UNIVERSIDAD POLITÉCNICA DE PUEBLA

MAESTRÍA EN INGENIERÍA EN AUTOMATIZACIÓN DE PROCESOS INDUSTRIALES

Tesis para obtener el grado de

MAESTRO EN INGENIERIA EN AUTOMATIZACION DE PROCESOS INDUSTRIALES

DESARROLLO DE UNA RED INALÁMBRICA AD HOC PARA REALIZAR TAREAS DE MONITOREO DE TEMPERATURA Y HUMEDAD

Presenta:

José Guadalupe Xicotencatl Flores

Asesores:

Dr. Juan Antonio Arizaga Silva

Dr. Mario Espinosa Tlaxcaltecatl

Puebla, Pue., junio 2024

Agradecimientos

Quiero expresar mi profundo agradecimiento a todas las personas que me han brindado su apoyo incondicional durante mi trayectoria académica y la realización de esta tesis.

Agradezco a mis asesores, el Dr. Juan Antonio Arizaga Silva y el Dr. Mario Espinosa Tlaxcaltecatl, por su guía, dedicación y paciencia en este proyecto.

También agradezco a la Universidad Politécnica de Puebla por proporcionarme las herramientas necesarias para llevar a cabo este estudio.

Rindo homenaje a mi padre, Magdaleno, quien nos dejó antes de verme culminar esta etapa. Su amor, sabiduría y constante motivación son mi mayor inspiración. Le dedico este logro y sé que su espíritu siempre estará presente en mi vida.

Agradezco a mi madre, Claudia. Su amor incondicional y sacrificios han sido pilares fundamentales en mi camino hacia la realización de mis sueños

A mi hermana, Elizabeth, por su guía constante en cada paso del camino. Mi admiración hacia ella me ha impulsado a superar los desafíos y alcanzar mis metas.

Y a mi esposa, Clara Belén, por su amor y apoyo incondicional. Su comprensión y paciencia han sido invaluables durante esta etapa de mi vida. Su presencia me ha dado fuerzas para seguir adelante y nunca rendirme.

A todas estas personas especiales, les agradezco de corazón por creer en mí y estar a mi lado en este viaje. Sin ustedes, este logro no hubiera sido posible.

¡Gracias, Dios los bendiga!

Resumen

En esta tesis se presenta un sistema integral de monitoreo de temperatura y humedad diseñado para proporcionar una solución efectiva y versátil en aplicaciones IoT. El sistema se basa en la implementación de una red ad hoc utilizando chips ESP32 con ESP Mesh, lo que permite una comunicación dinámica y autónoma entre los nodos distribuidos en el entorno a monitorear. Además, se diseñaron tarjetas específicas que integran el microcontrolador ESP32-C3-WROOM-02, junto con los sensores de temperatura y humedad DHT 11.

Una característica clave de este sistema es la inclusión de una batería de polímero de litio en cada nodo, lo que garantiza una operación autónoma y sin interrupciones. La batería proporciona la energía necesaria para el funcionamiento continuo del nodo, asegurando la captura y transmisión de datos incluso en condiciones adversas o en ausencia de una fuente de alimentación externa.

La información recopilada por los nodos, que incluye lecturas de temperatura y humedad, se envía de forma inalámbrica a través de la red ad hoc utilizando el protocolo ESP Mesh. Posteriormente, los datos son transmitidos a una página web dedicada utilizando el protocolo MQTT. Esta página web, implementada en la plataforma MQTTx, ofrece una interfaz amigable y accesible para la visualización en tiempo real de los datos de temperatura y humedad.

Abstract

This thesis presents a comprehensive system for monitoring temperature and humidity designed to provide an effective and versatile solution in IoT applications. The system is based on the implementation of an ad hoc network using ESP32 chips with ESP Mesh, enabling dynamic and autonomous communication between nodes distributed in the monitored environment. Additionally, specific cards were designed that integrate the ESP32-C3-WROOM-02 microcontroller, along with temperature and humidity sensors DHT 11.

A key feature of this system is the inclusion of a lithium polymer battery in each node, ensuring uninterrupted and autonomous operation. The battery provides the necessary energy for the continuous operation of the node, ensuring the capture and transmission of data even in adverse conditions or in the absence of an external power source.

The information collected by the nodes, including temperature and humidity readings, is wirelessly transmitted through the ad hoc network using the ESP Mesh protocol. Subsequently, the data is transmitted to a dedicated web page using the MQTT protocol. This web page, implemented on the MQTTx platform, offers a user-friendly and accessible interface for real-time visualization of temperature and humidity data.

Abreviaturas

IoT Internet de las Cosas, *Internet of Things.*

LAN Red de Área Local, *Local Area Network.*

WAN Red de Área Amplia, *Wide Area Network.*

UART Receptor-Transmisor Asíncrono Universal, *Universal Asynchronous Receiver-Transmitter.*

PAN Red de Área Personal, *Personal Area Network.*

WLAN Red de Área Local Inalámbrica, *Wireless Local Area Network.*

SAN Red de Área de Almacenamiento, *Storage Area Network.*

RAN Red de Acceso de Radio, *Radio Access Network.*

RAM Memoria de Acceso Aleatorio, *Random Access Memory.*

PWM Modulación por Ancho de Pulso, *Pulse Width Modulation.*

BLE Bluetooth de baja energía, *Bluetooth Low Energy.*

HTTP Protocolo de Transferencia de Hipertexto, *Hypertext Transfer Protocol.*

CoAP Protocolo de Aplicación Restringida, *Constrained Application Protocol.*

PCB Placa de Circuito Impreso, *Printed Circuit Board.*

IDE Entorno de Desarrollo Integrado, *Integrated Development Environment.*

MQTT Transporte de Telemetría de Mensajes en Cola, *Message Queuing Telemetry Transport.*

WROOM Módulo Incorporado de Radio Wi-Fi, *Wi-Fi Radio Onboard Module.*

DHT Humedad y Temperatura Digital, *Digital Humidity and Temperature.*

LoRa Largo Alcance, *Long Range.*

SSL Capa de Conexión Segura, *Secure Sockets Layer.*

TLS Seguridad de la Capa de Transporte, *Transport Layer Security.*

DFN Dispositivo de No Terminales Planos Doble, *Dual Flat No-leads.*

SOT Transistor de Contorno Pequeño, *Small Outline Transistor.*

Índice general

Índice de figuras 7

Índice de tablas 9

1. Introducción 10

1.1. Justificación ……..11
1.2. Objetivos……12
1.3. Alcances del proyecto……12
1.4. Contribuciones……...12
1.5. Estructura del documento……...12
1.6. Diagrama del sistema implementado……13

2. Trabajo Relacionado 14

2.1. Sistemas de monitoreo, control y comunicación entre módulos ESP32…….....14
2.2. Visualización de datos a través de una plataforma web……..…18

3. Marco Teórico 22

3.1. Redes ad hoc……..22
3.2. Microcontrolador ESP32……...23
3.3. Creación de redes ad hoc con el chip ESP32…….....27
3.4. Protocolos de comunicación entre ESP32 y una página web…….....32
3.4.1 Protocolo MQTT: conexión web-modulo ESP32……..34
3.5. Evaluación de materiales para el desarrollo del proyecto……...37
3.5.1. Sensores de temperatura y humedad ……..…...37

3.5.2. Baterías recurrentes en proyectos similares…….……38

4. Desarrollo 41

4.1. Análisis de los materiales utilizados en el desarrollo del proyecto……...….41

4.1.1. Microcontrolador ESP32-C3-WROOM-02 ……...…….41

4.1.2. Sensor DHT11…….….45

4.1.3. Batería recargable de polímero de litio…….46

4.2. Diseño y seccionamiento del circuito eléctrico implementado……..48

4.2.1 Circuito de activación……50

4.2.2. Circuito de carga……...…51

4.2.3. Circuito regulador de voltaje……54

4.2.4. Circuito de reparto de carga……...…57

4.2.5. Esquemático completo del circuito……..59

4.3. Diseño de la tarjeta PCB ……...60

4.3.1. Soldar los componentes en la PCB……..62

4.4. Programar los chips ESP32-C3-WROOM-02 …….....……63

4.4.1. Configurar la red ad hoc con ESP Mesh y MQTT……..67

4.5. Configuración a través de la página través de la página MQTTX……82

5. Resultados 86

6. Conclusiones 88

Referencias 90

Índice de figuras

Figura 1. Diagrama del sistema implementado……13

Figura 2. Microcontrolador ESP32-C3-WROOM-02……..43

Figura 3. Diseño de pines del Microcontrolador
ESP32-C3-WROOM-02 (vista superior)…….…….....43

Figura 4. Sensor DHT11…….……..45

Figura 5. Batería recargable de polímero de litio 3.7v 1000mAh……..46

Figura 6. Esquema de conexión de la Batería LiPo
con el ESP32-C3-WROOM-02 ……...…….47

Figura 7. Circuito de activación……..……..50

Figura 8. Formatos de paquete para MCP73831/2…….……..51

Figura 9. Circuito de carga utilizando el dispositivo MCP73831……..53

Figura 10. Perfil de carga para una batería de
polímero de litio de 1000 mAh……..…..53

Figura 11. Formatos de paquete para el dispositivo MCP73831……..…….54

Figura 12. Circuito regulador de voltaje……..……....….56

Figura 13. Circuito de reparto de carga…….....….57

Figura 14. Esquemático completo del circuito…….….59

Figura 15. Tarjeta prototipo PCB desarrollada…….…60

Figura 16. Tarjeta final PCB desarrollada…….…61

Figura 17. Tarjeta PCB con sus componentes…….....…..62

Figura 18. Diagrama de flujo del comportamiento del sistema……..…...64

Figura 19. Proceso de Integración de Tarjeta de Desarrollo en el Entorno de Desarrollo…….……65

Figura 20. Selección de placa y modelo del software……..……..66

Figura 21. Pantalla principal de MQTTX…….……83

Figura 22. Creación del proyecto dentro de MQTTx…….……...83

Figura 23. Configuración de los parámetros de conexión ……...84

Figura 24. Estableciendo conexión con el servidor……..…….……84

Figura 25. Configuración de la suscripción en MQTTx……..……..85

Figura 26. Nodo Maestro y ESP Bridge……86

Figura 27. Módulo ESP-WROOM-32 como Nodo Esclavo o Maestro-Esclavo……..87

Figura 28. Nodo esclavo con la Tarjeta PCB desarrollada……87

Figura 29. Interfaz de gestión de mensajes en la plataforma MQTTx……...88

Figura 30. Datos recibidos y procesados por la plataforma MQTTx…….....89

Figura 31. Errores de mensaje en la plataforma MQTTx……..…89

Índice de tablas

Tabla 1. Características empleadas por diversos autores……17

Tabla 2. Características empleadas por diversos autores utilizando una plataforma web…….21

Tabla 3. Ventajas y desventajas de las redes ad hoc……24

Tabla 3.1 Características similares de dispositivos similares al chip esp32…….27

Tabla 3.2. Diferentes maneras de crear una red ad hoc utilizando el chip ESP32……31

Tabla 3.3. Ventajas y desventajas con diferentes formas de hacer una red ad hoc utilizando esp32……32

Tabla 3.4 Ventajas y desventajas de los protocolos utilizados para establecer comunicación bidireccional entre un chip ESP32 y una página web……...35

Tabla 3.5. Ventajas y desventajas entre las plataformas que se pueden utilizar para establecer comunicación con el chip esp32 utilizando el protocolo MQTT……..38

Tabla 3.6. Características de los sensores de temperatura y humedad…….40

Tabla 3.7. Características de las baterías de que se pueden conectar al ESP32……...41

Tabla 3.8. Ventajas y desventajas de las baterías que se pueden conectar al ESP32……42

Tabla 4 Descripción de los pines del Microcontrolador ESP32-C3-WROOM-02……..46

Tabla 4.1. Descripción de los pines del microcontrolador ESP32-C3-WROOM-02……..54

Tabla 4.2. Descripción de los pines del dispositivo MCP1700……...57

Tabla 4.3. Descripción de los pines del circuito de reparto de carga……...60

Capítulo 1

Introducción

A través de los años el avance tecnológico le ha permitido al hombre optimizar diversos procesos. Desde la implantación de las comunicaciones inalámbricas en los distintos sectores de la industria, el cableado físico ha pasado a un segundo plano. Esto se debe a la mayor simpleza de los sistemas inalámbricos respecto a los cableados, así como a la reducción económica que supone prescindir de una red física de comunicaciones.

Una red ¨Ad hoc¨, consiste en un grupo de ordenadores que se comunican cada uno directamente con los otros a través de las señales de radio si usar un punto de acceso. Las configuraciones ¨Ad hoc¨, son comunicaciones de tipo punto a punto. Solamente los ordenadores dentro de un rango de transmisión definido pueden comunicarse entre ellos. La tecnología es utilizada en varios campos como en el ejército, celulares y juegos de videos. En fin, en la tecnología ¨Ad hoc¨, cada terminal de comunicación se comunica con sus compañeros para hacer una red ¨peer to peer¨. Los nodos se conectan de forma directa, dinámica y no jerárquicamente a otros nodos como sea posible y cooperar entre sí para enrutar eficientemente hacia el cliente. Esta falta de dependencia de un nodo, permite a cada nodo participar en la retransmisión de información. en una red de infraestructura tradicional.

Esta topología de red proporciona un área de cobertura mucho mayor ya que los nodos aún pueden lograr la interconectividad sin necesidad de estar en el rango del nodo central. También es menos susceptible a la sobrecarga ya que el número de nodos permitidos en la red ya no es limitado por un solo nodo central. El chip ESP32 es capaz de construir una red de malla, los nodos pueden auto organizarse y comunicarse dinámicamente entre sí. Cualquier nodo dentro de la red es capaz de transmitir paquetes de datos a cualquier otro nodo, dentro del rango, que luego puede reenviar paquetes de datos a través de la red hasta el destino final. Cada chip ESP32 puede actuar como un nodo en la red de malla.

La utilización de chips ESP32 permite una comunicación inalámbrica de bajo consumo de energía y alta velocidad a través de ESP Mesh, lo que posibilita una conexión más estable y confiable con los sensores y nodos en la red ad hoc. La integración con una página web mediante el protocolo MQTT proporciona una interfaz amigable y fácil de usar para visualizar y analizar los datos obtenidos de los nodos en tiempo real. Además, permite la integración con otras herramientas y servicios para mejorar la funcionalidad del sistema de monitoreo.

La información obtenida a través de los sensores, como la temperatura y la humedad, es esencial para la toma de decisiones en diferentes ámbitos, como la agricultura, la industria alimentaria, la construcción, la salud y la seguridad, entre otros. Un sistema de monitoreo con una red ad hoc con chips ESP32 utilizando ESP Mesh y conectado a una página web mediante el protocolo MQTT proporciona una solución efectiva y económica para la adquisición de datos de forma remota y en tiempo real, lo que permite tomar decisiones oportunas y precisas para mejorar los procesos y la calidad de vida en general.

1.1. Justificación

En la actualidad, es cada vez más común la necesidad de monitorear y recolectar datos en ambientes específicos, como por ejemplo en espacios de almacenamiento, invernaderos o incluso en la industria alimentaria. En este sentido, el uso de una red inalámbrica ad hoc con chips ESP32 y ESP Mesh se presenta como una solución efectiva y de bajo costo para llevar a cabo estas tareas de monitoreo.

La importancia de desarrollar una red inalámbrica ad hoc radica en la capacidad de estos dispositivos para establecer una conexión inalámbrica directa sin necesidad de contar con un punto de acceso o una red existente. Esto significa que pueden ser utilizados en ambientes remotos o en lugares donde no se cuenta con una infraestructura de red establecida. Además, al utilizar ESP Mesh, se logra una conexión estable y confiable entre los dispositivos, permitiendo un monitoreo preciso y en tiempo real de las variables ambientales.

En el caso específico de la monitorización de temperatura y humedad, el uso de sensores conectados a los nodos de la red ad hoc permite obtener mediciones precisas y en tiempo real de las variables ambientales en el lugar de interés. Estos datos pueden ser enviados al nodo principal para su procesamiento y análisis, y finalmente transmitidos a una página web mediante el protocolo MQTT para su visualización y gestión remota. La conexión a la página web permite un monitoreo en tiempo real de los datos recolectados y un fácil acceso a la información. Además, la plataforma ofrece herramientas para la configuración y gestión remota de los dispositivos, lo que facilita la implementación y el mantenimiento del sistema de monitoreo.

1.2. Objetivos

Para el trabajo de tesis se fijaron los siguientes objetivos:

Objetivo General

Diseñar e integrar una red inalámbrica Ad hoc con sensores digitales estableciendo comunicación bidireccional entre los nodos de la red mediante el uso del módulo ESP32 y el protocolo ESP Mesh para realizar tareas de control y monitoreo en ambientes de temperatura y humedad. Los datos recolectados por los sensores serán transmitidos a través de la red ad hoc y visualizados en tiempo real mediante una página web utilizando el protocolo MQTT.

Objetivos específicos

1. Diseñar el circuito electrónico del sistema de monitoreo, el cual incluye los chips ESP32 y los sensores de temperatura y humedad.
2. Desarrollar el software necesario para que los chips ESP32 puedan comunicarse entre sí mediante la red ad hoc y transmitir los datos de temperatura y humedad.
3. Integrar los chips ESP32 con una página web utilizando el protocolo MQTT para permitir la visualización y control remoto de los datos.

1.3. Alcances del Proyecto

El monitoreo de temperatura y humedad es importante en una amplia variedad de aplicaciones y entornos, algunos de los lugares más comunes donde se utiliza incluyen:

- Edificios comerciales y residenciales.
- Invernaderos para el cultivo de plantas.
- Bodegas y almacenes para garantizar la calidad de los productos almacenados.
- Hospitales y laboratorios para mantener las condiciones ideales para los pacientes y las muestras.
- Producción de alimentos y bebidas para garantizar la calidad y seguridad de los productos.
- Sistemas de refrigeración y aire acondicionado.
- Agricultura y la ganadería para garantizar el bienestar y la productividad de los animales.
- Procesos industriales y manufactureros para garantizar la calidad de los productos.

1.4. Contribuciones

La principal contribución de este proyecto se centra en la construcción de una red inalámbrica Ad hoc y su aplicación en el monitoreo y control de temperatura y humedad. Sin embargo, es importante destacar que este proyecto tiene el potencial de ser utilizado para otros propósitos. Con los cálculos adecuados, es posible modificar las señales de entrada y salida para trabajar con diferentes tipos de sensores y actuadores, ampliando así su aplicación en distintos escenarios. Esta flexibilidad y adaptabilidad del sistema brinda oportunidades para su implementación en una amplia variedad de campos y áreas de estudio.

1.5. Estructura del documento

La estructura de la tesis se compone de los siguientes capítulos. En el **capítulo 2** se realiza una revisión de trabajos similares de otros autores. En el **capítulo 3** se realiza una introducción a las técnicas utilizadas durante el desarrollo del sistema, donde se desglosan definiciones y términos relacionados a los sistemas de monitoreo, control y comunicación entre módulos ESP32, así como la visualización de datos a través de una plataforma web. En el **capítulo 4** se describen los procesos involucrados para la comunicación entre dos o más dispositivos ESP32 de manera inalámbrica, así como la comunicación entre una plataforma web. En el **capítulo 5** se muestran los resultados obtenidos. En el **capítulo 6** se presentan las conclusiones y el trabajo futuro. Por último, se incluye la bibliografía utilizada como parte del trabajo de investigación.

1.6. Diagrama del sistema implementado

La Figura 1 representa el funcionamiento integral del sistema desarrollado en este estudio. El objetivo principal de este proyecto radica en la implementación de una red ad hoc mediante la tecnología ESP Mesh, la cual actúa como un medio de conexión entre todos los nodos de la red para llevar a cabo el monitoreo de temperatura y humedad.

Cada nodo dentro de la red está equipado con un sensor DHT11 integrado. Los nodos esclavos se encargan de recopilar la información de temperatura y humedad del sensor DHT11 y la envían al nodo maestro. Posteriormente, el nodo maestro transmite estos datos al ESP Bridge a través de comunicación serial, utilizando los puertos TX y RX de ambos dispositivos. Este nuevo ESP32, denominado ESP32 Bridge, se encarga de enviar toda la información hacia la página MQTTX a través de la conexión Wi-Fi, utilizando el protocolo MQTT. Actúa como intermediario y conector entre la red ad hoc y la página web mencionada.

La página Mqttx recibe la información en formato JSON y publica los valores correspondientes al número de nodos dentro de la red, así como los valores de temperatura y humedad asociados a cada nodo. Esta solución proporciona una manera eficaz de supervisar y controlar las condiciones ambientales en tiempo real.

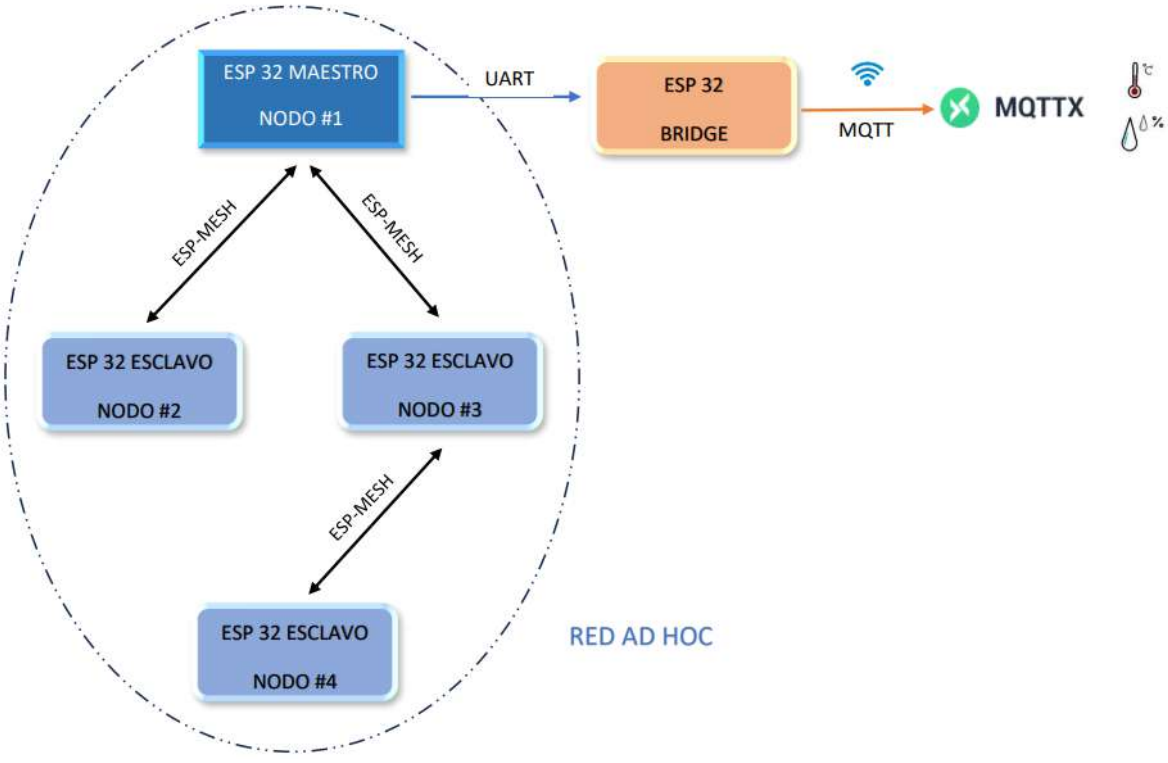


Figura 1. Diagrama del sistema implementado

Capítulo 2

Trabajo Relacionado

En este capítulo se revisan algunos trabajos de diferentes autores para poner en contexto el trabajo realizado. Las Tablas 1 y 2 resumen la metodología empleada por diversos autores tomados para este proyecto. A continuación, se describe lo reportado por cada artículo.

2.1. Sistemas de monitoreo, control y comunicación entre módulos ESP 32

El artículo "Establishing a Wireless-Local-Area-Network (WLAN) Connectivity between Multiple Nodes using ESP-Mesh Network Topology for IoT Applications" [1] describe la implementación de una red de área local inalámbrica (WLAN) utilizando la topología de red ESP-Mesh. El objetivo de este trabajo es proporcionar una solución escalable y rentable para establecer una red inalámbrica de IoT (Internet de las cosas) entre múltiples nodos. Se destaca el uso del chip ESP32 como dispositivo de red principal, que proporciona un conjunto de características útiles para la implementación de redes IoT, como la conectividad Wi-Fi y Bluetooth, así como el soporte para una amplia variedad de sensores y actuadores. Los autores explican cómo se implementó la red ESP-Mesh utilizando el protocolo de comunicación MQTT (Message Queuing Telemetry Transport) y cómo se estableció la comunicación bidireccional entre múltiples nodos en la red.

En [2], I.V. Haro Vilaña describe la implementación de un sistema de monitoreo y control automático de riego para invernaderos utilizando la tecnología LoRa y el microcontrolador ESP32. La tesis presenta los detalles de diseño e implementación de un sistema de bajo costo y alta eficiencia energética que permite la adquisición de datos ambientales y la activación de sistemas de riego de forma remota. El autor destaca el uso del chip ESP32 como dispositivo principal, que proporciona características útiles para la implementación de sistemas de IoT, como la conectividad Wi-Fi y Bluetooth, así como la capacidad de procesamiento y almacenamiento de datos. La tesis también explica cómo se utilizó la tecnología LoRa para la comunicación inalámbrica de larga distancia entre los sensores y la base central de control. El sistema implementado permite el monitoreo en tiempo real de las condiciones ambientales del invernadero, así como la activación automática de sistemas de riego en función de los datos recolectados. Además, el bajo costo y la eficiencia energética del sistema hacen que sea una solución atractiva para los agricultores que buscan mejorar la productividad y reducir los costos operativos.

