:41.553 -> load:0x40078000,len:13232
15:00:41.553 -> load:0x40080400,len:3028
15:00:41.553 -> entry 0x400805e4
15:00:41.590 -> [INFO] SHT31 ready
15:00:41.655 -> [INFO] ESP-NOW initialized
15:00:41.655 -> [INFO] Peer registered:
15:00:41.655 -> [INFO] MAC Address: 3C:84:27:E1:B2:CC
15:00:41.687 -> [INFO] Sensor data: Temp=20.79°C, Hum=73.20%
15:00:41.687 -> [INFO] TEMP_HUMID_DATA message sent successfully to 3C:84:27:E1:B2:CC
15:00:41.687 ->
15:00:41.687 -> [INFO] Entering deep sleep
```

Figura 27. Captura de Pantalla de la Consola de Arduino IDE.

En la figura se puede observar mensajes de depuración enviados por el microcontrolador al ordenador mediante comunicación serial y mostrados en el terminal del Arduino IDE, que introduce instantes de tiempo para cada mensaje. Podemos observar que el tiempo que transcurre desde que el microcontrolador carga el gestor de arranque (*bootloader*) hasta que entra de nuevo en modo de descanso profundo (*Deep Sleep*) es de aproximadamente 200ms. Dado que el *Jitter* de la comunicación serial se puede dar por negligible, este valor resulta una aproximación válida.

Comparando con otros ciclos se ha podido observar ligeras variaciones en los logs que oscilan en los 5ms de diferencia. Esta pequeña variación es probablemente debida a la variabilidad en el procesamiento y envío del ACK por parte del MasterDevice, la calidad de la señal o el *overhead* de la propia inicialización interna.

Anexo IV: Pines Disponibles en el ESP32

La siguiente tabla resume los pines disponibles en el ESP32, sus funciones principales y recomendaciones de uso. Esta información ha sido obtenida y adaptada del contenido publicado en un blog especializado en electrónica y programación[33].

GPIO	NOMBRE	PUEDE USARSE
0	IO0	Precaución. Configurada como Pull-UP. Debe ser HIGH al arranque. Low al flasheo. Genera PWM al arrancar.
1	TX	No. Pin TX. Salida de depuración al arrancar
2	IO2	Precaución. Debe ser flotante o LOW para flasheo. Conectado al LED.
3	RX	No. Pin RX. En HIGH al arrancar
4	IO4	Sí.
5	IO5	Precaución. Debe ser HIGH al arrancar. Genera PWM al arrancar.
6	CLK	No. FLASH SPI
7	D0	No. FLASH SPI
8	D1	No. FLASH SPI
9	D2	No. FLASH SPI
10	D3	No. FLASH SPI
11	CMD	No. FLASH SPI
12	IO12	Precaución. Debe ser LOW al arrancar. Debug JTAG.
13	IO13	Precaución. No emplear durante Debug JTAG
14	IO14	Precaución. Genera señal PWM al arrancar. Debug JTAG.
15	IO15	Precaución. Debe ser HIGH al arrancar. Debug JTAG.
16	IO16	Sí.
17	IO17	Sí.
18	IO18	Sí.
19	IO19	Sí.
21	IO21	Sí.
22	IO22	Sí.
23	IO23	Sí.
25	IO25	Sí.
26	IO26	Sí.
27	IO27	Sí.

32	IO32	Sí.
33	IO33	Sí.
34	IO34	Precaución. Solo entrada
35	IO35	Precaución. Solo entrada
36	VP	Precaución. Solo entrada
37	VP	Precaución. Solo entrada
38	VP	Precaución. Solo entrada
39	VN	Precaución. Solo entrada
EN	EN	No.

Tabla 10. Pines Disponibles en el ESP32

Anexo V: Tabla de conexiones del RoomNode y SensorNode

Componente	Conexión
LD2410C	
UART_Tx	GPIO16 (Serial2 Rx).
UART_Rx	GPIO17 (Serial2 Rx).
OUT	GPIO14
GND	Conexión a tierra.
VCC	5V (Entrada de alimentación)
TSL2591	
VIN	3.3V (Salida ESP32).
GND	Conexión a Tierra
SDA	GPIO21 (línea de datos I2C).
SCL	GPIO22 (línea de reloj I2C).
INT	No conectado
3VO	No conectado
FS1000A (RF)	
DATA	GPIO13

GND	Conexión a tierra.
VCC	3.3V (Salida ESP32)
Emisor IR	
DATA	GPIO15
GND	Conexión a tierra.
VCC	5V (Entrada de Alimentación).