El artículo "Indoor positioning utilizing bluetooth low energy RSSI on LoRa system" [3] presenta una solución para la localización en interiores utilizando la tecnología Bluetooth Low Energy (BLE) y la red LoRa. El trabajo propone un método para medir la intensidad de la señal recibida (RSSI) utilizando un dispositivo ESP32 para la transmisión de datos a través de la red LoRa.

Los autores realizaron experimentos para evaluar la precisión de la localización en interiores utilizando este enfoque y compararon los resultados con otras tecnologías como el GPS y la triangulación de antenas. Los resultados mostraron que la solución propuesta tiene una precisión razonable y puede ser útil en aplicaciones de localización en interiores. Este trabajo destaca el uso del dispositivo ESP32 para la transmisión de datos a través de la red LoRa, lo que demuestra las capacidades de esta plataforma para la implementación de soluciones de IoT (Internet de las cosas) en una amplia variedad de aplicaciones.

El artículo “Internet of Things Based Home Monitoring and Device Control Using Esp32” [4] describe la implementación de un sistema de monitoreo y control de dispositivos domésticos basado en IoT utilizando el chip ESP32. El objetivo del trabajo es proporcionar una solución para la automatización del hogar, permitiendo a los usuarios controlar y monitorear sus dispositivos desde cualquier lugar utilizando una aplicación móvil. El artículo destaca la capacidad del chip ESP32 para proporcionar conectividad Wi-Fi y Bluetooth de bajo consumo de energía, lo que lo convierte en una opción ideal para aplicaciones de IoT. Además, los autores explican cómo utilizaron el protocolo MQTT (Message Queuing Telemetry Transport) para permitir la comunicación entre los dispositivos y la aplicación móvil. El sistema implementado permite a los usuarios controlar dispositivos como luces, ventiladores y enchufes inteligentes, y también monitorear la temperatura y la humedad del hogar en tiempo real. El sistema se puede controlar mediante una interfaz de usuario intuitiva en la aplicación móvil, lo que lo hace fácil de usar para cualquier persona.

El artículo “Microclimate Monitoring System for a Home Greenhouse as Part of ESP32” [5] presenta la implementación de un sistema de monitoreo de microclima para un invernadero casero utilizando el ESP32. El sistema monitorea la temperatura, la humedad y la intensidad de la luz y utiliza el chip ESP32 como dispositivo principal para adquirir y enviar datos a través de una red Wi-Fi. El sistema también cuenta con una interfaz web para visualizar los datos en tiempo real y ajustar la configuración del sistema. El uso del ESP32 en este sistema demuestra su capacidad para integrarse en soluciones de IoT y monitoreo ambiental de bajo costo y escalables.

El trabajo de investigación "Posicionamiento de una cabina de ascensor mediante un ESP32 y sensores"[6] presenta una solución para el posicionamiento preciso de la cabina de un ascensor utilizando un ESP32 y sensores. El objetivo de este proyecto es mejorar la eficiencia y seguridad de los ascensores. Los autores explican cómo se implementó el sistema utilizando el chip ESP32 como dispositivo principal y sensores de ultrasonido para detectar la posición de la cabina. Además, se utilizó un algoritmo de control PID para regular la velocidad y posición de la cabina. El trabajo destaca la capacidad del ESP32 para procesar la información de los sensores y controlar el movimiento de la cabina de manera efectiva y precisa. Este enfoque representa una solución innovadora para mejorar la seguridad y eficiencia de los ascensores mediante el uso de tecnología IoT.

En [7], los autores discuten los desafíos y obstáculos prácticos de recopilar datos de malla Bluetooth utilizando microcontroladores ESP-32. El artículo se enfoca en la recolección de datos de temperatura y humedad utilizando nodos ESP-32 y sensores Bluetooth Low Energy (BLE). Los autores destacan las dificultades de la implementación, como la falta de documentación clara y los problemas de conectividad de los dispositivos, y proponen soluciones para superar estos desafíos. El artículo proporciona información útil para los desarrolladores de sistemas de IoT que utilizan ESP-32 y tecnología Bluetooth.

El artículo “Sistema Detector de Alacranes Usando IOT con BLE“[8] presenta el diseño y desarrollo de un sistema de detección de alacranes utilizando la tecnología de Internet de las cosas (IoT) y Bluetooth Low Energy (BLE) con el microcontrolador ESP32. El sistema consta de un sensor de vibración y un sensor de temperatura conectados al ESP32, que se encarga de procesar y enviar los datos a un servidor a través de BLE. Los datos se analizan en el servidor y se envía una alerta a los usuarios si se detecta actividad de alacranes. El artículo resalta la importancia de este sistema para la prevención de picaduras de alacrán y cómo la tecnología IoT y el ESP32 pueden ser utilizados para abordar este problema de manera efectiva.

El artículo [9] de M. García Gómez describe el diseño y la implementación de un sistema de control de casa inteligente utilizando el protocolo ESP-NOW. El protocolo permite una comunicación de bajo consumo energético entre dispositivos, lo que lo hace ideal para la implementación de sistemas IoT. El ESP-NOW se utiliza en conjunto con el ESP32, un microcontrolador que permite la conexión inalámbrica y el control de múltiples dispositivos. El sistema implementado se basa en la detección de presencia y la medición de la temperatura y la humedad, lo que permite controlar el encendido y apagado de dispositivos en la casa de manera automática y eficiente. El artículo destaca la importancia del uso de tecnologías de bajo consumo energético en la implementación de sistemas de IoT, y cómo el ESP32 y el protocolo ESP-NOW pueden ser una solución efectiva para este propósito.

En la tesis de J. Peris Martínez, titulada "Sistema de Monitorización Inalámbrica de Temperatura Mediante Sensor de Infrarrojos y el Microcontrolador ESP32"[10], se presenta un sistema de monitorización de temperatura inalámbrico utilizando un sensor de infrarrojos y un microcontrolador ESP32. El trabajo se enfoca en el diseño y desarrollo del sistema, incluyendo la selección de componentes, la programación del microcontrolador y las pruebas realizadas para validar el funcionamiento del sistema. El ESP32 se utiliza para la comunicación inalámbrica y el procesamiento de datos, y se hace énfasis en su importancia para la implementación del sistema.

El artículo "Smart Agro: IOT Based Rice Plant Health Monitoring System" [11] describe un sistema de monitoreo de salud de plantas de arroz basado en IoT. Se utilizó el microcontrolador ESP32 para recopilar datos de sensores que miden la temperatura, la humedad y la calidad del suelo en tiempo real. Estos datos se envían a través de una conexión inalámbrica a una aplicación en la nube para su análisis y visualización. El sistema también utiliza un modelo de aprendizaje automático para predecir la salud de las plantas y proporcionar recomendaciones de cuidado. Este sistema puede ayudar a los agricultores a monitorear y mejorar el crecimiento de las plantas de arroz de manera más eficiente.

Tabla 1. Características empleadas por diversos autores.

Referencia	Protocolo de Comunicación	Implementación
[1]	ESP Mesh	Se utilizan 3 módulos ESP32 para el desarrollo de una red de área local inalámbrica para la transferencia de datos de sensores.
[2]	Wifi	Se utilizan 4 módulos ESP32 para el control y monitoreo de temperatura y humedad dentro de un invernadero, con una pantalla HMI para poder visualizar los datos.
[3]	Bluetooth (BLE)	Sistema de posicionamiento en interiores basado en BLE LoRa, se implementa un filtro de partículas para mejorar la precisión.
[4]	Wifi y Bluetooth	Sistema de vigilancia y control del hogar usando ESP32, a través de Wi-Fi local para acceder y controlar dispositivos por parte de un usuario de forma remota usando una aplicación de teléfono.
[5]	Wifi y Bluetooth	Sistema de monitoreo de invernadero utilizando una red de sensores. Los datos pueden almacenarse en la nube y mostrarse a través de una aplicación móvil. Donde se implementa: refrigeración, riego e iluminación.
[6]	Wifi	Localización de planta a la que se encuentra una cabina mediante el uso de un microcontrolador ESP 32 y un sensor de presión atmosférica con el que se determina la altura de la cabina.
[7]	Bluetooth|	Se implementa una malla Bluetooth para entornos interiores y exteriores, utilizando microcontroladores ESP-32. Se comparan los resultados en pruebas físicas y en simulaciones.
[8]	Bluetooth (BLE)	Sistema para detectar la presencia de alacranes en una granja mediante 5 módulos ESP32 utilizando BLE, 4 esclavos y 1 maestro conectado a un servidor web en la red local.
[9]	ESP-NOW	Elaboración de un prototipo de 6 módulos para el control de luces y la vinculación con una aplicación para teléfono
[10]	Wifi Bluetooth	Sistema de monitoreo, en donde se tiene una etapa de censado de temperatura la cual es procesada por un módulo ESP 32 y enviada a Matlab, los datos también se pueden visualizar desde un smartphone.
[11]	Wifi	Sistema de monitoreo integrado con sensores de temperatura y humedad, incluido con un GPS para obtener información de ubicación. Los datos recogidos quedan subidos a un servidor web privado.

2.2. Visualización de datos a través de una plataforma web

El artículo “Solar Panel Remote Monitoring and Control System on Miniature Weather Stations Based on Web Server and ESP32” [12] presenta un sistema de monitoreo y control remoto de paneles solares en estaciones meteorológicas miniaturas basado en un servidor web y el microcontrolador ESP32. El ESP32 es utilizado para adquirir y enviar datos de los sensores a través de una conexión WiFi a un servidor web alojado en una Raspberry Pi. El sistema también permite el monitoreo y control remoto de los paneles solares mediante la interfaz web proporcionada por el servidor.

El artículo "Sistema de Monitoreo de temperatura y humedad en el hogar aplicando IoT de bajo costo" [13] describe el desarrollo de un sistema de monitoreo de temperatura y humedad en el hogar utilizando IoT de bajo costo. El sistema utiliza un microcontrolador ESP8266 y un sensor DHT11 para medir la temperatura y humedad en el hogar, y luego envía los datos a una página web a través de Wi-Fi. Los datos se muestran en la página web Cloud AMQP y también se pueden enviar alertas por correo electrónico si los niveles de temperatura o humedad se encuentran fuera del rango establecido. El sistema es fácil de usar y puede ser implementado en cualquier hogar para monitorear las condiciones ambientales.

El artículo “Parking System Optimization Based on IoT using Face and Vehicle Plat Recognition vía Amazon Web Service and ESP-32 CAM” [14] describe un sistema de estacionamiento inteligente basado en IoT que utiliza reconocimiento facial y de matrículas de vehículos. El sistema utiliza un módulo ESP-32 CAM como dispositivo de adquisición de imágenes y envía los datos recopilados a través de la plataforma en la nube de Amazon Web Services (AWS) para su procesamiento. La interfaz de usuario del sistema se presenta a través de una página web, lo que permite a los usuarios monitorear el estado del estacionamiento en tiempo real y reservar un lugar de estacionamiento de manera remota.

En el artículo “IoT Based Real Time Warehouse Monitoring using Sparkfun ESP8266 Thing Dev and Cayenne” [15] se describe un sistema de monitoreo de almacenes en tiempo real basado en IoT (Internet de las cosas) utilizando Sparkfun ESP8266 Thing Dev y Cayenne. Se utilizó un sensor DHT11 para medir la temperatura y la humedad en el almacén, y un sensor PIR (sensor infrarrojo pasivo) para detectar la presencia de personas en el almacén. El ESP8266 se programó para enviar los datos del sensor a la plataforma Cayenne en la nube, donde se puede acceder a ellos y monitorear el almacén en tiempo real a través de una página web. La plataforma Cayenne también permite configurar alertas para enviar notificaciones en caso de que se produzcan condiciones anormales. El sistema propuesto tiene el potencial de mejorar la eficiencia en la gestión de almacenes y garantizar la seguridad de los productos almacenados.

El artículo “Invernadero inteligente basado en ESP-MESH y AWS Smart greenhouse using ESP-MESH and AWS” [16] describe un proyecto de invernadero inteligente utilizando la tecnología ESP-MESH y AWS. El autor utiliza el microcontrolador ESP32 para la recopilación de datos ambientales, como temperatura, humedad, luz y humedad del suelo. Estos datos son enviados a través de una red mesh a un gateway ESP32, el cual se encarga de enviarlos a la nube AWS para su procesamiento y visualización en una interfaz web. La

implementación de la red mesh permite la comunicación entre los diferentes nodos del invernadero, permitiendo una mayor cobertura de la red. El trabajo muestra la utilidad del ESP32 para el monitoreo y control de invernaderos inteligentes.

En el proyecto “Diseño y desarrollo de un prototipo inalámbrico para un sistema de casilleros universitarios utilizando dispositivos iot y ubidots” [17] se desarrolla un prototipo inalámbrico, para un sistema de Casilleros Universitarios utilizando dispositivos RFID, IoT, módulos esp32 y la plataforma Ubidots; con el fin de establecer un procedimiento que permita la apertura del casillero a través de un lector RFID y una tarjeta de acceso estudiantil; la información generada por el prototipo permite el registro y análisis de los datos en la plataforma Web de Ubidots, además el sistema permite enviar un correo electrónico en tiempo real al estudiante en caso de ser violentado el casillero. Se realizan envíos de alertas, acceso y registro en la Plataforma Ubidots.

En el trabajo de titulación “Diseño e implementación de un prototipo para un sistema de medición, análisis y purificador de gases contaminantes en el aire utilizando arduino y ubidots ot" [18], se diseñó e implementó un prototipo para medición, análisis y purificación de gases contaminantes en el aire utilizando Arduino y Ubidots. El dispositivo está basado en un ESP8266 NodeMCU que se comunica con un sensor de gas MQ-135 para medir la calidad del aire y transmite los datos a través de Wi-Fi a Ubidots, una plataforma en la nube para el análisis de datos. El sistema también incluye un purificador de aire controlado por el NodeMCU y activado automáticamente según los valores medidos por el sensor.

En el artículo “Diseño de un dispositivo wearable para el monitoreo de la oxigenación y ritmo cardiaco” [19] se elabora un dispositivo en forma de muñequera, el cual monitorea de forma continua remotamente los signos vitales de pacientes afectados por la pandemia de coronavirus (SARS-CoV-2). El diseño está constituido por un sensor que registra el ritmo cardiaco y porcentaje de saturación de oxígeno en sangre. Este sensor es controlado mediante un microcontrolador ESP32, configurado bajo la plataforma de Arduino IDE, el cual por medio de su módulo Wi-Fi permite una transferencia de datos hacia la plataforma de Cayenne.

El artículo "Development of air temperature and soil moisture monitoring systems with LoRA technology"[20] describe el desarrollo de un sistema de monitoreo de temperatura del aire y humedad del suelo utilizando la tecnología LoRa. Los autores utilizan una plataforma basada en ESP32 para conectar los sensores de temperatura y humedad del suelo al módulo LoRa, y así transmitir los datos a un servidor en la nube. El servidor recibe los datos y los muestra en una página web utilizando la plataforma Cayenne. El sistema fue probado en un ambiente de cultivo de plantas y los resultados mostraron una alta precisión y estabilidad en la medición de la temperatura del aire y la humedad del suelo. En general, el artículo demuestra cómo la tecnología LoRa puede ser utilizada en aplicaciones de monitoreo ambiental, y destaca la importancia del uso de plataformas IoT para la recolección y análisis de datos.

En el artículo [21], A. Álvarez Carulla describe cómo comunicar un módulo ESP32 con la plataforma Ubidots mediante MQTT. El ESP32 es un microcontrolador de bajo costo y bajo consumo de energía que se utiliza comúnmente en proyectos de IoT. La plataforma Ubidots es una plataforma en la nube que permite a los usuarios almacenar, visualizar y analizar datos de sensores en tiempo real. El autor proporciona un tutorial paso a paso para conectar el ESP32 a Ubidots a través del protocolo MQTT. MQTT es un protocolo de mensajería ligero que se utiliza comúnmente en aplicaciones de IoT para transmitir datos entre dispositivos. El artículo incluye un código de ejemplo que muestra cómo enviar datos desde el ESP32 a Ubidots utilizando MQTT. El autor también proporciona instrucciones detalladas para configurar una cuenta en Ubidots y crear un dashboard para visualizar los datos.

El trabajo de investigación [22] titulado "Análisis de vulnerabilidades en la seguridad de la red WLAN con IoT en residencias que utilizan domótica y control remoto" se enfocó en analizar la seguridad de la red WLAN en hogares que utilizan dispositivos IoT para la domótica y el control remoto. creando un escenario en tiempo real con el módulo ESP32, de esta manera se conecta a las plataformas Firebase, Ubidots y Cayenne, se menciona que algunos dispositivos IoT utilizados en hogares pueden tener problemas de seguridad, como la falta de actualizaciones de software y el uso de contraseñas predeterminadas débiles, lo que puede poner en riesgo la privacidad y seguridad de los usuarios

Al realizar este proyecto es necesario conocer el lenguaje de programación adecuado. El libro [23] "Electronics Projects with the ESP8266 and ESP32" de N. Cameron, publicado por Apress en 2021, ofrece una guía completa para realizar proyectos con los microcontroladores mencionados. El libro comienza con una introducción a los microcontroladores, sus características y cómo programarlos. Luego, se presentan varios proyectos prácticos, incluyendo el control de dispositivos de iluminación, monitoreo de temperatura y humedad, y la creación de una cámara IP. El libro también incluye información sobre cómo conectar los microcontroladores a internet y cómo utilizar servicios web como IFTTT y AWS IoT.

Por otro lado, [24] "MycroPython ESP32 and ESP8266" de R. Santos y S. Santos se enfoca en cómo utilizar el lenguaje de programación MicroPython para programar los microcontroladores ESP32 y ESP8266. El libro comienza con una introducción al lenguaje MicroPython y cómo instalarlo en los microcontroladores mencionados. Luego, se presentan varios proyectos prácticos, incluyendo el control de dispositivos de iluminación, la creación de una cámara IP y la integración con servicios en la nube como AWS. Además, el libro también incluye información sobre cómo utilizar herramientas de depuración y cómo escribir pruebas unitarias para los proyectos.

Tabla 2. Características empleadas por diversos autores utilizando una plataforma web.

Referencia	Plataforma	Implementación
[12]	Servidor web de área local	Sistema para las estaciones de electricidad para ser monitoreados, controlados remotamente y proporcionar información sobre el uso de energía eléctrica en estaciones meteorológicas en un servidor web. utilizando un módulo ESP32
[13]	Cloud AMQP	Sistema de monitoreo de temperatura y humedad en el hogar. Utilizando el módulo ESP32, un sensor DHT11, y el lenguaje de programación Python, además se utilizó el servidor CloudAMQP con el protocolo MQTT.
[14]	Amazon Web Service	Prototipo de un sistema de estacionamiento, desarrollado para reconocer imágenes faciales y placas de vehículos con 2 cámaras usando la ESP32-Cam. El sistema puede ser asistido por los servicios en la nube de Amazon.
[15]	Cayenne MyDevices Cloud	Sistema de monitoreo de temperatura, humedad y el humo en interiores, además de enviar alertas de notificación vía SMS y e-mail. El método de análisis se realiza utilizando el módulo ESP8266 con Cayenne MyDevices.
[16]	AWS APP Mesh	Diseño de un invernadero utilizando el módulo ESP8266. La información recibida en la nube es procesada y se obtienen los valores de temperatura, humedad y luminosidad. Estos datos son procesados y estudiados en AWS APP Mesh.
[17]	Ubidots	Prototipo inalámbrico para un sistema de casilleros utilizando dispositivos RFID, IoT, módulos esp32 y la plataforma Ubidots.
[18]	Ubidots	Sistema de medición, análisis y purificador de gases contaminantes en el aire. Compuesto por una pantalla y varios sensores para controlar la calidad del aire. Se puede realizar el monitoreo mediante la plataforma Ubidots.
[19]	Cayenne MyDevices Cloud	Dispositivo en forma de muñequera, el cual monitorea el ritmo cardiaco y porcentaje de saturación de oxígeno en sangre de pacientes, mediante un microcontrolador ESP32, se envían los datos hacia la plataforma de Cayenne MyDevices Cloud.
[20]	Cayenne MyDevices Cloud	Sistema de monitoreo de temperatura y humedad, utilizando LoRA y módulos ESP32 que se conecta a un punto de acceso para el envío de datos en línea a través de la aplicación Cayenne. Implementado en campos agrícolas.
[21]	Ubidots	Se utiliza el módulo ESP 32 para hacer una aplicación que permita controlar un LED desde Ubidots donde además se visualizan los datos de la temperatura interna del módulo ESP32.
[22]	Firebase, Ubidots y Cayenne	Estudio y análisis de vulnerabilidades a las plataformas Firebase, Ubidots y Cayenne

Capítulo 3

Marco Teórico

En este capítulo se realiza una introducción a las redes ad hoc, al chip ESP32, a las diferentes formas de crear una red ad hoc con este microcontrolador, los protocolos de comunicación posibles y diversas configuraciones para él envió de datos a una plataforma web, así como una serie de conceptos que serán de utilidad para los siguientes capítulos.

3.1 Redes ad hoc

Existen diferentes tipos de redes según su alcance, topología, modo de acceso y tecnologías utilizadas. A continuación, se presentan algunos de los tipos de redes más comunes:

- Redes de Área Local (LAN): son redes que se utilizan para conectar dispositivos de una misma área geográfica, como un edificio o un campus universitario. Suelen ser redes privadas que permiten compartir recursos como archivos, impresoras, y conexiones a internet. Ejemplos de tecnologías LAN son Ethernet y Wi-Fi.
- Redes de Área Amplia (WAN): son redes que abarcan grandes distancias geográficas, como ciudades, países, e incluso continentes. Se utilizan para conectar dispositivos de diferentes lugares a través de proveedores de servicios de internet. Ejemplos de tecnologías WAN son el Protocolo de Internet (IP), la tecnología de líneas dedicadas, y las redes de área amplia definidas por software (SD-WAN).
- Redes de Área Personal (PAN): son redes que se utilizan para conectar dispositivos personales, como teléfonos móviles, tabletas, computadoras portátiles, y dispositivos inteligentes en un espacio cercano, como una habitación. Ejemplos de tecnologías PAN son Bluetooth y Near Field Communication (NFC).
- Redes de Almacenamiento (SAN): son redes que se utilizan para conectar dispositivos de almacenamiento de datos, como discos duros y unidades de cinta, a servidores y estaciones de trabajo. Suelen ser redes de alta velocidad y baja latencia, y se utilizan en entornos empresariales para el almacenamiento de datos críticos.
- Redes de Campus: son redes que se utilizan para conectar varios edificios dentro de una universidad, empresa, o institución gubernamental. Pueden ser LAN o WAN, y suelen incluir diferentes tecnologías de red para cubrir diferentes áreas y necesidades.
- Redes de Acceso de Radio (RAN): son redes que se utilizan para proporcionar conectividad inalámbrica a dispositivos móviles, como teléfonos móviles y tabletas, en áreas públicas o privadas. Ejemplos de tecnologías RAN son el 3G, 4G, y 5G [25].

Las redes ad hoc son un tipo especial de red que se distingue por su capacidad para operar sin una infraestructura fija o preexistente. En estas redes, los nodos se comunican directamente entre sí, formando una red temporal y autoorganizada. A diferencia de las redes tradicionales, las redes ad hoc son altamente móviles y dinámicas, lo que las hace especialmente útiles en situaciones donde la infraestructura de red no está disponible o es limitada.

En cuanto a las aplicaciones de las redes ad hoc, estas se pueden utilizar en una amplia variedad de escenarios:

- Monitorización ambiental: Proporcionan conectividad en áreas remotas para medir y analizar datos ambientales.
- Agricultura inteligente: Ayudan en la monitorización y gestión de cultivos mediante sensores distribuidos.
- Vigilancia y seguimiento de animales: Permiten rastrear y monitorear la vida silvestre o el ganado en tiempo real.
- Aplicaciones militares y de defensa: Utilizadas para crear redes de comunicación seguras y confiables en el campo de batalla.
- Comunicación de emergencia: Esenciales en situaciones de desastres naturales donde la infraestructura tradicional de comunicación está dañada.
- Telemedicina en zonas rurales: Facilitan la transmisión de datos médicos y permiten consultas a distancia en áreas con acceso limitado a servicios de salud.
- Entretenimiento en eventos masivos: Mejoran la conectividad temporal y soportan aplicaciones de realidad aumentada y realidad virtual.
- Ciudades inteligentes: Optimizan el tráfico, gestionan recursos energéticos y mejoran la seguridad pública mediante la interconexión de sensores y dispositivos IoT (Internet de las Cosas).

Entre las ventajas de las redes ad hoc se destaca su capacidad para proporcionar conectividad en áreas remotas o rurales y en situaciones de emergencia, como desastres naturales o conflictos militares. Además, estas redes pueden ser más económicas y fáciles de desplegar que las redes tradicionales, ya que no requieren una infraestructura fija y costosa.

Sin embargo, las redes ad hoc también presentan desafíos. Su escalabilidad puede ser limitada debido a la naturaleza dinámica de los nodos, y el consumo de energía puede ser un problema, especialmente en dispositivos móviles. La seguridad es otro aspecto crítico, ya que la falta de una infraestructura centralizada puede hacer que estas redes sean más vulnerables a ataques.

La tabla 3 proporciona una visión general de los principales beneficios y limitaciones de las redes ad hoc, destacando tanto su adaptabilidad y flexibilidad como los desafíos que presentan en términos de escalabilidad, consumo de energía y seguridad.

Tabla 3. Ventajas y desventajas de las redes ad hoc

Ventajas	Desventajas
Sin infraestructura: No requieren una infraestructura de red preexistente, lo que facilita su despliegue en entornos sin acceso a redes establecidas.	**Limitada escalabilidad**: A medida que el número de nodos aumenta, la complejidad de la red y los conflictos de comunicación también aumentan, lo que puede reducir la eficiencia.
Autoorganización: Las redes ad hoc son capaces de configurarse y reconfigurarse automáticamente, lo que permite una rápida adaptación a cambios en la topología de la red.	**Consumo de energía**: Los dispositivos en redes ad hoc suelen consumir más energía debido a la necesidad de participar en el encaminamiento y la retransmisión de paquetes.
Flexibilidad y movilidad: Permiten la comunicación entre dispositivos móviles sin necesidad de una infraestructura fija, lo que es ideal para situaciones de emergencia o entornos cambiantes.	**Seguridad limitada:** Las redes ad hoc pueden ser más vulnerables a ataques y brechas de seguridad debido a la falta de una infraestructura centralizada que gestione la seguridad.
Redundancia: Al no depender de un único punto de fallo, las redes ad hoc pueden ser más robustas en términos de resiliencia ante fallos de nodos individuales.	**Ancho de banda reducido:** La capacidad del ancho de banda puede ser menor comparada con redes basadas en infraestructura, especialmente en escenarios con muchos nodos activos.
Costo reducido: La ausencia de infraestructura física reduce significativamente los costos de implementación y mantenimiento.	**Latencia variable:** La latencia puede ser impredecible y variar dependiendo de la carga de la red y la topología cambiante.

3.2 Microcontrolador ESP32

El ESP32 es un microcontrolador altamente versátil y potente desarrollado por Espressif Systems, diseñado especialmente para aplicaciones de IoT (Internet de las cosas). Ofrece una combinación única de características que lo hacen adecuado para una amplia gama de proyectos, desde dispositivos conectados hasta sistemas embebidos complejos.

Una de las características más destacadas del ESP32 es su procesador de doble núcleo Xtensa LX6, que le otorga una potencia de cálculo significativamente mayor en comparación con otros microcontroladores de la misma gama. Esta capacidad de procesamiento dual permite ejecutar múltiples tareas simultáneamente, lo que es fundamental en aplicaciones donde se requiere un procesamiento paralelo, como el procesamiento de datos en tiempo real o la multitarea en sistemas complejos.

Además de su potente procesador, el ESP32 ofrece una amplia conectividad inalámbrica integrada, incluyendo Wi-Fi y Bluetooth. Esto permite una fácil conexión a redes locales e Internet, así como la comunicación con otros dispositivos y sistemas. La versatilidad de estas capacidades inalámbricas lo hace ideal para aplicaciones IoT que requieren comunicación remota y control, como el monitoreo ambiental, el control de dispositivos domésticos inteligentes y la automatización industrial.

Otra ventaja del ESP32 es su generosa cantidad de memoria flash y RAM, lo que proporciona espacio suficiente para almacenar programas y datos, así como ejecutar aplicaciones complejas sin problemas de almacenamiento o rendimiento. Esta capacidad de memoria es esencial en proyectos que requieren el manejo de grandes cantidades de datos o el procesamiento intensivo de datos.

En cuanto a interfaces de comunicación, el ESP32 ofrece una amplia variedad de opciones, incluyendo UART, SPI, I2C y CAN, lo que facilita la interconexión con una variedad de sensores, actuadores y otros dispositivos periféricos. Esto lo hace altamente adaptable a diferentes entornos y aplicaciones.

Ahora, al comparar el ESP32 con otros microcontroladores como el ESP8266, el Raspberry Pi Pico, el Arduino y el STM32, podemos observar algunas diferencias significativas en varias características clave:

Procesador El ESP32 tiene un procesador de doble núcleo, lo que le proporciona una ventaja significativa en comparación con el ESP8266, que tiene un solo núcleo. Esta capacidad de procesamiento adicional permite al ESP32 manejar tareas más complejas y ejecutar múltiples procesos simultáneamente. En contraste, el Raspberry Pi Pico utiliza un microcontrolador RP2040 de doble núcleo basado en ARM Cortex-M0+, que ofrece un rendimiento similar al ESP32 en términos de capacidad de procesamiento paralelo. Por otro lado, las placas Arduino y STM32 generalmente utilizan microcontroladores de un solo núcleo, lo que puede limitar su capacidad para manejar tareas más exigentes en comparación con el ESP32.

Conectividad Inalámbrica: Tanto el ESP32 como el ESP8266 tienen conectividad Wi-Fi y Bluetooth integrada, lo que les otorga una ventaja significativa en comparación con el Raspberry Pi Pico, Arduino y STM32, que generalmente requieren módulos adicionales para agregar estas capacidades. Sin embargo, es importante tener en cuenta que el Raspberry Pi Pico, Arduino y STM32 pueden ofrecer una conectividad inalámbrica a través de módulos externos o shields adicionales, lo que les permite adaptarse a diferentes requisitos de conectividad.

Memoria: El ESP32 generalmente ofrece más memoria flash y RAM que el ESP8266, lo que le permite manejar aplicaciones más complejas y almacenar una mayor cantidad de datos. Similarmente, el Raspberry Pi Pico tiene una cantidad decente de memoria flash y RAM integrada en comparación con las placas Arduino y STM32, lo que le proporciona una ventaja en términos de almacenamiento y capacidad de ejecución de programas. Sin embargo, las placas Arduino y STM32 ofrecen la flexibilidad de expandir la memoria utilizando módulos de memoria externa, lo que puede ser beneficioso en aplicaciones que requieren una gran cantidad de almacenamiento de datos.

Interfaces de Comunicación: El ESP32 ofrece una amplia variedad de interfaces de comunicación, al igual que el ESP8266, lo que facilita la interconexión con una variedad de dispositivos y sensores. El Raspberry Pi Pico también ofrece una variedad de interfaces de comunicación, incluyendo UART, SPI e I2C, lo que lo hace adecuado para una amplia gama de aplicaciones. Por otro lado, las placas Arduino y STM32 son conocidas por su compatibilidad con una amplia variedad de shields y módulos, lo que les permite adaptarse a diferentes requisitos de comunicación.

Opciones de GPIO: Tanto el ESP32 como el ESP8266 ofrecen una cantidad generosa de pines GPIO, lo que permite la conexión de una variedad de dispositivos y periféricos. El Raspberry Pi Pico también ofrece una cantidad decente de pines GPIO, lo que permite una mayor flexibilidad en términos de conectividad y expansión. Las placas Arduino y STM32 son conocidas por su amplia gama de pines GPIO, así como por su compatibilidad con una variedad de shields y módulos adicionales, lo que les permite adaptarse a una amplia gama de aplicaciones.

ADC y PWM: Tanto el ESP32 como el ESP8266 tienen ADC y PWM integrados, lo que permite la lectura de señales analógicas y la generación de señales PWM para controlar dispositivos externos. El Raspberry Pi Pico también tiene ADC y PWM integrados, lo que le permite realizar tareas similares. Las placas Arduino y STM32 ofrecen ADC y PWM integrados, así como la capacidad de expandir estas capacidades utilizando módulos externos.

Alimentación: Tanto el ESP32 como el ESP8266 son eficientes en cuanto al consumo de energía, lo que los hace adecuados para aplicaciones alimentadas por batería. El Raspberry Pi Pico también es eficiente en cuanto al consumo de energía, lo que lo hace adecuado para aplicaciones portátiles y alimentadas por batería. Las placas Arduino y STM32 generalmente consumen más energía en comparación con el ESP32, ESP8266 y Raspberry Pi Pico, lo que puede ser una consideración importante en aplicaciones que requieren una eficiencia energética óptima.

Existen otros microcontroladores similares al ESP32 en el mercado, como el STM32 y el Raspberry Pi. Sin embargo, estos dispositivos suelen ser más costosos y tienen mayores requerimientos de energía, lo que los hace menos adecuados para aplicaciones de IoT con baterías de larga duración. Además, el ESP32 cuenta con un diseño de bajo costo y una amplia gama de periféricos integrados, lo que lo hace una opción atractiva para proyectos de IoT.

En la tabla 3.1 se muestran las principales características de algunos dispositivos similares al ESP32.

Tabla 3.1 Características similares de dispositivos similares al chip esp32.

Característica	ESP32	ESP826	Raspberry Pi Pico	Arduino	STM32
Procesador	Xtensa Dual-Core 32-bit LX6	Tensilica L106 32-bit	Arm Cortex-MO+	AVR	Arm Cortex-M
Velocidad de Reloj	Hasta 240 MHz	Hasta 80 MHz	Hasta 133 MHz	Hasta 20 MHz	Hasta 200 MHz
Memoria Flash	Hasta 4 MB	Hasta 16 MB	2 MB	Hasta 256 KB	Hasta 2 MB
Memoria RAM	Hasta 520 KB	Hasta 520 KB	264 KB	Hasta 8 KB	Hasta 640 KB
Conectividad inalámbrica	Wi-Fi, Bluetooth, BLE	Wi-Fi	Requiere módulos externos	Requiere módulos externos	Requiere módulos externos
Interfaces de comunicación	SPI, I2C, I2S, UART, CAN, Ethernet, SD/MMC	SPI, I2C, I2S, UART	SPI, I2C, UART, ADC	SPI, I2C, UART	SPI, I2C, UART, CAN, USB, Ethernet
GPIO	Hasta 34	Hasta 17	26	Hasta 20	Hasta 169
ADC	Hasta 18 canales de 12 bits	Hasta 1 canal de 10 bits	3 canales de 12 bits	Hasta 8 canales de 10 bits	Hasta 20 canales de 12 bits
PWM	Hasta 16	Hasta 8	3	Hasta 6	Hasta 16
Alimentación	2.2-3.6V	2.5-3.6V	5V	5V o 3.3V	2.0-3.6V
Precio	Moderado	Bajo	Bajo	Bajo	Alto

Nota: Los datos presentados en esta tabla se han recopilado de varios datasheet de diferentes fuentes [26] - [30].

El uso del ESP32 tiene varias ventajas, entre las que se incluyen su bajo costo, su conectividad inalámbrica integrada, su amplia gama de periféricos integrados y su amplio soporte de software. Además, el ESP32 es altamente adaptable a diferentes aplicaciones de IoT, lo que lo hace una opción ideal para proyectos de prototipado rápido y desarrollo de productos a gran escala:

- ❖ Bajo costo: El ESP32 es una solución económica, lo que lo hace accesible para una amplia variedad de proyectos, desde prototipos hasta productos finales.
- ❖ Conectividad inalámbrica integrada: Incluye Wi-Fi y Bluetooth, lo que permite una fácil integración en redes inalámbricas y comunicación entre dispositivos.
- ❖ Amplia gama de periféricos integrados: El ESP32 viene con múltiples periféricos, como ADCs, DACs, interfaces UART, SPI, I2C, PWM, y más, lo que reduce la necesidad de componentes adicionales.
- ❖ Soporte de software extenso: Cuenta con un robusto ecosistema de desarrollo, incluyendo el SDK de Espressif, el soporte para el entorno de desarrollo Arduino y otros frameworks como MicroPython, facilitando el desarrollo de aplicaciones.
- ❖ Adaptabilidad a diversas aplicaciones de IoT: Su diseño versátil permite su uso en una variedad de contextos, desde dispositivos de hogar inteligente hasta soluciones industriales.

Además de estas ventajas, el ESP32 también se destaca por su capacidad de operación en modos de bajo consumo de energía, lo que es crucial para dispositivos IoT que necesitan funcionar durante largos períodos con baterías limitadas. Su capacidad para manejar múltiples tareas y conexiones simultáneas lo convierte en una solución potente para aplicaciones complejas.

En resumen, el chip ESP32 es una opción popular para aplicaciones de IoT debido a sus características de bajo costo, bajo consumo de energía, conectividad inalámbrica integrada, amplia gama de periféricos integrados y amplio soporte de software. Si bien existen otros microcontroladores similares en el mercado, el ESP32 sigue siendo una opción atractiva para proyectos de IoT debido a su diseño adaptable y su amplia comunidad de desarrolladores:

- Bajo consumo de energía: Crucial para dispositivos IoT que requieren una larga duración de la batería.
- Conectividad inalámbrica integrada: Facilita la comunicación entre dispositivos en una red IoT.
- Soporte de software: El extenso soporte y comunidad activa simplifican el desarrollo y la resolución de problemas.
- Versatilidad: Su capacidad para manejar múltiples tareas simultáneas y conexiones lo hace ideal para aplicaciones complejas.
- Comunidad de desarrolladores: Una comunidad grande y activa proporciona recursos, bibliotecas y soporte continuo, lo que agiliza el desarrollo y la implementación de proyectos.

Estas características hacen del ESP32 una solución preferida para una amplia gama de aplicaciones de IoT, desde simples sensores conectados hasta sistemas complejos de automatización y control.

3.3. Creación de redes ad hoc con el chip ESP32

Las redes ad hoc son una tecnología de comunicación inalámbrica que permite la conexión entre dispositivos sin necesidad de una infraestructura de red preexistente. Los dispositivos en una red ad hoc son capaces de comunicarse directamente entre sí, sin la necesidad de un punto de acceso o enrutador central. Los dispositivos se comunican entre sí y establecen su propia red temporal, también conocida como red ad hoc. Una de las aplicaciones más comunes para las redes ad hoc es en entornos donde no hay infraestructura de red existente, como en áreas rurales o remotas, y donde la creación de una infraestructura de red permanente puede ser costosa o poco práctica.

ESP32 es un microcontrolador de bajo costo y bajo consumo de energía que tiene una capacidad inalámbrica incorporada, lo que lo hace ideal para la creación de redes ad hoc. Hay varias formas de crear una red ad hoc con ESP32, cada una con sus propias ventajas y desventajas

- Wi-Fi Direct

  Esta opción permite a los dispositivos ESP32 conectarse entre sí sin necesidad de un punto de acceso intermedio. Los dispositivos pueden comunicarse directamente entre ellos utilizando el protocolo Wi-Fi Direct. La ventaja de esta opción es que no se requiere hardware adicional y es fácil de implementar. La desventaja es que la conexión puede ser menos estable y la distancia de transmisión puede ser limitada.

- Bluetooth

  Los dispositivos ESP32 pueden comunicarse entre sí a través de Bluetooth. Esta opción es fácil de implementar y no requiere hardware adicional. Sin embargo, la distancia de transmisión es limitada y la velocidad de transferencia de datos puede ser más lenta que otras opciones.

- Bluetooth Low Energy (BLE)

  Es una variante de Bluetooth para el ahorro de energía. Su principal aplicación es la transmisión a corta distancia de pequeñas cantidades de datos (bajo ancho de banda). BLE permanece en modo de suspensión constantemente excepto cuando se inicia una conexión a diferencia de Bluetooth que siempre está encendido. Esto hace que consuma muy poca energía, dependiendo el caso de uso, el consumo de energía puede llegar a ser 100 veces menor que Bluetooth.

  Además, BLE admite no solo la comunicación punto a punto, sino también el modo de transmisión y la red de malla, con Bluetooth Low Energy, hay dos tipos de dispositivos: el servidor y el cliente. El ESP32 puede actuar como cliente o como servidor.

Como se mencionó anteriormente, BLE también admite el modo de transmisión y la red de malla:

- Modo de emisión: el servidor transmite datos a muchos clientes que están conectados.
- Red de malla: todos los dispositivos están conectados, esta es una conexión de muchos a muchos.

El servidor se mantiene en espera de ser encontrado por otros dispositivos, y contiene los datos que el cliente puede leer. Por otra parte, el cliente escanea los dispositivos cercanos, y cuando encuentra el servidor que está buscando, establece una conexión y recibe los datos entrantes, a esto se le conoce como comunicación punto a punto.

- ESP NOW

Es un protocolo que permite a múltiples dispositivos lograr mantener una comunicación entre sí sin usar Wi-Fi, esto significa que después de emparejar un dispositivo a otro, la conexión es persistente. Es decir, si de repente una de sus placas pierde energía, se reinicia o por alguna otra razón se logra desconectar, cuando se reinicia, se conectará automáticamente para continuar la comunicación. Mediante este protocolo se puede tener comunicación unidireccional o bidireccional en diferentes configuraciones.

Además, ESP-NOW utiliza menos energía que otras opciones, lo que puede prolongar la vida de la batería en dispositivos alimentados por batería. La desventaja es que se requiere hardware adicional en los dispositivos para utilizar ESP-NOW.

- ESP MESH

Esta opción utiliza el protocolo ESP-NOW de Espressif. Es un protocolo de red construido sobre el protocolo Wi-Fi. ESP-MESH permite que numerosos dispositivos denominados nodos, distribuidos en una gran área se interconecten bajo una sola WLAN (Wireless Local Area Network). ESP-MESH se autoorganiza y se repara automáticamente, lo que significa que la red se puede construir y mantener de forma autónoma.

Los nodos de la red no necesitan conectarse a un nodo central, ya que todos los nodos son responsables de transmitir información entre sí. Esto permite que múltiples dispositivos se comuniquen dentro de una gran área de interés. Los nodos pueden autoorganizarse y comunicarse dinámicamente entre sí para garantizar que la información requerida llegue a su destino final del nodo. Si se elimina cualquier nodo de la red, puede autoorganizarse para asegurarse de que los paquetes lleguen a su destino.

La ventaja de esta opción es que permite una gran distancia de transmisión y puede cubrir áreas más grandes que otras opciones. Además, la red de malla permite una mayor estabilidad y redundancia en la comunicación. La desventaja es que se requiere hardware adicional en los dispositivos para utilizar ESP-MESH. Además, la implementación puede ser más compleja que otras opciones.

- LoRa

Los módulos ESP32 pueden conectarse a través de la tecnología de comunicación LoRa, que utiliza radiofrecuencias de baja potencia para la transmisión de datos de larga distancia. Esta opción es adecuada para aplicaciones en las que se necesita una comunicación a larga distancia, como monitoreo de sensores remotos o sistemas de vigilancia. Sin embargo, el hardware adicional necesario para esta opción puede ser costoso y la velocidad de transmisión puede ser lenta.

La Tabla 3.2 resume los datos recopilados de múltiples fuentes [1]-[11] sobre las principales características de las diferentes maneras de crear una red ad hoc utilizando el chip ESP32. Por otra parte, en la tabla 3.3. se muestran las ventajas y desventajas que se presentan utilizando los diferentes métodos.

Tabla 3.2. Diferentes maneras de crear una red ad hoc utilizando el chip ESP32

Protocolo	**Rango de distancia**	**Consumo de energía**	**Ancho de banda**	**Capacidad de nodos**	**Seguridad**
Wi-Fi Direct	200 metros	Alto	Alta	Hasta 8 nodos	Alto
Bluetooth	10 metros	Bajo	Baja	Hasta 4 nodos	Medio
BLE	100 metros	Bajo	Baja	Hasta 5 nodos	Alto
ESP-MESH	1 km	Bajo	Baja	+ de 5	Medio
ESP NOW	1 km	Bajo	Baja	Hasta 20 nodos	Alto
LoRa	Hasta 15 km	Bajo	Baja	+ de 3	Alto

Tabla 3.3. Ventajas y desventajas con diferentes formas de hacer una red ad hoc utilizando esp32

Método de Comunicación	Ventajas	Desventajas
Wi-Fi Direct	-Alta velocidad de transferencia de datos - Conexión directa entre dispositivos sin necesidad de un punto de acceso - Compatibilidad con otros dispositivos Wi-Fi -Soporte de la mayoría de sistemas operativos y plataformas	- Consumo elevado de energía - Alcance limitado - Requiere de una configuración previa y conexión a Internet para el establecimiento de la red
Bluetooth	- Compatibilidad con la mayoría de dispositivos móviles y otros dispositivos Bluetooth - Bajo costo -Alcance de hasta 10 metros –Alta velocidad de transferencia de datos	- Limitación en la cantidad de dispositivos conectados - Problemas de interferencia con otros dispositivos Bluetooth -Configuración y emparejamiento previos necesarios
BLE	- Consumo bajo de energía - Alta velocidad de transferencia de datos - Conexión fácil y rápida - Compatibilidad con la mayoría de dispositivos móviles - Bajo costo	- Alcance limitado - Limitaciones en la cantidad de dispositivos conectados - Problemas de interferencia con otros dispositivos Bluetooth
ESP-MESH	- Permite la conexión de múltiples dispositivos -Cobertura amplia y escalabilidad - Conexión automática y autónoma de los nodos - Red autoorganizada y auto configurable	- Requiere de configuración previa - Alcance limitado - No es compatible con otros dispositivos o protocolos
ESP NOW	- Alta velocidad de transferencia de datos -Consumo bajo de energía - Bajo costo - Conexión directa y punto a punto entre dispositivos - Permite la conexión de múltiples dispositivos	- Alcance limitado - Requiere de configuración previa - No es compatible con otros dispositivos o protocolos
LoRa	- Alcance muy amplio (hasta varios kilómetros) - Consumo bajo de energía - Bajo costo - Permite la conexión de múltiples dispositivos - Soporta un gran volumen de datos	- Baja velocidad de transferencia de datos - Interferencia en entornos urbanos - Requiere de antenas específicas para su uso

Cada uno de estos protocolos tiene sus ventajas y desventajas. Wi-Fi Direct tiene un rango de distancia mayor que los otros protocolos, pero consume mucha energía. Bluetooth y BLE tienen un bajo consumo de energía, pero un ancho de banda limitado y un número limitado de nodos. ESP-MESH tiene un rango de distancia de hasta 5 km, pero un ancho de banda limitado y un consumo de energía moderado. ESP NOW tiene un bajo consumo de energía y un rango de distancia de hasta 1 km, pero un número limitado de nodos. LoRa tiene un rango de distancia extremadamente largo, pero un ancho de banda muy bajo. En cuanto a la seguridad, tanto Wi-Fi Direct como ESP NOW tienen un alto nivel de seguridad debido al uso de cifrado de datos. BLE también tiene un alto nivel de seguridad, pero es susceptible a ataques de suplantación de identidad. ESP-MESH tiene un nivel de seguridad medio debido a su uso de claves de encriptación compartidas. LoRa también tiene un alto nivel de seguridad, pero es susceptible a ataques de interferencia.

El uso de ESP Mesh para establecer una red ad hoc presenta numerosas ventajas sobre otras tecnologías como Wi-Fi Direct, Bluetooth, BLE, ESP NOW y LoRa en ciertos escenarios específicos. ESP Mesh se destaca por su capacidad para crear una red de malla autoconfigurable y autoorganizable utilizando dispositivos ESP32. Esta red de malla permite que los dispositivos se comuniquen entre sí de manera eficiente, incluso en entornos donde no hay una conexión directa con un enrutador Wi-Fi central. Algunas de las razones clave para elegir ESP Mesh sobre otras tecnologías son:

- Escalabilidad: ESP Mesh permite la expansión de la red de forma dinámica y autónoma. Los nuevos nodos pueden unirse a la red de manera sencilla, y la red se reconfigura automáticamente para optimizar la comunicación entre los dispositivos existentes.
- Robustez: La topología de malla de ESP Mesh proporciona redundancia y tolerancia a fallos. Si un nodo falla o se desconecta, la red puede reenrutarse automáticamente para mantener la conectividad.
- Bajo consumo de energía: ESP Mesh está optimizado para dispositivos de bajo consumo de energía, lo que lo hace adecuado para aplicaciones alimentadas por batería. Los dispositivos pueden entrar en modo de bajo consumo cuando no están en uso, prolongando la duración de la batería.
- Latencia baja: La comunicación en una red de malla ESP Mesh tiende a tener una latencia baja, lo que la hace adecuada para aplicaciones que requieren respuestas rápidas, como el control remoto y la automatización del hogar.

En comparación, Wi-Fi Direct puede ser más adecuado para aplicaciones que requieren una conexión punto a punto entre dos dispositivos, pero puede ser menos escalable y más difícil de configurar para redes más grandes. Bluetooth y BLE son excelentes para comunicaciones de corto alcance, pero pueden no ser ideales para redes de malla de largo alcance. ESP NOW es una opción viable para la comunicación directa entre dispositivos ESP32, pero carece de la capacidad de autoorganización y re enrutamiento ofrecida por ESP Mesh. LoRa, por otro lado, es excelente para comunicaciones de largo alcance, pero puede tener limitaciones en términos de velocidad y escalabilidad en comparación con ESP Mesh para aplicaciones de red ad hoc.

3.4. Protocolos de comunicación entre ESP32 y una página web

Existen varios protocolos que se pueden utilizar para enviar datos desde un ESP32 a una página web, cada uno con sus propias características y ventajas. Algunos de los protocolos más comunes son los siguientes:

- HTTP (Hypertext Transfer Protocol)

  Este protocolo de aplicación se utiliza para la transferencia de datos en la World Wide Web. El ESP32 puede enviar datos a un servidor web a través de solicitudes HTTP POST o GET. La mayoría de los servicios en la nube, como AWS, Google Cloud y Azure, admiten HTTP, lo que lo hace una opción popular para la comunicación de datos en la web.

- MQTT (Message Queuing Telemetry Transport):

  Este es un protocolo de mensajería ligero diseñado para la comunicación de máquina a máquina en la Internet de las cosas (IoT). El ESP32 puede enviar datos a través del protocolo MQTT a un broker MQTT, que puede estar alojado localmente o en la nube. Los datos enviados pueden ser suscritos y consumidos por otros dispositivos o aplicaciones, lo que lo hace ideal para escenarios de IoT donde se requiere una comunicación eficiente y escalable.

- WebSocket:

  Este protocolo permite la comunicación bidireccional en tiempo real entre un servidor y un cliente a través de una conexión TCP. El ESP32 puede enviar datos a través de WebSocket a un servidor WebSocket que puede estar alojado localmente o en la nube. WebSocket es útil en aplicaciones que requieren una comunicación en tiempo real, como aplicaciones de chat o tableros de instrumentos en tiempo real.

- CoAP (Constrained Application Protocol):

  Este es un protocolo de aplicación diseñado específicamente para su uso en redes de sensores inalámbricos y dispositivos IoT. El ESP32 puede enviar datos a través del protocolo CoAP a un servidor CoAP, que puede estar alojado localmente o en la nube. CoAP es eficiente en términos de ancho de banda y recursos, lo que lo hace ideal para dispositivos con recursos limitados, como sensores IoT.