Tabla 11. Conexiones RoomNode

Componente	Conexión
SHT31	
VIN	3.3V ESP32
GND	GND ESP32
SDA	GPIO21 (línea de datos I2C).
SCL	GPIO22 (línea de reloj I2C).
Batería LiPo	
Positivo (B+)	Pin BAT ESP32
Negativo (B-)	GND ESP32

Tabla 12. Conexiones SensorNode

Anexo VI: LittleFS

LittleFS (Little Flash File System) es un sistema de archivos diseñado específicamente para dispositivos embebidos con memoria flash, como el ESP32. Surge como una alternativa más robusta y confiable frente a SPIFFS (SPI Flash File System), que fue ampliamente utilizado en este tipo de entornos, pero presenta ciertas limitaciones en términos de integridad y flexibilidad [34], [35].

Características Principales

LittleFS destaca por su enfoque en la confiabilidad y eficiencia en dispositivos de recursos limitados. A continuación, se enumeran algunas de sus características más relevantes:

- Resiliencia ante fallos: Integra un sistema de journaling que asegura la consistencia de los datos.

- Soporte para directorios: Aunque LittleFS permite estructuras de archivos más complejas con directorios anidados, en este proyecto no se utilizan, ya que los archivos estáticos se sirven directamente desde la raíz del sistema de archivos, simplificando la implementación.
- Nivelación de desgaste (“Wear Leveling”): Optimiza la distribución de las operaciones de escritura y borrado para prolongar la vida útil de la memoria flash.
- Eficiencia en recursos: Aunque tiene un ligero sobrecoste en comparación con SPIFFS, su diseño sigue siendo adecuado para dispositivos con recursos limitados, ofreciendo un buen equilibrio entre robustez y tamaño.

Ventajas frente a SPIFFS

Si bien SPIFFS fue durante años la solución estándar para la gestión de archivos en microcontroladores, se ha escogido LittleFS debido a su:

- Mayor fiabilidad: Mientras que SPIFFS es susceptible a la corrupción de datos en caso de fallos de energía, LittleFS minimiza este riesgo mediante mecanismos avanzados de recuperación.
- Compatibilidad futura: Espressif ha declarado que SPIFFS se encuentra en desuso, recomendando LittleFS como el estándar para nuevos desarrollos, garantizando soporte y actualizaciones continuas[35].

Anexo VII: Cálculo de Costes por Nodo

Este anexo detalla el cálculo de costes asociados a la implementación del sistema domótico propuesto, desglosando el precio de cada componente necesario para los distintos tipos de nodos: SensorNodes, RoomNodes y MasterDevice.

Las estimaciones de costes se basan en precios obtenidos en la plataforma de comercio electrónico Aliexpress. No se han considerado costes de cables, encapsulado del dispositivo, envío u otros componentes adicionales (e.g., alimentación del RoomNode o MasterDevice).

SensorNode

El coste total por SensorNode es de aproximadamente 13 euros por unidad.

Componente	Descripción	Precio Unitario (€)
LOLIN D32	Microcontrolador ESP32	4
Sensor SHT31	Sensor de temperatura y humedad	2
Módulo de carga TP4056	Módulo de carga para baterías	2

Batería LiPo 3.7V 1000mAh	Batería recargable	7
----------------------------------	--------------------	---

Tabla 13. Coste de Componentes SensorNode

RoomNode

El coste total por RoomNode es de aproximadamente 13.5 euros por unidad.

Componente	Descripción	Precio Unitario (€)
ESP32 DevKit	Microcontrolador ESP32	3
LD2410C	Sensor de presencia radar	3
Transmisor RF 433 MHz	Módulo transmisor de radiofrecuencia	1
Emisor IR	Módulo emisor infrarrojo	2
Sensor de luz TSL2591	Sensor de luminosidad ambiental	4.5

Tabla 14. Coste de Componentes RoomNode

MasterDevice

El coste total por MasterDevice es de aproximadamente 4 euros por unidad, ya que solo requiere de:

Componente	Descripción	Precio Unitario (€)
ESP32-S3	Microcontrolador ESP32	4

Tabla 15. Coste de Componentes MasterDevice

Anexo VIII: Montaje Físico Entorno de Pruebas

El entorno de pruebas consiste en tres SensorNodes, un RoomNode y un MasterDevice. Para visualizar su distribución concreta en la casa consultar Figura 26.

Un nodo de los SensorNodes es alimentado por batería, para comprobar su correcto funcionamiento, el resto, por comodidad se alimentan por cable:



Figura 28. SensorNode en la Cocina alimentado por batería.



Figura 29. SensorNode en la Habitación 3 alimentado por cable.