Cada uno de estos protocolos tiene sus propias ventajas y desventajas, y la elección del protocolo adecuado dependerá de los requisitos específicos del proyecto y las características de la aplicación. En la tabla 3.4 se muestra la información recopilada de diversas fuentes [12]-[22] entre las principales ventajas y desventajas entre los diferentes protocolos de comunicación que se pueden utilizar para establecer una comunicación bidireccional entre un chip ESP32 y una página web.

Tabla 3.4 Ventajas y desventajas de los protocolos utilizados para establecer comunicación bidireccional entre un chip ESP32 y una página web.

Método	Ventajas	Desventajas
HTTP	- Ampliamente utilizado y compatible con la mayoría de los navegadores - Fácil implementación - Permite enviar y recibir datos en diferentes formatos (JSON, XML, HTML, etc.).	- No es adecuado para aplicaciones que requieren una comunicación en tiempo real. - No es seguro para transmitir datos confidenciales, ya que la información se envía en texto plano.
MQTT	- Ideal para aplicaciones de Internet de las cosas (IoT). - Muy eficiente en términos de consumo de ancho de banda y energía. - Permite la comunicación bidireccional. -Soporta un gran número de clientes.	- Requiere un broker MQTT para mediar entre los clientes y los suscriptores.
Web Sockets	- Permite la comunicación bidireccional en tiempo real. - Compatible con la mayoría de los navegadores modernos. - Fácil implementación. - Eficiente en términos de ancho de banda y recursos.	- No es adecuado para aplicaciones que requieren una gran cantidad de transferencia de datos. - Requiere un servidor WebSockets.
CoAP	- Es un protocolo de bajo consumo de energía y ancho de banda, ideal para dispositivos IoT. -Soporta la comunicación multicast y el descubrimiento de recursos. - Soporta la seguridad y la autenticación. -Fácil implementación.	- No es ampliamente utilizado y puede ser difícil de encontrar - No es adecuado para aplicaciones que requieren una gran cantidad de transferencia de datos. - Requiere un servidor CoAP.

En el caso de la red ad hoc mencionada previamente, el protocolo MQTT fue seleccionado debido a su ligereza, eficiencia y capacidad para manejar la comunicación de máquina a máquina en entornos de IoT. El protocolo MQTT es altamente escalable y eficiente en términos de ancho de banda, lo que lo hace ideal para entornos donde se necesitan múltiples nodos de sensores para enviar datos a un servidor central. Además, MQTT es compatible con la mayoría de los brokers MQTT, lo que permite una fácil integración con plataformas en la nube y sistemas de gestión de datos.

En resumen, el protocolo MQTT fue seleccionado para la red ad hoc debido a su ligereza, eficiencia y capacidad para manejar la comunicación de máquina a máquina en entornos de IoT, lo que lo hace ideal para el monitoreo de temperatura y humedad en el proyecto mencionado.

3.4.1. Protocolo MQTT: conexión web-modulo ESP32

La comunicación entre dispositivos en diferentes redes Wi-Fi requiere una solución diferente a la comunicación entre dispositivos dentro de una misma red Wi-Fi. El protocolo MQTT (Message Queuing Telemetry Transport) permite que la comunicación entre dispositivos y un intermediario pase información entre éste y un dispositivo y entre éste y un segundo dispositivo, con los dos dispositivos en diferentes redes Wi-Fi, el MQTT permite la transferencia de datos entre dispositivos sin violar el firewall salvaguardas. Cuando un dispositivo en una red Wi-Fi solicita información de un segundo dispositivo en otra red, la información se permite a través del firewall de la red, ya que la solicitud provino de la red Wi-Fi [12]-[22].

- Firebase

Es una plataforma de Google que proporciona una base de datos en tiempo real para aplicaciones móviles y web. Una de las ventajas de Firebase es que es fácil de integrar con el ESP32, ya que existen bibliotecas específicas para su uso. Además, Firebase ofrece una interfaz intuitiva para visualizar los datos y una gran cantidad de herramientas para analizarlos. Sin embargo, Firebase es una plataforma de pago, por lo que no es una opción viable para proyectos de bajo presupuesto.

- Ubidots

Es otra plataforma popular para la gestión de datos en la nube. Al igual que Firebase, Ubidots proporciona bibliotecas específicas para ESP32 y una interfaz intuitiva para visualizar los datos. Además, Ubidots ofrece planes gratuitos y de pago, lo que lo hace una opción asequible para proyectos de diferentes presupuestos. Sin embargo, la cantidad de datos que se pueden enviar en un plan gratuito es limitada y el precio de los planes de pago puede ser costoso para proyectos más grandes.

- Amazon Web Service (AWS)

Es una plataforma de nube muy popular que ofrece una amplia gama de servicios de computación en la nube. AWS proporciona una gran cantidad de servicios para IoT, como AWS IoT Core, que permite la gestión y visualización de datos de dispositivos IoT. AWS IoT Core es fácil de integrar con ESP32 y ofrece una gran cantidad de herramientas para el procesamiento y análisis de datos. Sin embargo, AWS puede ser costoso para proyectos de baja escala y su uso requiere de cierto conocimiento técnico.

- AWS App Mesh

Es una plataforma que permite la gestión y visualización de datos de dispositivos IoT a través de una aplicación móvil. AWS App Mesh es fácil de integrar con ESP32 y proporciona herramientas para el procesamiento y análisis de datos. Además, AWS App Mesh es fácil de usar y tiene una interfaz intuitiva para visualizar los datos. Sin embargo, al igual que AWS, su uso puede ser costoso para proyectos de baja escala.

- Cloud AMQP

  Es una plataforma de mensajería que permite la transferencia de datos entre dispositivos IoT. Cloud AMQP es fácil de integrar con ESP32 y proporciona herramientas para el procesamiento y análisis de datos. Además, Cloud AMQP es fácil de usar y tiene una interfaz intuitiva para visualizar los datos. Sin embargo, Cloud AMQP no proporciona una base de datos para almacenar los datos enviados, lo que puede ser un problema para algunos proyectos.

- MQTTX

  Es una herramienta de código abierto diseñada para facilitar el desarrollo y la depuración de aplicaciones que utilizan el protocolo MQTT. Proporciona una interfaz gráfica de usuario intuitiva que permite a los desarrolladores visualizar y gestionar la comunicación MQTT de forma sencilla y eficaz. Algunas de las características principales de MQTTx incluyen: Monitoreo y visualización de mensajes, gestión de clientes MQTT y Soporte para SSL/TLS

La Tabla 3.5 resume los datos recopilados de múltiples fuentes [12]-[22], entre las principales ventajas y desventajas entre las plataformas que se pueden utilizar para establecer comunicación con el chip esp32 utilizando el protocolo MQTT

En este contexto, MQTTX se destaca como una opción atractiva para la gestión y visualización de datos MQTT generados por dispositivos IoT como el ESP32. MQTTX es una herramienta de código abierto diseñada para facilitar el desarrollo y la depuración de aplicaciones que utilizan el protocolo MQTT. Proporciona una interfaz gráfica de usuario intuitiva que permite a los desarrolladores visualizar y gestionar la comunicación MQTT de manera sencilla y eficaz.

Algunas de las características principales de MQTTX incluyen:

- Monitoreo y visualización de mensajes: Permite ver los mensajes MQTT en tiempo real, lo que facilita la depuración y el análisis de la comunicación entre dispositivos.
- Gestión de clientes MQTT: Permite gestionar múltiples clientes MQTT y suscripciones, lo que facilita la administración de dispositivos y la organización de datos.
- Soporte para SSL/TLS: Proporciona seguridad adicional mediante el soporte para SSL/TLS, lo que garantiza que la comunicación entre dispositivos y brokers MQTT sea segura y privada.

En resumen, MQTTX es una opción sólida para la gestión y visualización de datos MQTT generados por dispositivos IoT como el ESP32, gracias a su interfaz intuitiva, sus características avanzadas y su naturaleza de código abierto que permite una mayor flexibilidad y personalización

Tabla 3.5. Ventajas y desventajas entre las plataformas que se pueden utilizar para establecer comunicación con el chip esp32 utilizando el protocolo MQTT

Plataforma	Ventajas	Desventajas
Firebase	• Almacenamiento y sincronización en tiempo real de datos en la nube. • Fácil integración con otras herramientas de Google, como Google Analytics y Google Cloud Storage. • Soporte para múltiples lenguajes de programación y sistemas operativos.	• Costos elevados para grandes volúmenes de datos. • Limitaciones en el tipo de datos que se pueden enviar.
Ubidots	• Interfaz gráfica de usuario fácil de usar para la visualización de datos. • Almacenamiento y análisis de datos en la nube. • Integración con múltiples dispositivos IoT.	• Costos elevados para grandes volúmenes de datos. • Limitaciones en la cantidad de dispositivos y variables que se pueden conectar.
Amazon Web Services	• Escalabilidad y flexibilidad para manejar grandes volúmenes de datos. • Integración con otras herramientas de AWS. • Soporte para múltiples protocolos de comunicación.	• Configuración y uso complicados. • Costos elevados para grandes volúmenes de datos.
AWS App Mesh	• Permite la comunicación entre múltiples servicios de AWS. • Integración con otras herramientas de AWS. • Escalabilidad y flexibilidad para manejar grandes volúmenes de datos.	• Configuración y uso complicados. • Limitaciones en la cantidad de servicios que se pueden conectar.
Cloud AMQP	• Configuración y uso sencillos. • Permite el envío y recepción de grandes volúmenes de datos en tiempo real. • Escalabilidad y flexibilidad para manejar múltiples conexiones.	• Costos elevados para grandes volúmenes de datos. • No admite todos los protocolos de comunicación.
MQTTX	• Interfaz gráfica intuitiva para facilitar la depuración. • Soporte para suscripción y publicación de temas MQTT. • Funcionalidades avanzadas de gestión de clientes MQTT. • Monitoreo y visualización en tiempo real de mensajes MQTT. • Capacidad para conexiones seguras mediante SSL/TLS.	• Puede tener una curva de aprendizaje para usuarios no familiarizados con MQTT. • Es una herramienta específica para MQTT y puede no ser compatible con otros protocolos de comunicación.

3.5. Evaluación de materiales para el desarrollo del proyecto

La selección adecuada de los materiales es esencial para lograr el funcionamiento óptimo del sistema y alcanzar los objetivos deseados. Además, se explicará el proceso de selección de los materiales y las consideraciones clave que se tuvieron en cuenta al elegirlos.

3.5.1 Sensores de temperatura y humedad

Existen varios tipos de sensores de temperatura y humedad compatibles al chip ESP32. A continuación, se presentan algunos de los más comunes:

- DHT11

  Este es uno de los sensores de temperatura y humedad más populares. Es económico y fácil de usar, lo que lo hace una buena opción para proyectos de bricolaje. Puede medir la temperatura y la humedad relativa y su rango de medición es de 0 a 50 grados Celsius para la temperatura y del 20% al 90% para la humedad relativa [31].

- DHT22

  Este sensor es similar al DHT11, pero tiene una mayor precisión y rango de medición. Puede medir temperaturas entre -40 y 80 grados Celsius, y humedad relativa entre 0 y 100%. Es una buena opción para proyectos que requieren mayor precisión [32].

- BME280

  Este sensor puede medir la temperatura, la humedad y la presión atmosférica. Tiene una precisión superior a los sensores DHT y su rango de medición es más amplio. Puede medir temperaturas entre -40 y 85 grados Celsius, humedad relativa de 0 a 100% y presión atmosférica de 300 a 1100 hPa [33].

- DS18B20

  Este sensor es de alta precisión y puede medir la temperatura en un rango de -55 a 125 grados Celsius. Utiliza una interfaz de un solo cable y se puede conectar varios sensores a un solo pin del ESP32 [34].

- AM2302

  Este sensor es similar al DHT22 y también puede medir la temperatura y la humedad relativa. Su rango de medición es de -40 a 80 grados Celsius para la temperatura y del 0 al 100% para la humedad relativa [35].

Si se necesita alta precisión, se pueden elegir sensores como el BME280 o el DS18B20. Si se requiere una solución más económica, se puede optar por el DHT11, DHT22 o el AM2302. En cualquier caso, todos estos sensores son compatibles con el ESP32 y pueden ser fácilmente integrados en proyectos de IoT.

La tabla 3.6 presenta una comparativa de las principales características de los sensores de temperatura y humedad que son compatibles con el chip ESP32

Tabla 3.6. Características de los sensores de temperatura y humedad.

Sensor	Rango de Temperatura	Rango de Humedad	Precisión Temperatura	Precisión Humedad	Interfaz	Precio (USD)
DHT11	0°C a 50°C	20% a 80%	±2°C	±5% RH	Digital	1-2
DHT22	-40°C a 80°C	0% a 100%	±0.5°C	±2-5% RH	Digital	4-6
DS18B20	-55°C a 125°C	N/A	±0.5°C	N/A	Digital	2-3
BME280	-40°C a 85°C	0% a 100%	±1°C	±3% RH	I2C/SPI	5-8
SHT31	-40°C a 125°C	0% a 100%	±0.3°C	±2% RH	I2C	8-12

Nota: Los datos presentados en esta tabla se han recopilado de varios datasheet de diferentes fuentes [31] - [35].

3.5.2 Baterías recurrentes en proyectos similares

Existen diferentes tipos de baterías que pueden ser usadas para alimentar el chip ESP32. Algunas son:

- Baterías de iones de litio (Li-ion):

  Son una opción popular para aplicaciones portátiles ya que tienen una alta densidad de energía y una larga vida útil. Vienen en diferentes tamaños y capacidades, como 18650, 14500, entre otros [36] , [37].

- Baterías de polímero de litio (Li-Po)

  Son similares a las baterías de iones de litio, pero tienen una forma más flexible y se pueden encontrar en tamaños y capacidades similares [38] , [39].

- Baterías de níquel-metal hidruro (NiMH)

  Son una opción más económica en comparación con las baterías de iones de litio y ofrecen una buena relación entre el costo y la capacidad. [40] , [41].

- Baterías de plomo-ácido

  Son una opción económica y confiable para aplicaciones que no requieren una gran movilidad. Tienen una baja densidad de energía en comparación con las baterías de iones de litio y pueden ser pesadas y voluminosas. [42] , [43].

Es importante tener en cuenta que la elección de la batería adecuada dependerá de las necesidades específicas de cada proyecto, como la duración de la batería, el tamaño y la movilidad requerida, entre otros factores. Además, es importante utilizar baterías de buena calidad y seguir las instrucciones de seguridad correspondientes para evitar daños o riesgos de seguridad.

La tabla 3.7 resume los datos recopilados de las fuentes [36]-[43] sobre las características de las baterías de que se pueden conectar al ESP32. Por otra parte, en la tabla 3.8 se muestra una tabla comparativa entre las ventajas y desventajas de las baterías de que se pueden conectar al ESP32

Tabla 3.7. Características de las baterías de que se pueden conectar al ESP32.

Tipo de batería	**Capacidad (mAh)**	**Voltaje (V)**	**Peso (g)**	**Tamaño (mm)**	**Costo ($)**	**Vida útil (ciclos)**
Batería de litio polímero (LiPo)	100-5000	3.7	5-50	Variado	5-50	300-500
Batería de iones de litio (Li-ion)	100-3000	3.7	15-50	Variado	10-50	500-1000
Batería de níquel-metal hidruro (NiMH)	500-4000	1.2	25-50	Variado	5-15	500-1000
Batería de níquel-cadmio (NiCd)	300-1500	1.2	20-50	Variado	5-20	500-1000

Tabla 3.8. Ventajas y desventajas de las baterías que se pueden conectar al ESP32.

Tipo de batería	**Ventajas**	**Desventajas**
Batería de polímero de litio (LiPo)	Alta densidad de energía, bajo peso, baja autodescarga, alta eficiencia	Requiere un cargador específico, posibilidad de inflamación o explosión si se manipula incorrectamente, riesgo de sobrecarga o descarga profunda si no se utiliza un circuito de protección adecuado
Batería de iones de litio (Li-ion)	Alta densidad de energía, baja autodescarga, mayor seguridad que las baterías LiPo	Requiere un cargador específico, posibilidad de inflamación o explosión si se manipula incorrectamente, riesgo de sobrecarga o descarga profunda si no se utiliza un circuito de protección adecuado
Batería de níquel-metal hidruro (NiMH)	Bajo costo, mayor capacidad que las baterías de NiCd, bajo efecto memoria	Tendencia a perder carga rápidamente, menor densidad de energía que las baterías de litio, menor vida útil que las baterías de litio
Batería de níquel-cadmio (NiCd)	Bajo costo, alta capacidad de descarga, mayor vida útil que las baterías de NiMH	Menor densidad de energía que las baterías de litio, posibilidad de contaminación por cadmio, menor vida útil que las baterías de litio

Capítulo 4

Desarrollo

Pasos a seguir para crear un sistema de monitoreo con una red ad hoc utilizando ESP32 con BLE y sensores DHT11:

4.1 Análisis de los materiales utilizados en el desarrollo del proyecto

En este capítulo se presentarán los materiales utilizados en la realización del proyecto. Se describirá detalladamente cada uno de los componentes y dispositivos utilizados en el proyecto, incluyendo sus características técnicas, capacidades y limitaciones. Este capítulo proporcionará una visión general completa de los materiales necesarios para la construcción del sistema

4.1.1 Microcontrolador ESP32-C3-WROOM-02

Es un microcontrolador WiFi de bajo consumo de la familia ESP32, fabricado por la compañía Espressif Systems. Este dispositivo es una versión más reciente del ESP32 y se caracteriza por tener un tamaño más pequeño y un consumo de energía más bajo que sus predecesores, lo que lo hace ideal para aplicaciones de IoT (Internet de las cosas) que requieren dispositivos compactos y con bajo consumo de energía.

Entre las características más importantes del ESP32-C3-WROOM-02 se encuentran:

- CPU y memoria On-Chip

  - ESP32-C3 integrado, Procesador de un solo núcleo RISC-V de 32 bits, hasta 160 MHz.
  - ROM de 384 KB
  - 400 KB de SRAM (16 KB para caché)
  - 8 KB de SRAM en RTC

- Wi-Fi

  - Cumple con IEEE 802.11 b/g/n
  - Rango de frecuencia central del canal operativo: 2412 ~ 2484MHz
  - Admite ancho de banda de 20 MHz y 40 MHz en 2,4 GHz banda
  - Modo 1T1R con velocidad de datos de hasta 150 Mbps
  - Wi-Fi multimedia (WMM)
  - TX/RX A-MPDU, TX/RX A-MSDU
  - ACK de bloqueo inmediato
  - Fragmentación y desfragmentación
  - Oportunidad de transmisión (TXOP)
  - Monitoreo automático de balizas (hardware TSF)
  - 4 × interfaces Wi-Fi virtuales

- Soporte simultáneo para Infraestructura BSS en modo estación, modo SoftAP, Estación + SoftAP modo y modo promiscuo
  Tener en cuenta que cuando la serie ESP32-C3 escanea en Station modo, el canal SoftAP cambiará junto con el canal de la estación
- 802.11mcFTM

➢ Bluetooth

- Bluetooth Low Eenergy: Bluetooth 5, malla Bluetooth
- Velocidad: 125 Kbps, 500 Kbps, 1 Mbps, 2 Mbps
- Extensiones publicitarias
- Múltiples conjuntos de anuncios
- Algoritmo de selección de canal n.º 2
- Mecanismo interno de convivencia entre Wi-Fi y Bluetooth para compartir la misma antena

➢ Periféricos

- GPIO, SPI, UART, I2C, I2S, control remote periférico, Controlador LED PWM, DMA general controlador, controlador TWAI (compatible con ISO 11898-1, es decir, especificación CAN 2.0), USB Controlador serie/JTAG, sensor de temperatura, SAR ADC, temporizadores de uso general, temporizadores de vigilancia

➢ Componentes integrados en el módulo

- Oscilador de cristal de 40 MHz
- Flash SPI

➢ Opciones de antena

- Antena PCB integrada (ESP32-C3-WROOM-02)
- Antena externa mediante conector (ESP32-C3-WROOM-02U)

➢ Condiciones de operación

- Tensión de funcionamiento/Fuente de alimentación: 3,0 ~ 3,6 V
- Temperatura ambiente de funcionamiento:
  – Módulo versión 85: – 40 °C ~ 85 °C
  – Módulo versión 105: – 40 °C ~ 105 °C

➢ Certificación

- Certificación RF: Ver certificados
- Certificación ecológica: RoHS/REACH

El ESP32-C3-WROOM-02 (Figura 2) es compatible con diversas herramientas de desarrollo, incluyendo el entorno de desarrollo integrado (IDE) de Espressif Systems, así como plataformas como Arduino y MicroPython. Además, es posible programar el dispositivo en lenguajes de programación como C y C++.

Figura 2. Microcontrolador ESP32-C3-WROOM-02

Diseño de conexión y descripción funcional de los pinesdel Microcontrolador ESP32-C3-WROOM-02

El siguiente diagrama de pines presentado en la figura 3 proporciona una representación visual de la ubicación aproximada de los pines en el módulo. Este esquema es útil para comprender la disposición general de los pines y facilita la identificación de las conexiones necesarias para conectar dispositivos externos al módulo ESP32

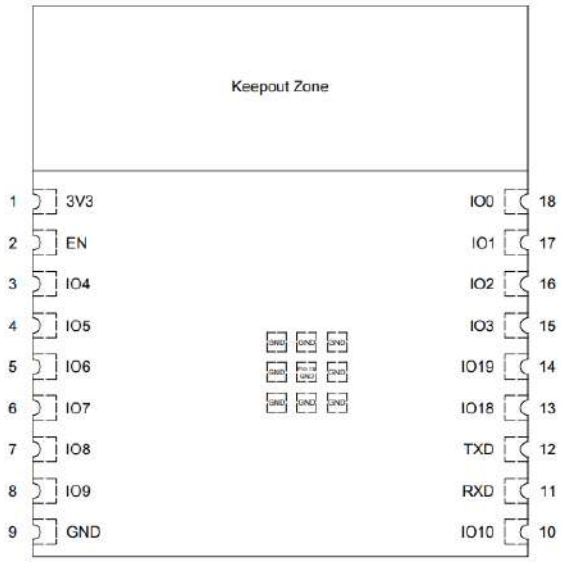


Figura 3. Diseño de pines del Microcontrolador ESP32-C3-WROOM-02 (vista superior)

El módulo tiene 19 pines en total, los cuales desempeñan funciones específicas en el dispositivo. En la tabla 4 se describen en detalle las funciones de cada uno de estos pines, lo que facilita su identificación y su uso adecuado en aplicaciones de diseño electrónico.

Tabla 4 Descripción de los pines del Microcontrolador ESP32-C3-WROOM-02

Nombre	No.	Tipo	Función
3V3	1	P	Fuente de alimentación
EN	2	I	High: encendido, habilita el chip. Low: apagado, el chip se apaga. Nota: No deje el pin EN flotando
IO4	3	I/O/T	GPIO4, ADC1_CH4, FSPIHD, MTMS
IO5	4	I/O/T	GPIO5, ADC2_CH0, FSPIWP, MTDI
IO6	5	I/O/T	GPIO6, FSPICLK, MTCK
IO7	6	I/O/T	GPIO7, FSPID, MTDO
IO8	7	I/O/T	GPIO8
IO9	8	I/O/T	GPIO9
GND	9 , 19	P	Ground
IO10	10	I/O/T	GPIO10, FSPICS0
RXD	11	I/O/T	GPIO20, U0RXD
TXD	12	I/O/T	GPIO21, U0TXD
IO18	13	I/O/T	GPIO18, USB_D
IO19	14	I/O/T	GPIO19, USB_D+
IO3	15	I/O/T	GPIO3, ADC1_CH3
IO2	16	I/O/T	GPIO2, ADC1_CH2, FSPIQ
IO1	17	I/O/T	GPIO1, ADC1_CH1, XTAL_32K_N
IO0	18	I/O/T	GPIO0, ADC1_CH0, XTAL_32K_P

P: power supply; I: input; O: output; T: high impedance.

4.1.2 Sensor DHT11.

Es un sensor de humedad y temperatura de bajo costo, que utiliza un sensor capacitivo de humedad y un termistor para medir la temperatura ambiente. Este sensor es capaz de medir la humedad relativa del aire en un rango de 20% a 90% con una precisión de ±5%, y la temperatura ambiente en un rango de 0°C a 50°C con una precisión de ±2°C.

El sensor DHT11 (Figura 4) se comunica con el microcontrolador a través de un solo pin digital, lo que lo hace fácil de usar y conectar a diferentes plataformas. Además, es un sensor de bajo consumo, por lo que es ideal para aplicaciones de IoT que requieren un consumo de energía reducido.

Se puede utilizar en diferentes proyectos de electrónica y automatización, como control de clima en invernaderos, monitoreo de temperatura y humedad en viviendas, y en sistemas de control de aire acondicionado y calefacción. Es compatible con diferentes plataformas de desarrollo, como Arduino, Raspberry Pi y otros microcontroladores.

Conectar el sensor DHT11 al ESP32-C3-WROOM-02:

- Pin VCC del sensor a 3.3V del ESP32-C3-WROOM-02
- Pin GND del sensor a GND del ESP32-C3-WROOM-02
- Pin DATA del sensor a cualquier pin digital del ESP32-C3-WROOM-02 (por ejemplo, el pin 5)

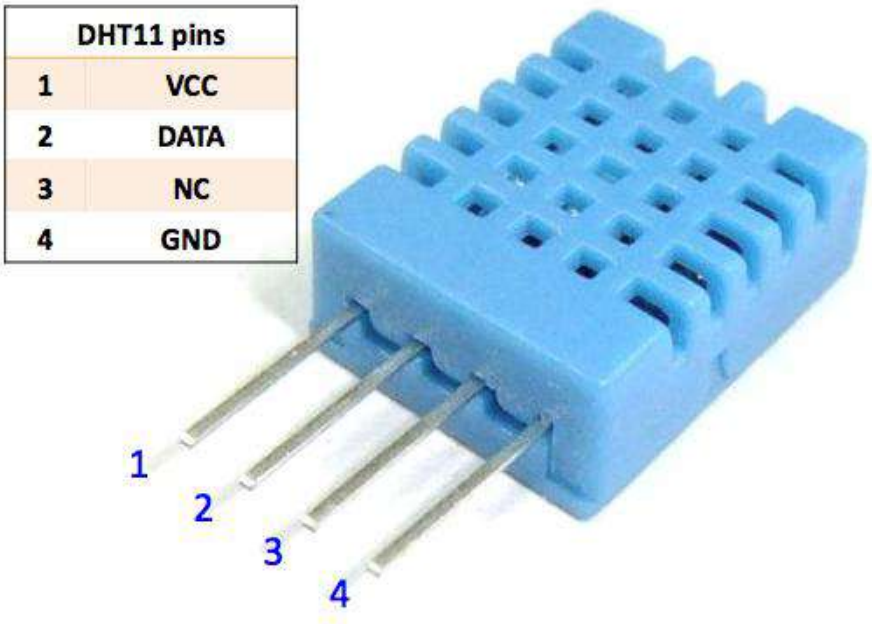


Figura 4. Sensor DHT11.

4.1.3 Batería recargable de polímero de litio

La Batería Recargable LiPo 3.7v 1000mAh, que se muestra en la Figura 4, es una pequeña batería de iones de litio utilizada comúnmente en proyectos electrónicos que requieren una fuente de energía de baja potencia. Esta batería tiene una capacidad de 1000mAh, lo que significa que puede suministrar una corriente de 1000mA durante una hora antes de agotarse por completo.

Las baterías de iones de litio son una elección popular para proyectos electrónicos debido a su alta densidad de energía, su bajo peso y su capacidad de carga rápida. La Batería LiPo es delgada y compacta, lo que la hace ideal para proyectos con limitaciones de espacio. Además, al ser recargable, representa una alternativa más sostenible y rentable a las baterías desechables.

Para cargar la Batería LiPo (Figura 5), se puede utilizar un cargador de batería de iones de litio estándar que se conecta a la batería a través de dos pines. Es importante usar un cargador compatible que evite la sobrecarga o el sobrecalentamiento de la batería. Se recomienda revisar las especificaciones de la batería y el cargador antes de usarlos juntos.

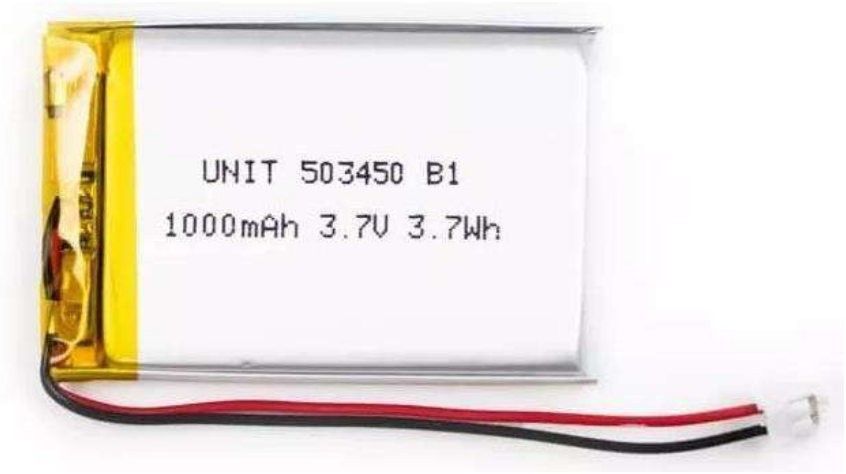


Figura 5. Batería recargable de polímero de litio 3.7v 1000mAh

Para utilizar la batería con el ESP32-C3-WROOM-02 de manera segura y eficiente, es necesario conectarla a través de un circuito de carga y protección que evite la sobrecarga o la descarga excesiva de la batería. Este circuito actúa como una barrera de seguridad para garantizar el funcionamiento óptimo y la vida útil prolongada de la batería. Además, es esencial tener en cuenta que la capacidad limitada de la batería puede influir en la duración de la alimentación del ESP32-C3-WROOM-02, por lo que es importante gestionar adecuadamente el consumo de energía para optimizar su rendimiento.

La Figura 6 proporciona una visualización detallada de cómo se realiza la conexión de la Batería LiPo 3.7V 1000mAh con el ESP32-C3-WROOM-02. A continuación, se detallan los pasos para realizar dicha conexión:

Conexión de la Batería LiPo al circuito de carga y protección:

- Conecte el pin positivo de la batería al pin de entrada de carga del circuito de carga y protección. Esta conexión permite que la batería reciba la corriente necesaria para su carga.

- Conecte el pin negativo de la batería al pin de salida del circuito de carga y protección. Este enlace asegura que la energía fluya correctamente desde la batería hacia el ESP32-C3-WROOM-02, sin riesgo de sobrecarga o descarga excesiva.

Conexión del circuito de carga y protección al ESP32-C3-WROOM-02:

- Conecte el pin de salida del circuito de carga y protección al pin de alimentación del ESP32-C3-WROOM-02. Esta conexión permite que la energía proveniente de la batería alimente directamente al microcontrolador ESP32.

- Conecte el pin de entrada del circuito de carga y protección a una fuente de alimentación externa estándar para baterías de iones de litio. Esta fuente proporciona la energía necesaria para cargar la batería cuando no está en uso o cuando se requiere una carga adicional para mantener su rendimiento óptimo.

A través de estas conexiones cuidadosamente establecidas, se garantiza una alimentación segura y eficiente del ESP32-C3-WROOM-02 mediante la Batería LiPo, lo que permite un funcionamiento fiable y prolongado del sistema en aplicaciones de IoT y otros proyectos electrónicos.

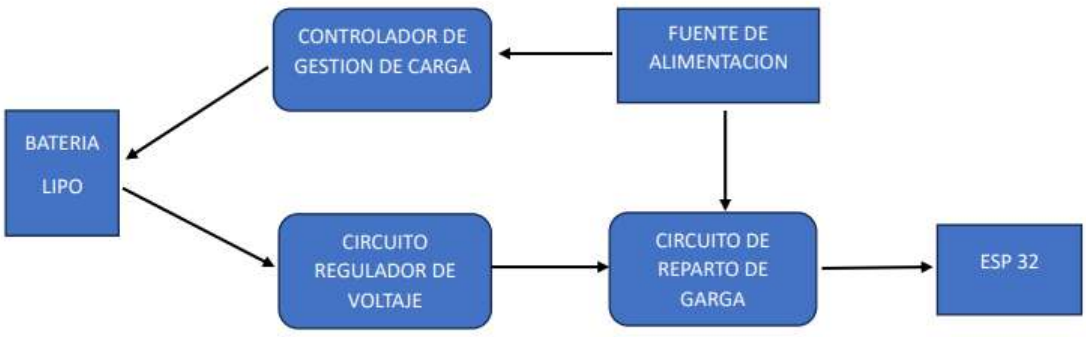


Figura 6. Esquema de conexión de la Batería LiPo con el ESP32-C3-WROOM-02

4.2 Diseño y seccionamiento del circuito eléctrico implementado

Con la ayuda de un software de diseño de circuitos, se debe diseñar el circuito eléctrico para conectar el ESP32, los sensores DHT11 y la batería. Este diseño también debe incluir los componentes necesarios para la comunicación entre los dispositivos.

Eagle es un software ampliamente utilizado en la industria electrónica para el diseño de placas de circuito impreso (PCB, por sus siglas en inglés). Es una herramienta muy completa y potente, que ofrece muchas funcionalidades para diseñar PCBs de alta calidad y precisión. Entre las diversas características que ofrece Eagle, se destacan la facilidad de uso, la compatibilidad con diversos formatos de archivos y la posibilidad de personalizar los componentes y la huella de los mismos.

Eagle es conocido por su facilidad de uso y la amplia gama de herramientas que ofrece para el diseño d circuitos electrónicos. El software tiene una interfaz de usuario intuitiva y fácil de usar, lo que permite a los diseñadores crear diseños de PCB de alta calidad de manera rápida y eficiente. Una de las características más destacadas de Eagle es su amplia biblioteca de componentes, lo que facilita la selección de componentes y la integración de los mismos en el diseño.

Otra característica destacada de Eagle es su capacidad para generar archivos de fabricación de PCB, lo que facilita la fabricación de prototipos y la producción en masa. Los archivos generados por Eagle son compatibles con la mayoría de las empresas de fabricación de PCB, lo que hace que el proceso de fabricación sea más fácil y rentable.

La importancia de Eagle en la industria de la electrónica es innegable. Eagle es utilizado por ingenieros y diseñadores de todo el mundo para diseñar circuitos electrónicos para una amplia gama de aplicaciones, incluyendo la industria automotriz, la industria aeroespacial y la electrónica de consumo. La facilidad de uso de Eagle y la capacidad de generar archivos de fabricación de PCB compatibles con la mayoría de las empresas de fabricación de PCB hacen que sea una herramienta imprescindible para cualquier diseñador de circuitos electrónicos.

Eagle es un software muy útil en el diseño de PCBs con un chip ESP32, ya que permite personalizar los componentes y la huella de los mismos. Esto es especialmente importante en el caso del ESP32, ya que es un chip relativamente nuevo y no todos los fabricantes tienen una huella estándar para él. La personalización de la huella de los componentes es esencial para garantizar una conexión adecuada y minimizar las interferencias y el ruido en la señal.

Al diseñar una PCB con un chip ESP32 utilizando Eagle, se deben tener en cuenta varios aspectos. Primero, se debe seleccionar el tamaño y la forma adecuados de la placa para alojar el chip y los componentes necesarios. Luego, se debe crear el esquemático de la PCB, donde se colocan los componentes y se establecen las conexiones entre ellos. En este punto, es importante prestar atención a la correcta colocación y orientación de los componentes para evitar errores en la conexión y evitar interferencias.

Dentro del circuito completo del sistema, se pueden identificar y seccionar los siguientes circuitos principales:

- Circuito de Activación

El circuito de activación, es esencial en el diseño del chip ESP32-C3-WROOM-02. El pin de "enable" o "EN" controla el funcionamiento del chip, determinando si está en modo operativo o de bajo consumo. Conectado a una fuente de voltaje, habilita el chip; conectado a tierra, lo desactiva para ahorro de energía. Este pin es una entrada activa en bajo, es decir, un voltaje bajo activa el chip y un voltaje alto lo desactiva, influyendo directamente en el consumo de energía y la duración de la batería.

- Circuito de Carga

El circuito de carga, basado en el MCP73831, es crucial para cargar de manera segura una batería de polímero de litio Li-Po de 3.7V y 40mAh. Este integrado proporciona una solución completa, deteniendo la carga automáticamente para evitar sobrecargas y ofreciendo protección contra sobrecorriente y sobrecalentamiento. También mantiene la batería en estado de flotación tras la carga completa, optimizando el tiempo de carga y la eficiencia. Disponible en paquetes DFN de 8 pines y SOT-23 de 5 pines, es ideal para aplicaciones portátiles debido a su tamaño compacto y mínimo número de componentes externos.

- Circuito Regulador de Voltaje

El MCP1700 es una familia de reguladores de voltaje CMOS de baja caída (LDO), capaz de entregar hasta 250 mA de corriente con solo 1,6 µA de corriente de reposo. Con un rango operativo de entrada de 2,3V a 6,0V, es ideal para aplicaciones con baterías de dos y tres celdas primarias o de una celda Li-Ion. En el circuito de carga de una batería Li-Po de 3.7V y 40mAh, el MCP1700-3302E regula la tensión a 3.3V, asegurando una salida estable y constante. Las opciones de paquete incluyen SOT23, SOT89-3 y TO92.

- Circuito de Reparto de Carga

El circuito de reparto de carga, fundamental en aplicaciones portátiles, gestiona la energía de fuentes externas como un puerto USB, para cargar la batería y alimentar el dispositivo. Emplea un MOSFET como interruptor electrónico para controlar el flujo de corriente. Con la conexión USB, el MOSFET permite la carga de la batería; desconectada la fuente externa, la batería alimenta el dispositivo. El circuito incluye un divisor de voltaje, con resistencias configuradas para crear una caída de voltaje proporcional a sus valores, un capacitor para filtrar y estabilizar el voltaje de salida, y un diodo Schottky para proteger contra polaridad inversa. Estas características aseguran un suministro continuo y estable de energía, incluso en ausencia de la fuente externa.

4.2.1 Circuito de activación

El circuito de activación, representado en la figura 7, es una parte esencial del diseño del chip ESP32-C3-WROOM-02. El pin de "enable" o "EN" es utilizado para controlar el funcionamiento del chip y determinar si se encuentra en modo de operación o en modo de bajo consumo. Este pin puede ser conectado a una fuente de voltaje para habilitar el chip y ponerlo en funcionamiento, o conectado a tierra para deshabilitarlo y ponerlo en modo de bajo consumo.

El término "pin de 'enable'" se refiere a un pin específico en un dispositivo electrónico que se utiliza para habilitar o deshabilitar una función o un componente del dispositivo. En este contexto, el pin de "enable" se usa para activar o desactivar el chip, es decir, para encenderlo o apagarlo.

Es importante destacar que la activación y desactivación del chip tiene un impacto directo en el consumo de energía y la duración de la batería. Por lo tanto, es fundamental diseñar un circuito de activación eficiente y adecuado para el proyecto en cuestión. Además, es importante tener en cuenta que el pin de "enable" es una entrada activa en bajo, lo que significa que se requiere una señal de nivel bajo para desactivar el chip.

Cuando se menciona que el pin de "enable" es una "entrada activa en bajo", significa que su estado activo se alcanza cuando se aplica un nivel de voltaje bajo (0 voltios o "LOW" en el caso de señales digitales). En otras palabras, para activar el chip y permitir que funcione, se debe aplicar un voltaje bajo al pin de "enable". Por el contrario, cuando se aplica un voltaje alto (nivel lógico "HIGH"), el chip se desactiva y no realiza ninguna función.

Por lo tanto, para desactivar el chip y detener su funcionamiento, se requiere una señal de nivel bajo, es decir, se debe aplicar un voltaje bajo al pin de "enable". Esta es una consideración importante a tener en cuenta al diseñar circuitos que incluyan este chip, ya que la activación o desactivación del chip estará determinada por el estado del pin de "enable".

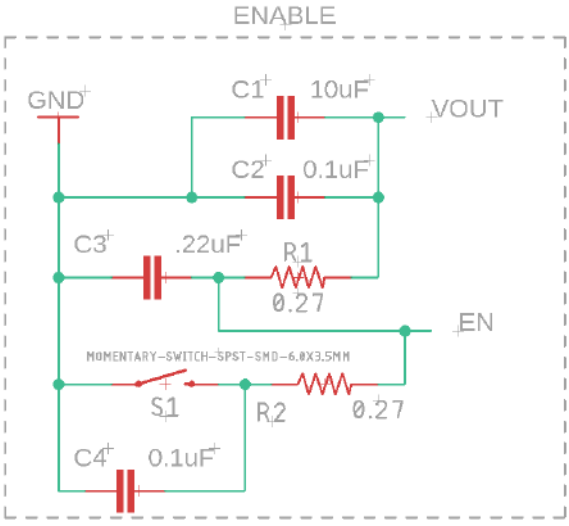


Figura 7. Circuito de activación

4.2.2. Circuito de carga

El circuito de carga es una parte esencial en este sistema, y se basa en el uso del MCP73831, un circuito integrado de carga de batería de iones de litio que proporciona una solución completa para cargar de manera segura y eficiente una batería de polímero de litio Li-Po Este circuito es altamente confiable y seguro, ya que cuenta con una función de desconexión de la carga que protege la batería de la sobrecarga. Una vez que la batería está completamente cargada, el MCP73831 detiene automáticamente la carga para evitar la sobrecarga de la batería, además de ofrecer protección contra sobre corriente y sobrecalentamiento, asegurando la seguridad tanto del circuito de carga como de la batería en todo momento.

El MCP73831 también puede actuar como un circuito de gestión de energía. Una vez que la batería está completamente cargada, puede mantenerla en un estado de flotación para preservar su carga, lo que resulta especialmente útil para aplicaciones que requieren una larga duración de la batería. Además, incorpora una función de "pre-acondicionamiento" que reduce el tiempo de carga de la batería, optimizando su eficiencia y velocidad de carga. Los dispositivos MCP73831 emplean un algoritmo de carga de corriente constante/voltaje constante con preacondicionamiento y terminación de carga seleccionables. La regulación de voltaje constante está fijada con cuatro opciones disponibles: 4.20V, 4.35V, 4.40V o 4.50V, para adaptarse a los nuevos y emergentes requisitos de carga de batería. El valor de corriente constante se establece con un resistor externo. Limitan la corriente de carga en función de la temperatura del sustrato durante condiciones de alta potencia o ambiente elevado, optimizando el tiempo del ciclo de carga y manteniendo la confiabilidad del dispositivo.

Los dispositivos MCP73831 son controladores de gestión de carga lineal altamente avanzados diseñados para su uso en aplicaciones con espacio limitado y sensibles al costo. Están disponibles en un paquete DFN de 8 pines de 2 mm x 3 mm o en un paquete SOT-23 de 5 pines tal como se observan en la figura 8. Además de su reducido tamaño físico, el bajo número de componentes externos requeridos hace que los MCP73831/2 sean idealmente adecuados para aplicaciones portátiles. Cumplen con todas las especificaciones que rigen el bus de energía USB para aplicaciones que se cargan desde un puerto USB.

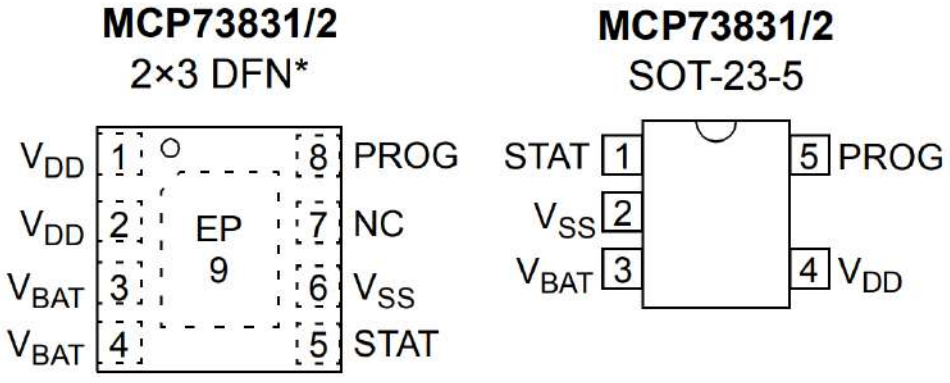


Figura 8. Formatos de paquete para MCP73831/2

La descripción de los pines del MCP73831, presentada en la Tabla 4.1, ofrece una visión detallada de la disposición física y la función de cada pin en este controlador de carga. Esto facilita la comprensión de cómo conectar y configurar correctamente el MCP73831 en un diseño electrónico, lo que asegura un rendimiento óptimo y una operación segura durante la carga de la batería.

Tabla 4.1. Descripción de los pines del microcontrolador ESP32-C3-WROOM-02

No. Pin		Símbolo	Función
DFN	SOT- 23		
1	4	V_{DD}	Suministro de entrada de gestión de batería
2	---	V_{DD}	Suministro de entrada de gestión de batería
3	3	V_{BAT}	Salida de control de carga de batería
4	---	V_{BAT}	Salida de control de carga de batería
5	1	STAT	Salida de estado de carga
6	2	V_{SS}	Gestión de batería 0V Referencia
7	---	NC	Sin conexión
8	5	PROG	Conjunto de regulación actual y habilitación de control de carga
9	---	EP	Almohadilla térmica expuesta (EP); debe estar conectado a V_{SS}

Los dispositivos MCP73831/2 están diseñados para funcionar junto con un microcontrolador host o en una aplicación independiente. Proporcionan el algoritmo de carga preferido para celdas de iones de litio y polímero de litio. La selección de los componentes externos en la figura 9 es crucial para la integridad y confiabilidad del sistema de carga para una batería de polímero de litio de 1000 mAh.

Existen varias opciones disponibles para el umbral de preacondicionamiento, el valor de corriente de preacondicionamiento, el valor de terminación de carga y el umbral de recarga automática. El preacondicionamiento puede desactivarse. Los dispositivos MCP73831/2 están totalmente especificados en el rango de temperatura ambiente de -40°C a +85°C.

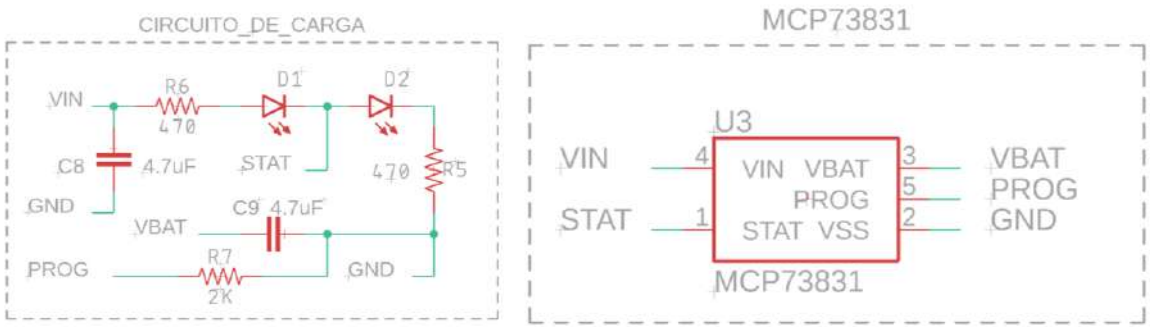


Figura 9. Circuito de carga utilizando el dispositivo MCP73831

El algoritmo de carga del MCP73831/2 utiliza una corriente constante seguida de un método de carga de voltaje constante para cargar eficientemente las baterías de iones de litio y polímero de litio. La Figura 10 representa un perfil de carga típico adjunto para una batería de polímero de litio de 1000 mAh, ilustrando cómo varía la corriente y el voltaje durante el proceso de carga. Este algoritmo asegura una carga segura y eficiente de la batería, optimizando su rendimiento y durabilidad.

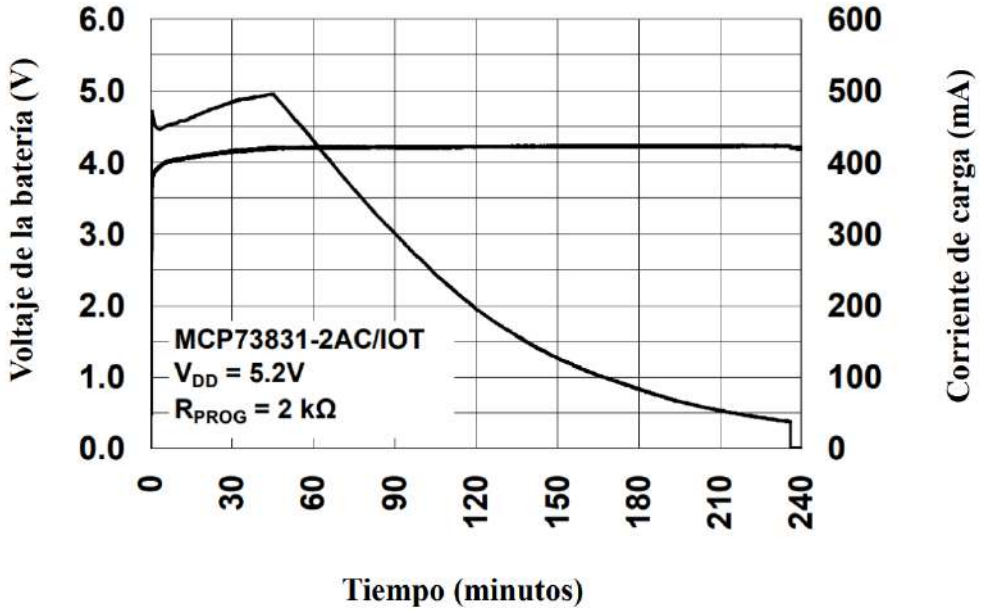


Figura 10. Perfil de carga para una batería de polímero de litio de 1000 mAh

4.2.3. Circuito regulador de voltaje

El MCP1700 es una familia de reguladores de voltaje CMOS de baja caída (LDO) que pueden entregar hasta 250 mA de corriente mientras consumen solo 1,6 µA de corriente de reposo (típico). El rango operativo de entrada está especificado de 2,3V a 6,0V, haciéndolo ideal para aplicaciones alimentadas por baterías de dos y tres celdas primarias, así como aplicaciones de Li-Ion de celda única.

El MCP1700-3302E es un regulador de voltaje de bajo consumo de corriente adecuado para aplicaciones que requieren un voltaje constante y estable. En el circuito de carga de la batería de polímero de litio Li-Po de 3.7V y 1000mAh, se utiliza para regular la tensión de la batería a un nivel constante de 3.3V.

La regulación del voltaje es crucial en cualquier circuito electrónico, especialmente en aquellos que implican la carga de la batería. La batería de polímero de litio Li-Po es un tipo de batería que puede ser cargada y descargada muchas veces sin sufrir daños. Sin embargo, para asegurar una carga segura y eficiente, se necesita un regulador de voltaje confiable como el MCP1700-3302E.

El MCP1700-3302E es un regulador de voltaje lineal de bajo consumo que puede proporcionar una salida de voltaje estable y constante de 3.3V con una corriente máxima de 250mA. Se conecta entre la batería y la carga para garantizar que la carga reciba una tensión constante y estable, independientemente de la carga de la batería Las opciones de paquete incluyen SOT23, SOT89-3 y TO92, como se observa en la figura 11.