En la Habitación 1, se encuentran 1 nodo de cada tipo, conectados al ordenador para realizar pruebas y visualizar los mensajes de depuración:

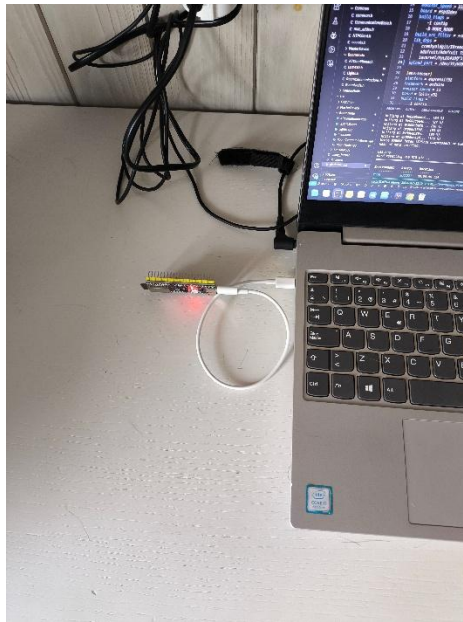


Figura 30. MasterDevice en la Habitación 1.

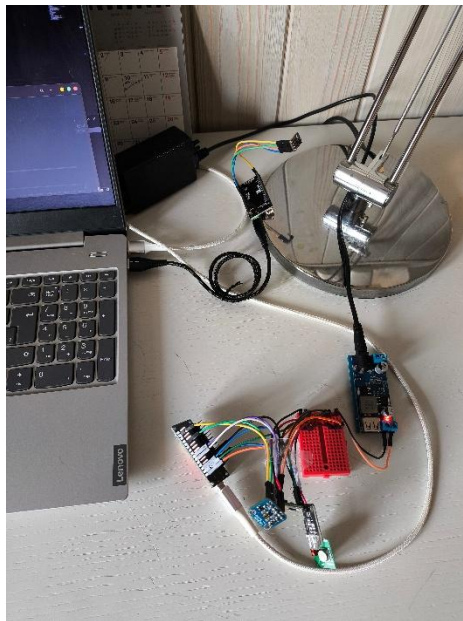


Figura 31. SensorNode y RoomNode en la Habitación 1.

Para poder conectar el RoomNode al ordenador, se ha implementado una pequeña variación respecto al esquema de conexión proporcionado en la Figura 18. Como se puede ver en la Figura 32 el cable que alimentaba con 5V el pin VIN del ESP32 se ha eliminado y en su lugar se alimenta mediante USB.

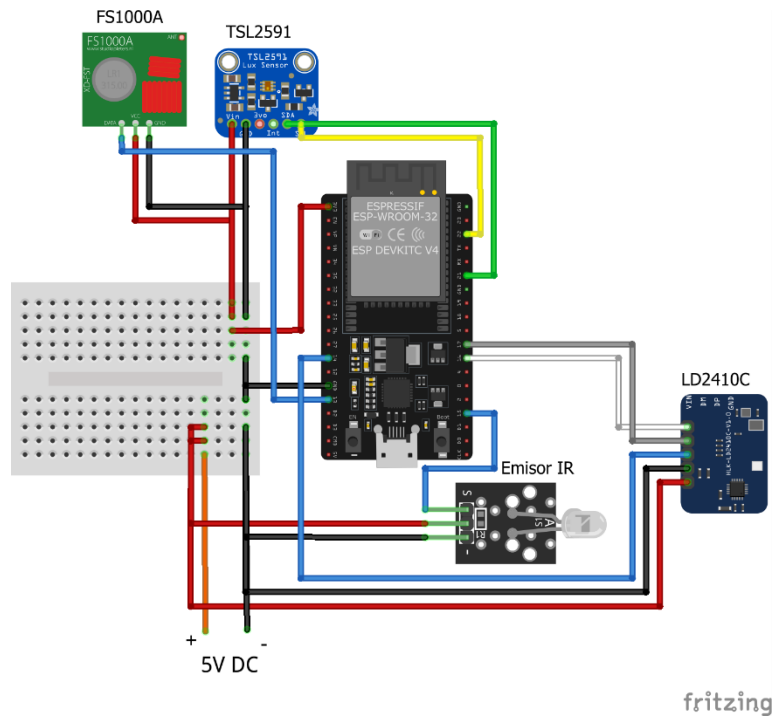


Figura 32. Esquema de Conexión RoomNode Pruebas

Además, se ha empleado el módulo convertidor *Buck* para proporcionar 5V con una corriente de 1.5A, más que suficiente para alimentar los módulos que funcionan a este voltaje.



Figura 33. Buck Converter 24-12V a 5V