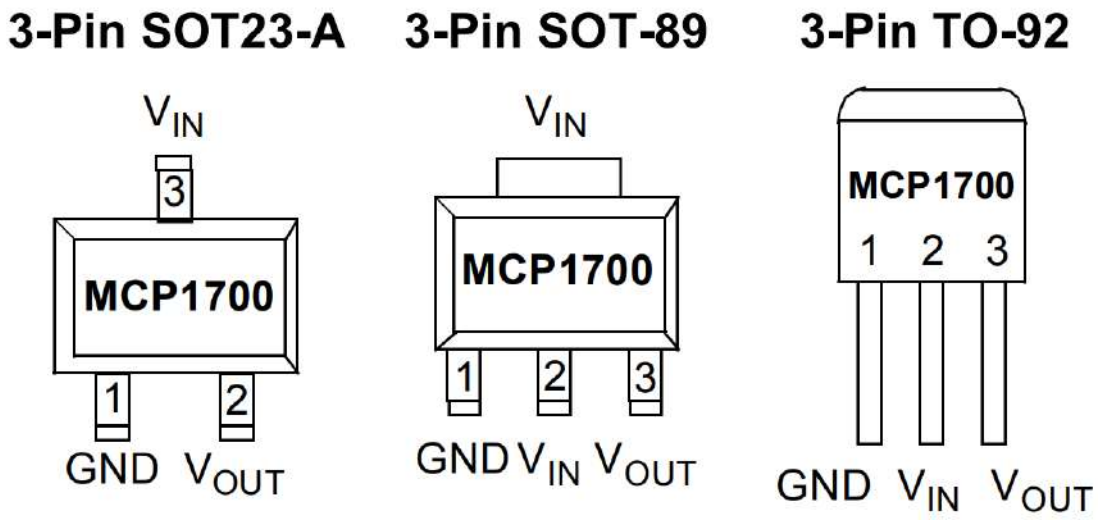


Figura 11. Formatos de paquete para el dispositivo MCP73831

El MCP1700 es capaz de entregar 250 mA con solo 178 mV de diferencial de voltaje de entrada a salida (VSALIDA = 2,8 V). La tolerancia del voltaje de salida del MCP1700 suele ser ±0,4 % a +25 °C y ±3 % máximo sobre la temperatura de la unión operativa en un rango de -40°C a +125°C. Los voltajes de salida disponibles para el MCP1700 varían desde 1.2 V hasta 5.0 V. La salida LDO es estable cuando se usa solo una capacitancia de salida de 1 µF. Se pueden utilizar capacitores cerámicos, de tantalio o electrolíticos de aluminio tanto para entrada como para salida. El límite de sobre corriente y sobre temperatura proporciona una solución sólida para cualquier aplicación.

Entre las características más importantes del MCP1700 se encuentran:

- Corriente de reposo típica de 1,6 µA
- Rango de voltaje de funcionamiento de entrada: 2,3 V a 6,0 V
- Rango de voltaje de salida: 1,2 V a 5,0 V
- Corriente de salida de 250 mA para voltajes de salida ≥ 2,5 V
- Corriente de salida de 200 mA para voltajes de salida < 2,5 V
- Voltaje de caída baja (LDO):

  - 178 mV típico a 250 mA para VOUT = 2,8 V

- Tolerancia típica del voltaje de salida del 0,4%
- Opciones de voltaje de salida estándar:

  - 1,2 V, 1,8 V, 2,5 V, 3,0 V, 3,3 V, 5,0 V

- Estable con condensador de salida cerámica de 1,0 µF
- Protección contra cortocircuitos
- Protección contra sobrecalentamiento

La descripción de los pines del MCP1700, presentada en la Tabla 4.2, ofrece una visión de la disposición física y la función de cada pin en este dispositivo.

Tabla 4.2. Descripción de los pines del dispositivo MCP1700

Pin No. SOT232-A	Pin No. SOT89	Pin No. TO-92	Nombre	Función
1	1	1	GND	Terminal de tierra
2	3	3	V_{OUT}	Salida de voltaje regulada
3	2	2	V_{IN}	Voltaje de suministro no regulado

La utilización del MCP1700-3302E en el circuito de carga de la batería de polímero de litio Li-Po de 3.7V y 1000mAh es fundamental para proteger tanto la batería como la carga. Además, el bajo consumo de corriente del regulador asegura una eficiente utilización de la energía, lo que es esencial para aplicaciones en las que la duración de la batería es crítica. En la Figura 12 se puede apreciar el circuito regulador de voltaje.

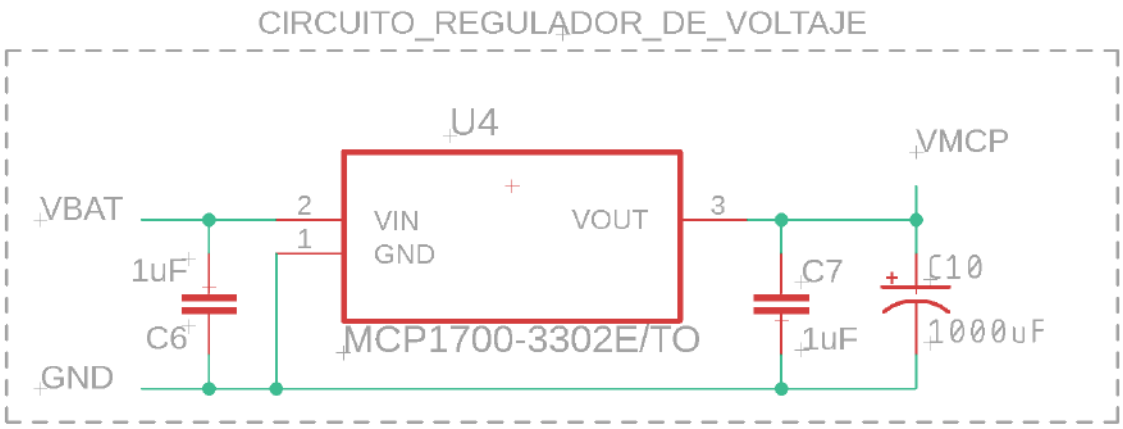


Figura 12. Circuito regulador de voltaje

4.2.4. Circuito de reparto de carga

El circuito de reparto de carga es una solución fundamental en numerosas aplicaciones portátiles, especialmente en dispositivos alimentados por batería, donde es crucial mantener una carga de batería constante y estable. Este circuito se encarga de gestionar la energía entrante proveniente de fuentes externas, como un puerto USB, y dirigirla adecuadamente para cargar la batería y alimentar el dispositivo conectado.

En su funcionamiento, el circuito de reparto de carga emplea un componente llamado MOSFET, abreviatura de Transistor de Efecto de Campo de Metal-Óxido-Semiconductor, el cual actúa como un interruptor electrónico. Este MOSFET permite controlar el flujo de corriente entre la fuente de alimentación externa y la batería. Cuando se conecta un puerto USB para cargar la batería, el MOSFET se activa y permite que la corriente fluya hacia la batería, lo que inicia el proceso de carga. Por otro lado, cuando se desconecta la alimentación externa, el MOSFET se desactiva y permite que la carga almacenada en la batería alimente el dispositivo conectado, asegurando así un suministro continuo de energía, incluso en ausencia de la fuente externa.

Además, el circuito incluye un divisor de voltaje, una configuración de resistencias que reduce el voltaje de la batería y proporciona una señal más baja al sistema de control. Las resistencias en el divisor de voltaje están configuradas en una relación de división de voltaje, lo que significa que crean una caída de voltaje proporcional a la relación de sus valores. En este caso, se seleccionan tres resistencias con valores específicos para obtener el voltaje de salida deseado. El capacitor se utiliza para filtrar y estabilizar el voltaje de salida, eliminando cualquier ruido o fluctuaciones no deseadas, lo que ayuda a mantener un voltaje constante y suaviza cualquier variación que pueda ocurrir debido a cambios en la carga o en la fuente de alimentación. El diodo Schottky se coloca en paralelo con la salida del divisor de voltaje y actúa como una protección contra polaridad inversa, evitando que la corriente fluya en la dirección opuesta, lo que podría dañar el circuito o la batería.

La figura 13 ilustra el circuito de reparto de carga, mientras que la Tabla 4.3 detalla la Descripción de los Pines del circuito.

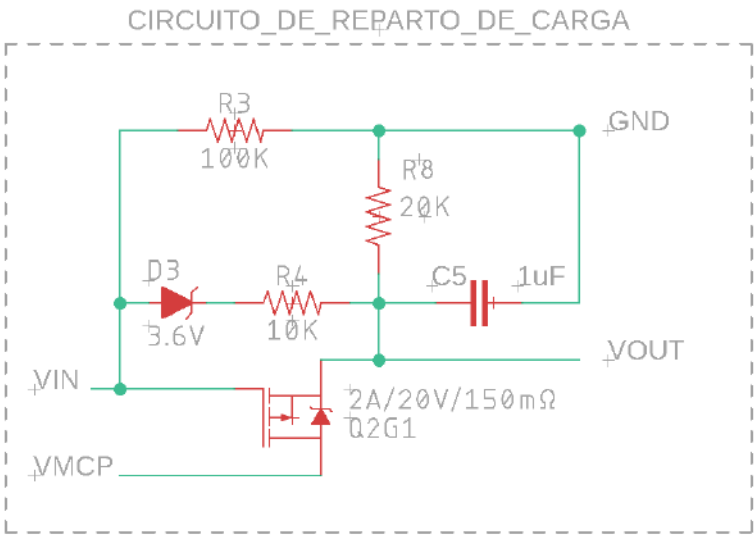


Figura 13. Circuito de reparto de carga

Tabla 4.3. Descripción de los pines del circuito de reparto de carga

Nombre	Descripción
VIN	Fuente de alimentación
VMCP	Entrada de voltaje regulado
VOUT	Salida de voltaje dirigido al microcontrolador y el sensor
GND	Terminal de tierra

4.2.5. Esquemático completo del circuito

El esquemático completo del circuito se puede observar en la Figura 14. Consta de varios componentes, incluyendo el sensor DHT11 para medir la temperatura y la humedad, el chip ESP32-C3-WROOM-02 para controlar y procesar los datos, una batería de litio para la alimentación de energía, un dip switch, un puerto micro USB AB para la carga y la comunicación, y varios otros componentes como resistencias, capacitores y diodos para el correcto funcionamiento del circuito. Todos estos componentes están conectados de manera interdependiente en el esquemático, lo que permite que el circuito funcione de manera óptima y cumpla con los objetivos de medición y monitoreo de temperatura y humedad en tiempo real

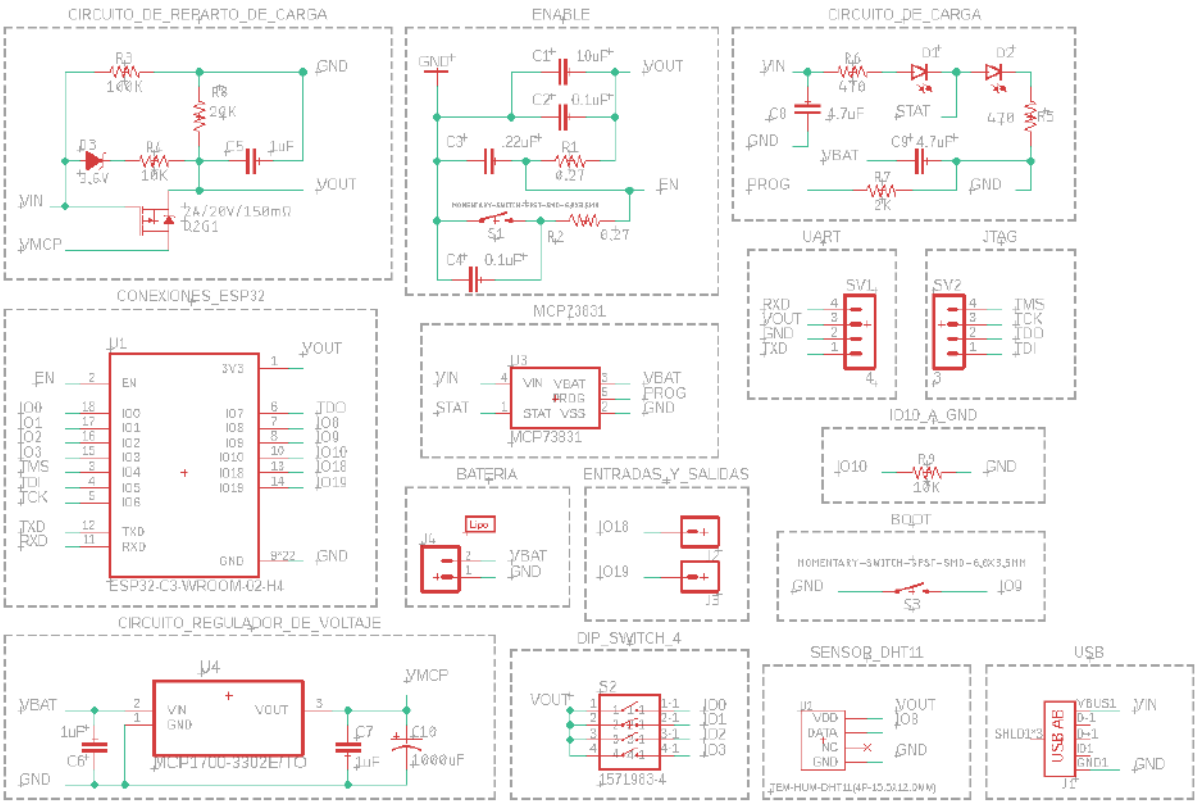


Figura 14. Esquemático completo del circuito.

4.3. Diseño de la tarjeta PCB

El diseño de PCB es una parte crucial del proceso de desarrollo de un dispositivo electrónico. Una vez que el esquemático ha sido diseñado y verificado, se debe proceder a la creación del diseño de la PCB. En este proceso, se deben definir las capas de la PCB, la posición y orientación de los componentes, y establecer las pistas de conexión entre ellos. Además, se deben definir las zonas de alimentación y tierra, y los planos de cobre, que son esenciales para reducir el ruido y mejorar la calidad de la señal.

En el diseño de la PCB para este proyecto, se deben incluir varios componentes, entre ellos, la batería de polímero de litio Li-Po de 3.7V y 40mAh, el MCP1700-3302E, el MCP73831, el sensor DHT11, un dip switch, un puerto micro USB AB, así como varios otros componentes como resistencias, capacitores y diodos.

La batería de polímero de litio Li-Po de 3.7V y 40mAh es uno de los componentes más importantes del circuito, ya que es la fuente de energía para el dispositivo. Por lo tanto, es crucial ubicarla en un lugar adecuado y seguro. Para ello, se puede colocar en una esquina de la PCB donde no interfiera con otros componentes. Además, se debe considerar la ubicación del regulador de voltaje MCP1700-3302E, ya que este componente es responsable de regular la tensión de la batería a un nivel constante de 3.3V. Por lo tanto, debe colocarse lo más cerca posible de la batería para minimizar las pérdidas de voltaje en el camino.

El MCP73831 es el circuito integrado utilizado para cargar la batería y debe colocarse cerca de la batería para reducir la longitud de las pistas de conexión y minimizar las pérdidas de voltaje. El sensor DHT11, que se utiliza para medir la temperatura y la humedad, se puede colocar cerca del microcontrolador para simplificar las conexiones.

El puerto micro USB AB se utilizan para controlar la alimentación del dispositivo y para conectarlo a una fuente de alimentación externa. Se deben colocar en un lugar de fácil acceso para facilitar el uso del dispositivo. En cuanto a los componentes adicionales, como resistencias, capacitores y diodos, se deben ubicar cerca de los componentes que están destinados a proteger o filtrar las señales.

El diseño completo de la tarjeta prototipo PCB se puede apreciar en la Figura 15.

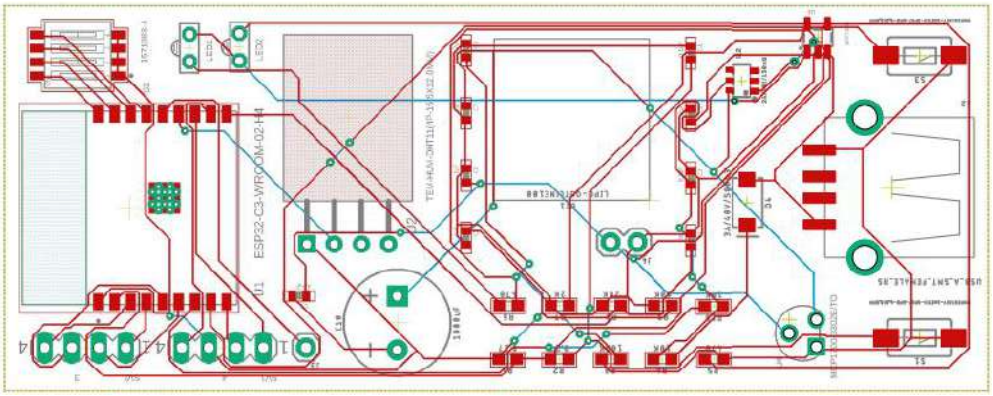


Figura 15. Tarjeta prototipo PCB desarrollada.

Después de la fase prototipo, se implementaron mejoras significativas en el diseño. Estas mejoras abarcaron la redistribución y dimensionamiento de los componentes, así como la reducción general del tamaño de la tarjeta. Además, se optimizó la posición del ESP32 para cumplir con requisitos específicos, lo que resultó en conexiones más limpias y ordenadas. Estas modificaciones se pueden apreciar claramente en la figura 16. También se realizaron cambios en los componentes utilizados en el puerto USB, donde los capacitores y resistores de tamaño 0401 se reemplazaron por unos de tamaño 1206. Además, se optó por utilizar un conector micro USB tipo B para mejorar la durabilidad y la facilidad de uso del dispositivo.

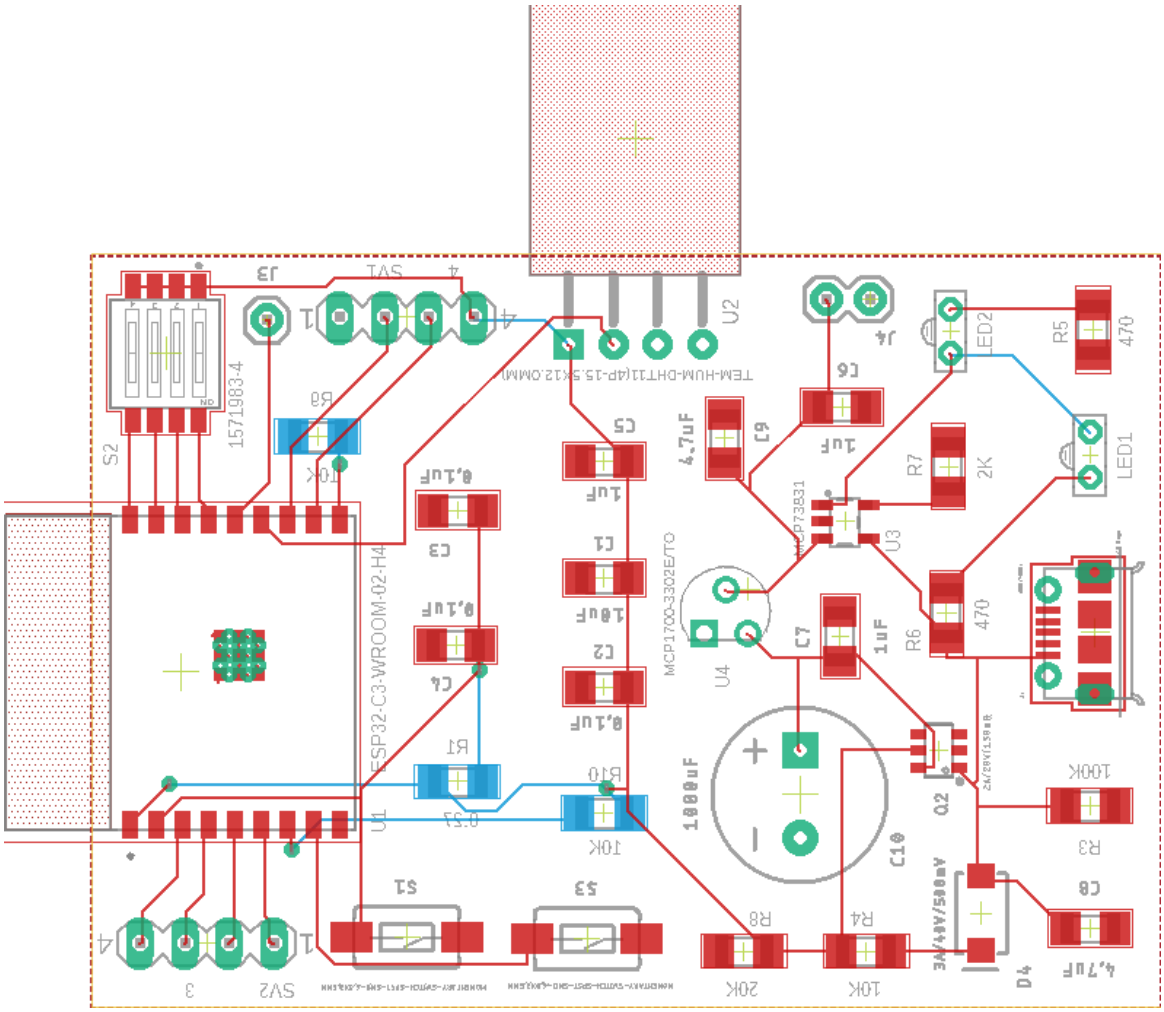


Figura 16. Tarjeta final PCB desarrollada.

4.3.1 Soldar los componentes en la PCB

Una vez que se ha completado el diseño del circuito eléctrico, el siguiente paso implica la soldadura de los componentes en la PCB. Es crucial ejecutar este proceso con precisión, evitando errores y siguiendo las instrucciones de soldadura específicas para cada componente. La figura 17 proporciona una visión detallada de la tarjeta con todos los componentes integrados, ofreciendo así una representación visual de la tarjeta final desarrollada. Este paso es esencial para garantizar la funcionalidad y confiabilidad del circuito en su conjunto.

Figura 17. Tarjeta PCB con sus componentes.

4.4 Programar los chips ESP32-C3-WROOM-02

El sistema se fundamenta en la utilización de nodos ESP32 con tecnología ESP Mesh para establecer una red ad hoc, conectados a sensores de temperatura y humedad DHT11. La operatividad del sistema se describe detalladamente:

- Nodo Maestro:

Un nodo ESP32 es designado como el maestro de la red ad hoc utilizando la tecnología ESP Mesh. Este nodo central se encarga de recopilar información de temperatura y humedad de todos los nodos de la red, incluyendo el suyo propio. Posteriormente, esta información se envía al ESP32 designado como puente o bridge a través de comunicación serial, utilizando el puerto UART de ambos dispositivos para establecer una conexión confiable y eficiente.

- Nodos Esclavos:

Los demás nodos ESP32 son designados como esclavos y se conectan directamente al nodo maestro. Cada uno de estos nodos esclavos está equipado con un sensor DHT11, el cual se encarga de medir la temperatura y humedad del entorno. Una vez recopilada esta información, se envía al nodo maestro para su procesamiento y posterior transmisión a la plataforma.

- Nodo Maestro y Esclavo Simultáneo:

Al menos un nodo ESP32 actúa como un nodo maestro y esclavo simultáneamente. Este nodo en particular se conecta directamente al nodo maestro principal, facilitando la comunicación y coordinación entre los demás nodos esclavos. Además, este nodo asume el papel de maestro hacia otros nodos esclavos adicionales, lo que permite la expansión de la red y la adición de más nodos para monitorear la temperatura y humedad en diversas ubicaciones.

- Bridge:

El nodo designado como puente, o bridge, desempeña un papel crucial en el sistema. Por un lado, se comunica con la red ad hoc mediante comunicación serial a través del puerto UART del nodo maestro, asegurando la transmisión confiable de datos. Por otro lado, establece una conexión con la plataforma utilizando el protocolo MQTT, donde la información recopilada se visualiza en la página web MQTTX. Esta funcionalidad del puente garantiza la integración efectiva entre la red ad hoc y la plataforma de visualización, permitiendo un monitoreo remoto y accesible de la temperatura y humedad en tiempo real.

En situaciones normales, los nodos permanecen en espera de una orden de activación sin realizar ninguna función específica.

Este sistema basado en el chip ESP32 y la red ad hoc con ESP Mesh permite una comunicación eficiente y confiable entre los nodos de la red. Los sensores de temperatura y humedad DHT11 proporcionan datos precisos para el monitoreo ambiental. La plataforma web y el protocolo MQTT permiten una visualización y análisis convenientes de los datos recopilados.

La Figura 18 presenta el diagrama de flujo del comportamiento del sistema, representando las diferentes etapas y las interacciones entre los nodos. Este diagrama de flujo ilustra claramente el flujo de datos y las acciones realizadas en cada etapa del sistema, proporcionando una representación visual comprensible y organizada de las operaciones que se llevan a cabo. Con este sistema implementado, se logra un monitoreo de la temperatura y humedad en diferentes ubicaciones, permitiendo tomar decisiones informadas y realizar ajustes según sea necesario. Además, la estructura de red ad hoc con ESP Mesh brinda flexibilidad y escalabilidad para expandir el sistema en el futuro, adaptándolo a nuevas necesidades y áreas de monitoreo.

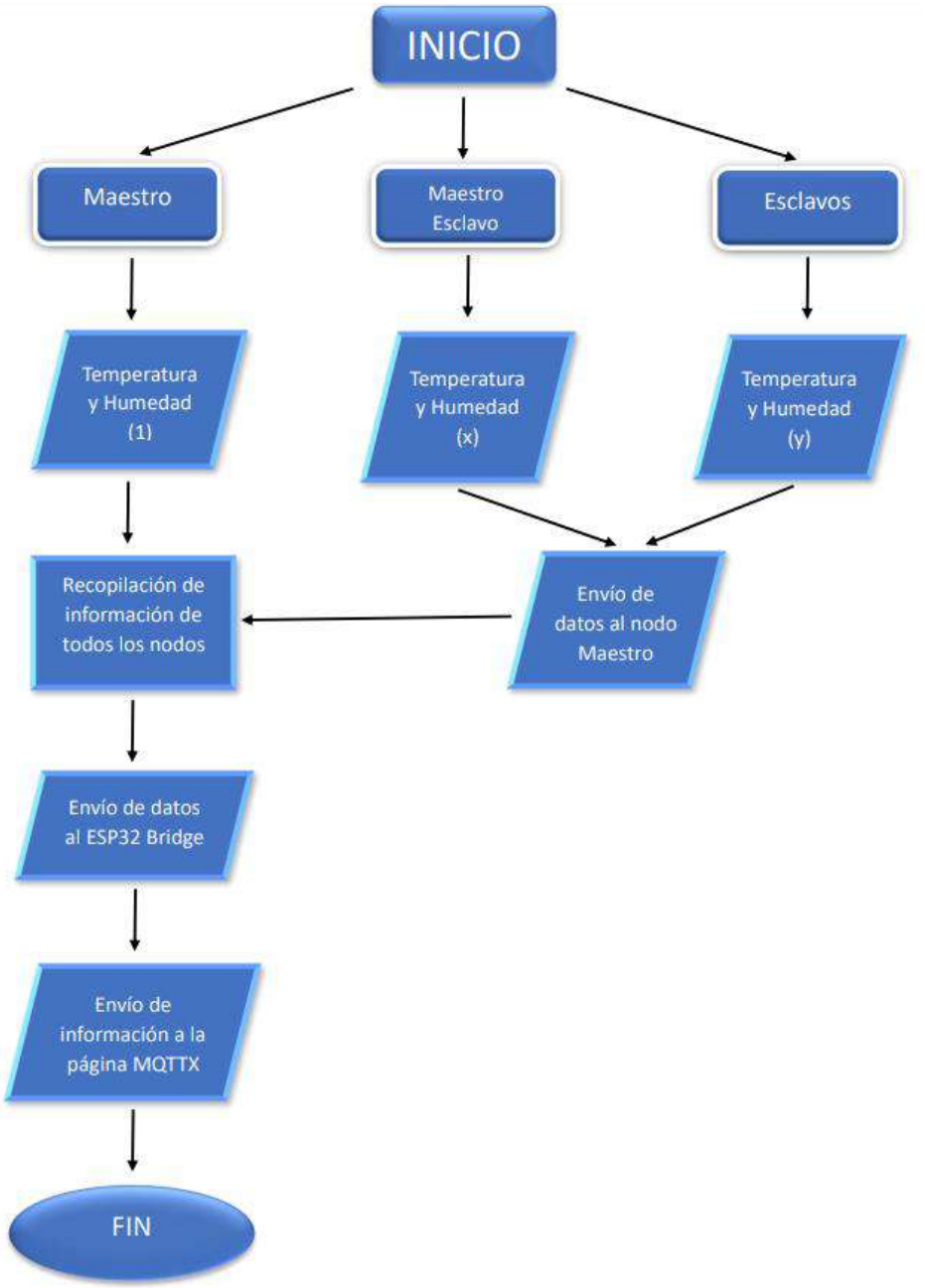


Figura 18. Diagrama de flujo del comportamiento del sistema

Para programar los chips ESP32-C3-WROOM-02, es necesario descargar el software adecuado para programar la placa. La mayoría de los fabricantes de placas tienen un sitio web donde se puede descargar el software necesario. Se debe descargar e instalar en el ordenador.

Para integrar la tarjeta de desarrollo en el entorno, es necesario agregar el siguiente enlace en el archivo de configuración, específicamente en la sección de Preferencias

https://raw.githubusercontent.com/espressif/arduino-esp32/pages/pachage_esp32_index.json-

Este proceso facilita la inclusión y el reconocimiento de la tarjeta dentro del entorno de desarrollo, lo que permite su uso eficiente en proyectos posteriores. Este enlace proporciona acceso directo a las herramientas y bibliotecas necesarias para programar y trabajar con la tarjeta de desarrollo de manera efectiva. Además, garantiza una configuración correcta y una integración sin problemas con el software de desarrollo. La Figura 19 muestra el proceso de cómo agregar la tarjeta de desarrollo en el archivo de configuración.

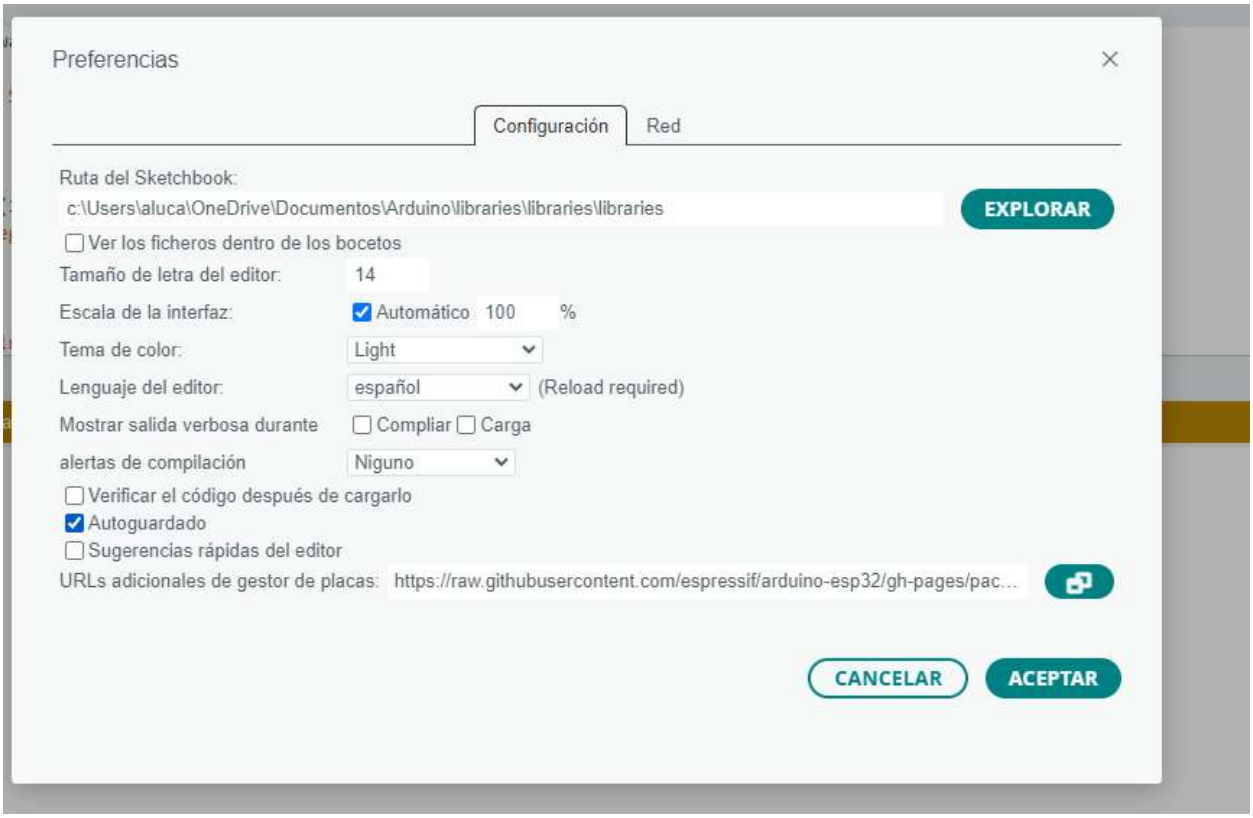


Figura 19. Proceso de Integración de Tarjeta de Desarrollo en el Entorno de Desarrollo

Una vez instalado el software, el siguiente paso consiste en conectar la placa utilizando un cable USB. Es posible que sea necesario instalar los controladores específicos de la placa para que el ordenador pueda reconocerla correctamente. Una vez que la placa está conectada, se procede a abrir el software y seleccionar la placa que se desea programar. Este proceso se puede observar en la figura 20.

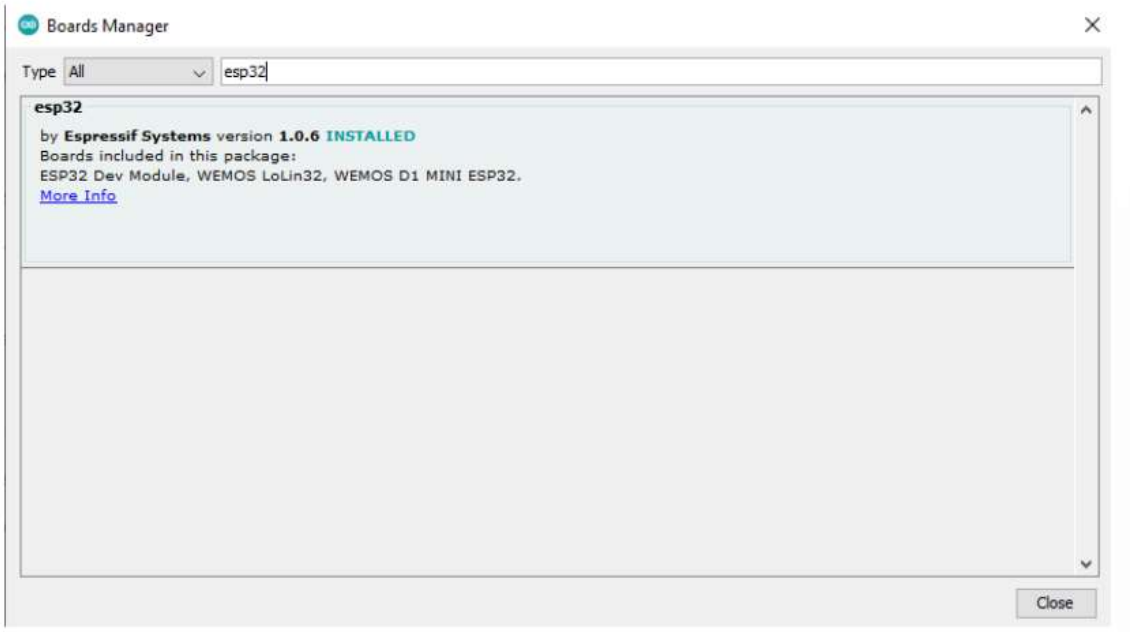


Figura 20. Selección de placa y modelo del software

Una vez que se ha seleccionado la placa y se ha establecido la conexión, se debe escribir el código que se desea cargar en ella. El código debe ser escrito en el lenguaje de programación adecuado para la placa específica que se está utilizando. En la siguiente sección se explicará en detalle el código necesario, así como otros aspectos relevantes del proceso de programación. Se debe tener en cuenta que cada placa puede requerir un lenguaje de programación diferente, por lo que es importante investigar cuál es el lenguaje adecuado para la placa en cuestión. Una vez que se ha escrito el código, se debe verificar para asegurarse de que no haya errores. Si el código está libre de errores, se puede compilar y cargar en la placa. Una vez que se ha cargado el código en la placa, ésta ejecutará las instrucciones que se hayan escrito.

4.4.1 Configurar la red ad hoc con ESP Mesh y MQTT

Para establecer la red ad hoc, se debe configurar cada ESP32 con ESP Mesh para que puedan comunicarse entre sí. Cada nodo de la red se identificará con una configuración de nodo y tendrá un sensor DHT11 para medir la temperatura y humedad ambiente.

El proceso para configurar la red ad hoc comenzará con la conexión de los nodos mediante ESP Mesh. Cada nodo se identificará con un número único a través de su configuración, lo que permitirá la identificación individual de cada uno. A continuación, se procederá a la medición de la temperatura y la humedad ambiente utilizando el sensor DHT11, lo que permitirá la recolección de datos para su posterior análisis.

Una vez recolectados los datos, se procederá a enviarlos al nodo principal de la red ad hoc. Este nodo será el encargado de recopilar los datos de todos los nodos y enviarlos por comunicación serial al esp32 bridge

El esp32 bridge se conecta a la plataforma de IoT, en este caso, a través del protocolo MQTT a una página web dedicada al monitoreo. La plataforma permitirá el monitoreo en tiempo real de los datos recolectados, así como la generación de alertas y notificaciones en caso de valores anormales.

Para programar la red ad hoc con ESP Mesh utilizando ESP32-C3-WROOM-02, se necesita el software adecuado y un conocimiento básico de programación en lenguaje C++. Se inició incluyendo las librerías necesarias para el uso de ESP Mesh y el sensor DHT11. A continuación, se configuró el pin del sensor y se definió la dirección MAC de cada nodo utilizando la configuración dentro de la red mesh. También se configuró la batería de polímero de litio y se ajustaron los tiempos de medición del sensor.

Luego, se configura la red ad hoc utilizando la librería ESP32-Mesh. Se definió el servicio y las características de ESP Mesh que se utilizarían para la transmisión de datos y se creó la función que manejaría los eventos de la red. Además, se configuró el nodo principal para recibir los datos de los demás nodos y enviarlos al esp32 bridge y este a su vez a la página web mediante el protocolo MQTT.

Una vez que se tenía la programación lista, se cargó el firmware en cada nodo utilizando un cable USB y el software adecuado para la programación. Luego se procedió a la instalación física de los nodos y se probó el funcionamiento de la red ad hoc. Este enfoque con ESP Mesh y MQTT proporciona una solución robusta y eficiente para el monitoreo remoto de temperatura y humedad.

A continuación, se explica en detalle el código del nodo maestro de la red ad hoc:

Inclusión de bibliotecas:

- Adafruit_Sensor.h: Esta librería proporciona una interfaz común para acceder a los datos de los sensores. Es utilizada por muchos sensores de Adafruit para proporcionar una interfaz de lectura uniforme de datos como temperatura, humedad, luz, movimiento, etc.

- DHT.h: Esta librería permite interactuar con los sensores de temperatura y humedad DHT. Los sensores DHT son dispositivos simples y económicos que pueden medir tanto la temperatura como la humedad relativa del ambiente. Esta librería simplifica la lectura de datos de estos sensores.

- PainlessMesh.h: Esta librería proporciona una manera sencilla de configurar y gestionar una red de malla (mesh) utilizando dispositivos basados en ESP8266/ESP32. Permite la comunicación entre nodos de la red de malla de forma sencilla, con soporte para enrutamiento automático y recuperación automática de nodos caídos.

- Arduino_JSON.h: Esta librería facilita el análisis y la generación de datos en formato JSON en dispositivos Arduino. Permite analizar cadenas de texto JSON para extraer datos específicos y también generar cadenas JSON a partir de variables de datos en el programa.

- HardwareSerial.h: Esta librería permite utilizar los puertos seriales de hardware disponibles en la placa. Proporciona funciones para la lectura y escritura de datos a través de estos puertos serie físicos. Esto es útil cuando se necesitan múltiples puertos serie o cuando se requiere una comunicación serie de alta velocidad y fiabilidad.

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include "painlessMesh.h"
#include <Arduino_JSON.h>
#include <HardwareSerial.h>
```

Definición de constantes:

DHT_PIN: Esta línea define el pin al que está conectado el sensor DHT en el microcontrolador. En este caso, el valor 5 indica que el sensor está conectado al pin 5.

DHT_TYPE: Esta línea define el tipo de sensor DHT que se está utilizando. En este caso, se está utilizando un sensor DHT22, que es un tipo común de sensor de temperatura y humedad.

MESH_PREFIX: Esta línea define el prefijo que se utilizará para la red de malla (mesh). Cuando los nodos se conectan a la red de malla, utilizarán este prefijo para identificar la red a la que se están uniendo.

MESH_PASSWORD: Esta línea define la contraseña que se utilizará para la red de malla. Los nodos que deseen unirse a la red de malla deberán proporcionar esta contraseña para autenticarse y unirse a la red.

MESH_PORT: Esta línea define el puerto que se utilizará para la comunicación en la red de malla. Todos los nodos en la red de malla utilizarán este puerto para enviar y recibir mensajes entre sí. En este caso, se está utilizando el puerto 5555.

```
#define DHT_PIN 5
#define DHT_TYPE DHT22
#define MESH_PREFIX "RNTMESH"
#define MESH_PASSWORD "MESHpassword"
#define MESH_PORT 5555
```

Variables globales:

- int nodeNumber = 1;: Esta línea declara una variable llamada `nodeNumber` de tipo entero e inicializa su valor en 1. Esta variable se utiliza para almacenar el número de nodo en el sistema.

- String readings;: Aquí se declara una variable de tipo `String` llamada `readings`. Esta variable se utiliza para almacenar las lecturas obtenidas del sensor antes de enviarlas a través de la red de malla.

- Scheduler userScheduler;: Se declara un objeto de tipo `Scheduler` llamado `userScheduler`. El objeto `Scheduler` se utiliza para programar tareas en el sistema.

- painlessMesh mesh;: Se declara un objeto de tipo `painlessMesh` llamado `mesh`. Este objeto se utiliza para configurar y administrar la red de malla.

- double receivedTemp;: Se declara una variable de tipo `double` llamada `receivedTemp`. Esta variable se utiliza para almacenar la temperatura recibida del sensor.

- double receivedHum;: Se declara una variable de tipo `double` llamada `receivedHum`. Esta variable se utiliza para almacenar la humedad recibida del sensor.

- void sendMessage();: Se declara un prototipo de función llamada `sendMessage()`. Esta función se utilizará para enviar mensajes a través de la red de malla.

- String getReadings();**: Se declara un prototipo de función llamada `getReadings()`. Esta función se utilizará para obtener las lecturas del sensor.

- TasktaskSendMessage(TASK_SECOND*5,TASK_FOREVER,&sendMese: Se declara una tarea llamada `taskSendMessage` que ejecutará la función `sendMessage()` cada 5 segundos de manera indefinida.

- DHT dht(DHT_PIN, DHT_TYPE);: Se declara un objeto de tipo `DHT` llamado `dht`. Este objeto se utiliza para interactuar con el sensor de temperatura y humedad DHT.

- HardwareSerial SerialPort(2);: Se declara un objeto de tipo `HardwareSerial` llamado `SerialPort` que utiliza el puerto UART número 2 para la comunicación serial. Este objeto se utilizará para comunicarse con otros dispositivos a través de UART.

```
int nodeNumber = 1;
String readings;
Scheduler userScheduler;
painlessMesh mesh;
double receivedTemp;
double receivedHum;
void sendMessage();
String getReadings();
Task taskSendMessage(TASK_SECOND * 5, TASK_FOREVER, &sendMessage);
DHT dht(DHT_PIN, DHT_TYPE);
HardwareSerial SerialPort(2);
```

Configuración del Entorno (Setup):

En esta sección, se llevan a cabo las configuraciones iniciales necesarias antes de que el programa entre en su bucle principal. Aquí se establecen las comunicaciones serie, se inicializan los sensores y se configura la red de malla.

- Inicio de la Comunicación Serial:
    - Se inicia la comunicación serial principal a una velocidad de 115200 baudios utilizando Serial.begin(115200).
    - También se inicia la comunicación serial secundaria (Serial2) con una velocidad de 15200 baudios, utilizando SerialPort.begin(15200, SERIAL_8N1, 16, 17).
- Inicialización del Sensor DHT:
    - Se inicia el sensor DHT para permitir la lectura de temperatura y humedad utilizando dht.begin().
- Configuración de la Red de Malla:
    - Se establece el tipo de mensajes de depuración para la red de malla con mesh.setDebugMsgTypes(ERROR | STARTUP).
    - Se inicializa la red de malla con el prefijo, la contraseña, el programador de tareas y el puerto especificados usando mesh.init(MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT).
    - Se definen funciones de devolución de llamada para manejar la recepción de mensajes, las nuevas conexiones y los cambios en la red de malla.
- Función de Recepción de Mensajes:
    - Se define una función de recepción de mensajes utilizando mesh.onReceive([](uint32_t from, String &msg) {...}).
    - Dentro de esta función, se analiza el mensaje recibido en formato JSON.
    - Se extraen los valores de "node", "temp" y "hum" del mensaje JSON recibido.
    - Se obtiene el número de nodos en la red de malla y se imprime junto con los valores de "node", "temp" y "hum" en el monitor serie.
    - Se crea un nuevo objeto JSON con los valores extraídos y se convierte a una cadena JSON.
    - La cadena JSON resultante se envía a través de la comunicación serial secundaria (Serial2) utilizando SerialPort.println(jsonStr).
    - Se realiza una pausa de 100 milisegundos antes de continuar.

- Funciones de Manejo de Conexiones de la Red:

    - Se definen funciones de devolución de llamada para manejar nuevos nodos conectados, cambios en las conexiones y ajustes de tiempo en la red de malla.

- Configuración del Programador de Tareas:

    - Se agrega y habilita una tarea de envío de mensajes periódica utilizando el programador de tareas userScheduler.addTask(taskSendMessage) y taskSendMessage.enable().

```
void setup() {
  Serial.begin(115200);
  SerialPort.begin(15200, SERIAL_8N1, 16, 17);
  dht.begin();
  mesh.setDebugMsgTypes(ERROR | STARTUP);
  mesh.init(MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT);
  mesh.onReceive([](uint32_t from, String &msg) {
    Serial.printf("Received from %u msg=%s\n", from, msg.c_str());
    JSONVar myObject = JSON.parse(msg.c_str());
    int node = myObject["node"];
    receivedTemp = myObject["temp"];
    receivedHum = myObject["hum"];
    int nodeCount = mesh.getNodeList().size() + 1;
    Serial.printf("numero de nodos dentro de la red: %d \n", nodeCount);
    Serial.print("Nodo: ");
    Serial.println(node);
    Serial.print("Temperatura: ");
    Serial.print(receivedTemp);
    Serial.println(" C");
    Serial.print("Humedad: ");
    Serial.print(receivedHum);
    Serial.println(" %");
  String receivedTempStr = String(receivedTemp, 2);
  String receivedHumStr = String(receivedHum, 2);
JSONVar jsonObj;
jsonObj["numero_de_nodos"] = nodeCount;
jsonObj["nodo"] = node;
jsonObj["temperatura"] = receivedTempStr;
jsonObj["humedad"] = receivedHumStr;
String jsonStr = JSON.stringify(jsonObj);
SerialPort.println(jsonStr);
delay(100);
  float tempe = dht.readTemperature();
```

```
  float hume = dht.readHumidity();
  String tempeStr = String(tempe, 2);
  String humeStr = String(hume, 2);
JSONVar jsonObj2;
jsonObj2["numero_de_nodos"] = nodeCount;
jsonObj2["nodo"] = nodeNumber;
jsonObj2["temperatura"] = tempeStr;
jsonObj2["humedad"] = humeStr;
String jsonStr2 = JSON.stringify(jsonObj2);
SerialPort.println(jsonStr2);
delay(100);
});
  mesh.onNewConnection([](uint32_t nodeId) {
    Serial.printf("New Connection, nodeId = %u\n", nodeId);
  });
  mesh.onChangedConnections([]() {
    Serial.printf("Changed connections\n");
  });
  mesh.onNodeTimeAdjusted([](int32_t offset) {
    Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(),
offset);
  });
  userScheduler.addTask(taskSendMessage);
  taskSendMessage.enable();
}
```

Bucle Principal (Loop)

En el bucle principal del programa, se lleva a cabo la actualización periódica de la red de malla y otras operaciones relacionadas con el mantenimiento y funcionamiento del sistema.

- Actualización de la Red de Malla:

    - Se llama al método mesh.update() en cada iteración del bucle para permitir que la red de malla procese cualquier actividad pendiente, como el enrutamiento de mensajes y la gestión de conexiones.
    - Esta función es esencial para el correcto funcionamiento de la red de malla, ya que garantiza que la comunicación entre los nodos se mantenga actualizada y eficiente.
    - La actualización de la red de malla se realiza en cada iteración del bucle para garantizar una comunicación fluida y oportuna entre los nodos de la red

```
void loop() {
  mesh.update();
}
```

Función para Obtener Lecturas (`getReadings()`):

Esta función se encarga de recopilar los datos de temperatura y humedad del sensor DHT, así como el número de nodo, y luego formatear estos datos en un objeto JSON para su posterior procesamiento o transmisión. Aquí está el desglose de la función:

- Creación de un Objeto JSON:
  - Se declara una variable llamada `jsonReadings` del tipo `JSONVar`, que se utilizará para almacenar los datos recopilados.
  - Se asignan valores a las claves del objeto JSON utilizando la sintaxis de acceso a través del operador de indexación `[]`.
  - El número de nodo (`nodeNumber`) se asigna a la clave `"node"`.
  - La temperatura (`dht.readTemperature()`) se asigna a la clave `"temp"`.
  - La humedad (`dht.readHumidity()`) se asigna a la clave `"hum"`.

- Serialización del Objeto JSON:
  - Se utiliza la función `JSON.stringify(jsonReadings)` para convertir el objeto JSON (`jsonReadings`) en una cadena de texto JSON válida.
  - Esta cadena JSON se asigna a la variable `readings`.

- Retorno de las Lecturas Formateadas:
  - La función devuelve la cadena JSON (`readings`) que contiene los datos de temperatura, humedad y número de nodo en formato JSON.

```
String getReadings() {
  JSONVar jsonReadings;
  jsonReadings["node"] = nodeNumber;
  jsonReadings["temp"] = dht.readTemperature();
  jsonReadings["hum"] = dht.readHumidity();
  readings = JSON.stringify(jsonReadings);
  return readings;
}
```

Función para Enviar Mensajes (`sendMessage()`):

Esta función se encarga de enviar mensajes a través de la red mesh utilizando el objeto `painlessMesh`. Aquí está el desglose de la función:

- Obtención de Lecturas:
  - Se llama a la función `getReadings()` para obtener las lecturas de temperatura, humedad y número de nodo en formato JSON.
  - La cadena JSON resultante se almacena en la variable `msg`.
- Envío del Mensaje:
  - Se utiliza el método `sendBroadcast(msg)` del objeto `painlessMesh` para enviar el mensaje a todos los nodos de la red mesh.
  - Este método transmite el mensaje a través de la red mesh, lo que permite que otros nodos en la red lo reciban y lo procesen.
- Retardo:
  - Se agrega un pequeño retardo de 100 milisegundos (`delay(100)`) después de enviar el mensaje.
  - Este retardo puede ser útil para evitar la transmisión excesiva de mensajes y para permitir que otros nodos en la red tengan tiempo suficiente para procesar el mensaje recibido.

```
void sendMessage() {
  String msg = getReadings();
  mesh.sendBroadcast(msg);
  delay(100);
}
```

Código para los nodos esclavos y el nodo maestro-esclavo :

Este nodo comparte las mismas bibliotecas y constantes que el nodo maestro, con la única diferencia de la habilitación y uso de la comunicación serial. Además, cada nodo esclavo se identifica con un número único asignado, lo que simplifica su identificación y seguimiento por parte del usuario. Esta característica resulta fundamental para el funcionamiento adecuado del sistema, ya que facilita el establecimiento de una conexión y comunicación eficiente entre los nodos maestros y esclavos, simplificando así la implementación y gestión de la red en su totalidad. No es necesario explicar el código en detalle, ya que en esencia contiene las mismas funciones, con algunas excepciones con respecto al nodo maestro. Se puede encontrar este código en la sección de **anexos** de este documento.

Código para el ESP32 bridge

Inclusión de bibliotecas:

- ❖ WiFi.h: Esta biblioteca proporciona las funciones necesarias para conectarse y comunicarse a través de una red WiFi. Permite al dispositivo ESP32 (o ESP8266) establecer una conexión WiFi con un punto de acceso (router) y realizar operaciones como la configuración de la dirección IP, la conexión y desconexión de la red, entre otras.

- ❖ PubSubClient.h: Esta biblioteca permite la comunicación con un servidor MQTT (Message Queuing Telemetry Transport). MQTT es un protocolo de mensajería ligero y basado en suscripciones, utilizado comúnmente en aplicaciones de Internet de las cosas (IoT) para el intercambio de datos entre dispositivos y servidores. La biblioteca PubSubClient facilita la publicación y suscripción de mensajes MQTT desde un dispositivo Arduino o ESP32.

- ❖ HardwareSerial.h: Esta librería permite utilizar los puertos seriales de hardware disponibles en la placa. Proporciona funciones para la lectura y escritura de datos a través de estos puertos serie físicos. Esto es útil cuando se necesitan múltiples puertos serie o cuando se requiere una comunicación serie de alta velocidad y fiabilidad.

- ❖ ArduinoJson.h: Esta biblioteca facilita la manipulación y generación de datos en formato JSON (JavaScript Object Notation) en dispositivos Arduino y ESP32. Permite la creación, lectura, escritura y manipulación de objetos y arrays JSON, lo que resulta útil para el intercambio de datos estructurados entre dispositivos y servicios web en aplicaciones IoT, entre otros usos.

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <HardwareSerial.h>
#include <ArduinoJson.h>
```

Inicialización de Objetos y Configuración de Parámetros de Red

- HardwareSerial SerialPort(2): Esta línea inicializa un objeto `HardwareSerial` llamado `SerialPort` que utiliza el puerto serie 2 del microcontrolador para la comunicación serial. Esto permite la comunicación serial a través de un puerto específico.
- WiFiClient esp32Client: Aquí se crea un objeto `WiFiClient` llamado `esp32Client`, que se utilizará para manejar la conexión Wi-Fi del dispositivo ESP32.

- PubSubClient mqttClient(esp32Client): Se instancia un objeto `PubSubClient` llamado `mqttClient`, que se encargará de establecer la conexión MQTT utilizando el cliente Wi-Fi `esp32Client` como su cliente de red subyacente.
- const char* ssid = "wifi": Se define una cadena constante `ssid` que representa el nombre de la red Wi-Fi a la que se conectará el dispositivo ESP32.
- const char* password = "password": Aquí se define otra cadena constante `password` que representa la contraseña de la red Wi-Fi especificada en `ssid`.
- char* server = "broker… ": Se define una cadena de caracteres `server` que representa la dirección IP o el nombre de dominio del servidor MQTT al que se conectará el cliente MQTT.
- int port = 1883: Se especifica el número de puerto (1883 es el puerto predeterminado para MQTT) al que se conectará el cliente MQTT para comunicarse con el servidor MQTT.

```
HardwareSerial SerialPort(2);
WiFiClient esp32Client;
PubSubClient mqttClient(esp32Client);
const char* ssid = "wifi";
const char* password = "password";
char* server = "broker… ";
int port = 1883;
```

Inicialización y Conexión a la Red Wi-Fi

- void wifiInit() {`: Se define una función llamada `wifiInit()` que se encarga de inicializar y conectar el dispositivo ESP32 a una red Wi-Fi.
- Serial.print("Conectándose a "): Se imprime en el puerto serie un mensaje indicando que se está intentando conectar a una red Wi-Fi.
- Serial.println(ssid): Se imprime en el puerto serie el nombre de la red Wi-Fi a la que se intenta conectar, obtenido de la variable `ssid`.
- WiFi.begin(ssid, password): Se llama a la función 'WiFi.begin()' para iniciar el proceso de conexión a la red Wi-Fi especificada por `ssid` y `password`.
- while (WiFi.status() != WL_CONNECTED) {`: Se inicia un bucle while que espera hasta que el dispositivo se conecte correctamente a la red Wi-Fi. Mientras el estado de la conexión no sea `WL_CONNECTED`, el bucle imprime puntos en el puerto serie para indicar que el dispositivo está intentando conectarse.
- Serial.println(""): Se imprime una línea vacía en el puerto serie para indicar que se ha completado el proceso de conexión.
- Serial.println("Conectado a WiFi"): Se imprime un mensaje indicando que el dispositivo se ha conectado exitosamente a la red Wi-Fi.
- Serial.println("Dirección IP: "): Se imprime un mensaje indicando que se mostrará la dirección IP asignada al dispositivo.
- Serial.println(WiFi.localIP()): Se imprime en el puerto serie la dirección IP asignada al dispositivo ESP32 después de conectarse a la red Wi-Fi.

```
void wifiInit() {
    Serial.print("Conectándose a ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }

    Serial.println("");
    Serial.println("Conectado a WiFi");
    Serial.println("Dirección IP: ");
    Serial.println(WiFi.localIP());
}
```

Función de Callback para Mensajes MQTT

- void callback(char* topic, byte* payload, unsigned int length) {`: Se define una función de callback llamada `callback()` que se ejecuta cuando se recibe un mensaje MQTT.
- Serial.print("Mensaje recibido ["): Se imprime en el puerto serie un mensaje indicando que se ha recibido un mensaje MQTT.
- Serial.print(topic): Se imprime en el puerto serie el tema (topic) al que pertenece el mensaje recibido.
- Serial.print("] "): Se imprime un delimitador en el puerto serie.
- char payload_string[length + 1];`: Se declara un arreglo de caracteres llamado `payload_string` con longitud igual a `length + 1`, donde `length` es la longitud del payload del mensaje recibido.
- int resultI: Se declara una variable entera llamada `resultI`.
- memcpy(payload_string, payload, length): Se copia el contenido del payload del mensaje recibido al arreglo `payload_string`.
- payload_string[length] = '\0': Se agrega un carácter nulo al final del arreglo `payload_string` para asegurar que sea una cadena de caracteres válida.
- resultI = atoi(payload_string): Se convierte el payload del mensaje a un valor entero y se asigna a la variable `resultI`.
- Serial.println(): Se imprime una línea vacía en el puerto serie

```
void callback(char* topic, byte* payload, unsigned int length) {

    Serial.print("Mensaje recibido [");
    Serial.print(topic);
    Serial.print("] ");
    char payload_string[length + 1];
    int resultI;
    memcpy(payload_string, payload, length);
    payload_string[length] = '\0';
    resultI = atoi(payload_string);
    Serial.println();
}
```

Función de Reconexion al Servidor MQTT

- void reconnect() {: Se define una función llamada `reconnect()` que se encarga de intentar reconectarse al servidor MQTT.
- while (!mqttClient.connected()) {: Se ejecuta un bucle mientras el cliente MQTT no esté conectado al servidor.
- Serial.print("Intentando conectarse MQTT..."): Se imprime en el puerto serie un mensaje indicando que se está intentando establecer la conexión MQTT.
- if (mqttClient.connect("arduinoClient")) {: Se intenta establecer la conexión MQTT con un cliente identificado como "arduinoClient".
- Serial.println("Conectado"): Si la conexión se establece con éxito, se imprime en el puerto serie un mensaje indicando que se ha conectado al servidor.
- Serial.print("Fallo, rc="): Si la conexión falla, se imprime en el puerto serie un mensaje indicando el fallo y el código de estado de la conexión.
- Serial.print(mqttClient.state()): Se imprime el código de estado de la conexión MQTT.
- Serial.println(" intentar de nuevo en 5 segundos"): Se imprime un mensaje indicando que se volverá a intentar la conexión en 5 segundos.
- delay(5000): Se realiza una pausa de 5 segundos antes de intentar nuevamente la conexión.

```
void reconnect() {
    while (!mqttClient.connected()) {
        Serial.print("Intentando conectarse MQTT...");
        if (mqttClient.connect("arduinoClient")) {
            Serial.println("Conectado");
        } else {
            Serial.print("Fallo, rc=");
            Serial.print(mqttClient.state());
            Serial.println(" intentar de nuevo en 5 segundos");
            delay(5000);
        }
    }
}
```

Configuración Inicial del Sistema

- void setup() {: Se define la función de configuración inicial del sistema.
- Serial.begin(115200): Se inicializa la comunicación serial con una velocidad de transmisión de 115200 baudios.
- delay(10): Se realiza una pausa de 10 milisegundos para permitir que la comunicación serial se estabilice.
- wifiInit(): Se llama a la función `wifiInit()` para inicializar la conexión WiFi.
- mqttClient.setServer(server, port): Se configura el servidor MQTT al que se conectará el cliente MQTT con la dirección y el puerto especificados por las variables `server` y `port`, respectivamente.
- SerialPort.begin(15200, SERIAL_8N1, 16, 17): Se inicia la comunicación serial en el puerto 2 con una velocidad de transmisión de 15200 baudios y una configuración de 8 bits de datos, sin bit de paridad y 1 bit de parada. Los pines 16 y 17 se utilizan para la comunicación serial.

```
void setup() {
    Serial.begin(115200);
    delay(10);
    wifiInit();
    mqttClient.setServer(server, port);
    SerialPort.begin(15200, SERIAL_8N1, 16, 17);
}
```

Bucle Principal de Funcionamiento

- void loop() {: Se define el bucle principal de funcionamiento del programa.
- if (!mqttClient.connected()) { reconnect(); }: Se verifica si el cliente MQTT está conectado al servidor MQTT. En caso de que no esté conectado, se llama a la función `reconnect()` para intentar reconectar.
- mqttClient.loop(): Se llama al método `loop()` del cliente MQTT para mantener la conexión y procesar los mensajes entrantes.
- if (SerialPort.available()) { ... }: Se verifica si hay datos disponibles para leer en el puerto serie. Si hay datos disponibles, se procede a leerlos.
- String data = SerialPort.readStringUntil('\n'): Se lee una cadena de caracteres desde el puerto serie hasta encontrar el carácter de nueva línea ('\n').
- Serial.print("Datos recibidos: "); Serial.println(data): Se imprime en el monitor serial la cadena de caracteres recibida desde el puerto serie.
- delay(200): Se agrega un pequeño retraso de 200 milisegundos.
- mqttClient.publish("Nombre del proyecto en mqttx", data.c_str());`: Se publica la cadena de caracteres recibida como mensaje MQTT en el tema " Nombre del proyecto en mqttx ".
- delay(2000): Se agrega un retraso de 2000 milisegundos antes de que se vuelva a ejecutar el bucle principal.

```
void loop() {
    if (!mqttClient.connected()) {
        reconnect();
    }

    mqttClient.loop();

    if (SerialPort.available()) {
        String data = SerialPort.readStringUntil('\n');
        Serial.print("Datos recibidos: ");
        Serial.println(data);
        delay(200);
        mqttClient.publish("Nombre del proyecto en mqttx ", data.c_str());
    }
    delay(2000);
}
```

4.5. Configuración a través de la página MQTTX

MQTTX es una herramienta de código abierto y multiplataforma diseñada para facilitar el desarrollo, prueba y administración de aplicaciones MQTT. Permite la suscripción y publicación de mensajes MQTT, así como la visualización de mensajes recibidos en tiempo real.

Funcionalidades Avanzadas:

Guardado de Sesiones:

- MQTTX ofrece la posibilidad de guardar sesiones de conexión para facilitar la reconexión en futuras sesiones.

Configuración Avanzada:

- Explorar las opciones avanzadas de configuración para personalizar la experiencia de usuario, como el formato de visualización de mensajes y las teclas de acceso rápido.

Uso de Plugins:

- MQTTX admite plugins que proporcionan funcionalidades adicionales. Explorar la galería de plugins para encontrar extensiones útiles para un flujo de trabajo.

Integración con GitHub:

- MQTTX está alojado en GitHub, lo que significa que se puede contribuir al desarrollo del proyecto o informar de problemas que se encuentren

A continuación se presentan los pasos de configuración :

Uso desde la Página Web o Descarga e Instalación:

- Al visitar el sitio web oficial de MQTTX (https://mqttx.app/), descargar la versión adecuada para el sistema operativo. Si se prefiere, también se puede trabajar desde la página web sin necesidad de descargar la aplicación, como se muestra en la figura 21.

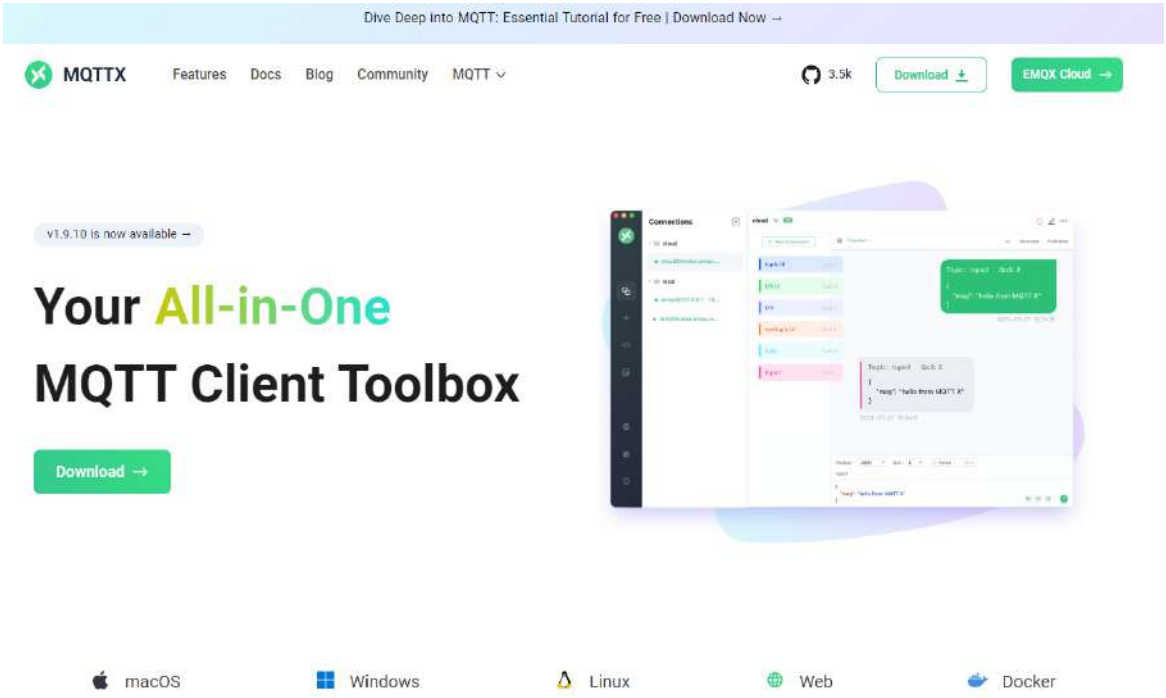


Figura 21. Pantalla principal de MQTTX

Una vez descargado el archivo de instalación, se siguen las instrucciones del instalador para completar la instalación en tu sistema operativo. Si se trabaja en la aplicación o en la página web, se debe crear una cuenta con un usuario y contraseña, además de crear el nombre del proyecto con contraseña, mostrado en la figura 22

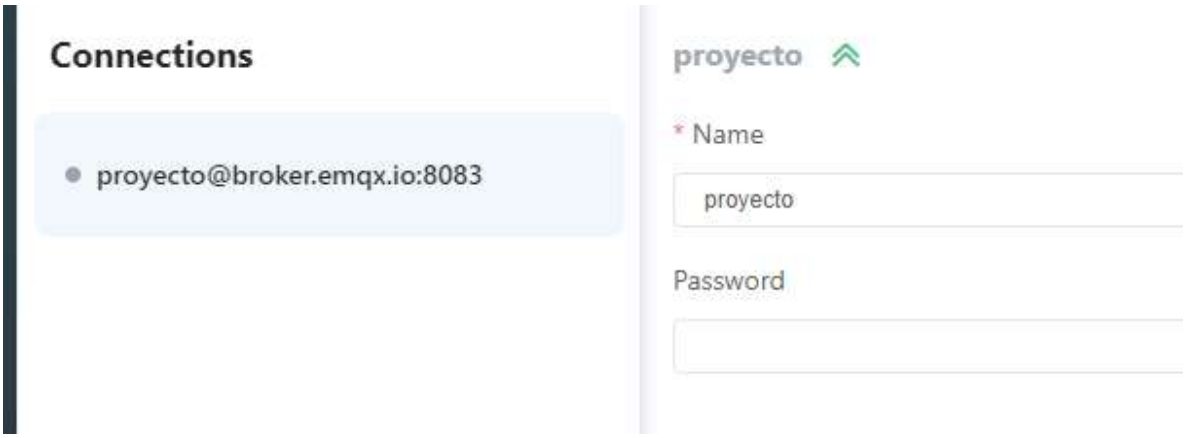


Figura 22. Creación del proyecto dentro de MQTTx

Configuración:

En la ventana principal de MQTTX, se pueden configurar los parámetros de conexión MQTT. Esto incluye el servidor MQTT, el puerto, y el nombre de usuario y contraseña, si es necesario. Esto se muestra en la figura 23.

Figura 23. Configuración de los parámetros de conexión

Conexión al Servidor:

Se debe hacer clic en el botón "Conectar" para establecer una conexión con el servidor MQTT especificado, como se indica en la figura 24.

Figura 24. Estableciendo conexión con el servidor

Suscripción a Temas (Topics):

En la pestaña "Subscription", se pueden agregar temas MQTT a los que se desea suscribir. Se introduce el nombre del tema y se hace clic en "Confirm", tal como se muestra en la Figura 25

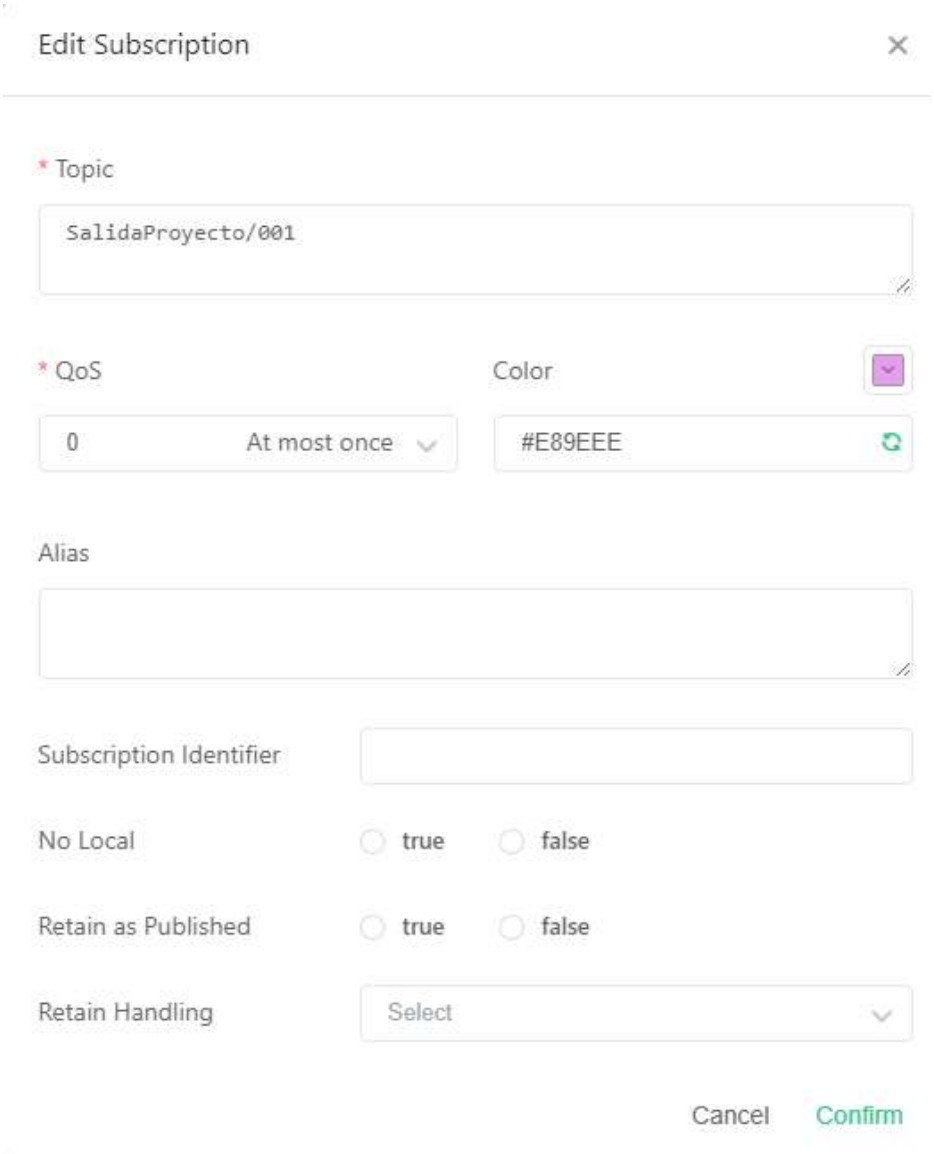


Figura 25. Configuración de la suscripción en MQTTx.

Capítulo 5

Resultados

En este capítulo, se presentan los resultados obtenidos después de la configuración de la red ad hoc y la programación de los nodos. Es de vital importancia realizar pruebas exhaustivas del sistema para verificar su correcto funcionamiento.

La red ad hoc constituye el elemento central de la infraestructura de comunicación en este proyecto. Esta red, implementada utilizando los módulos ESP32 y el método ESP Mesh, permite la interconexión de los nodos de manera dinámica y autónoma, sin depender de una infraestructura de red preexistente. Los datos recogidos de la temperatura y humedad de cada nodo son transmitidos a través de esta red ad hoc, asegurando una comunicación fluida y confiable entre los dispositivos.

En la Figura 26 se observa el nodo maestro y el ESP bridge. El nodo maestro cuenta con un sensor DHT integrado, el cual es responsable de medir la temperatura y la humedad del entorno. Este nodo maestro mantiene una conexión UART con el ESP bridge. El ESP bridge, a su vez, es el encargado de enviar la información recopilada de temperatura y humedad de cada nodo dentro de la red ad hoc.

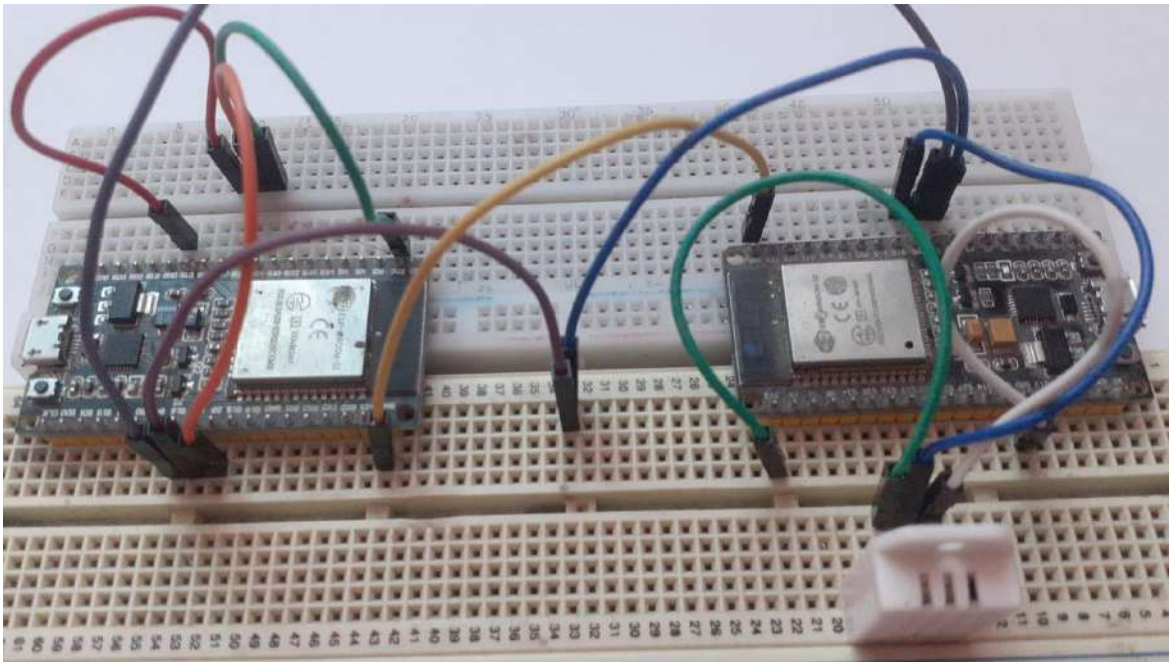


Figura 26. Nodo Maestro y ESP Bridge

En la Figura 27 se muestra un nodo esclavo o un nodo maestro-esclavo según sea necesario. En este caso, se utiliza el módulo ESP-WROOM-32, con un sensor dht integrado.

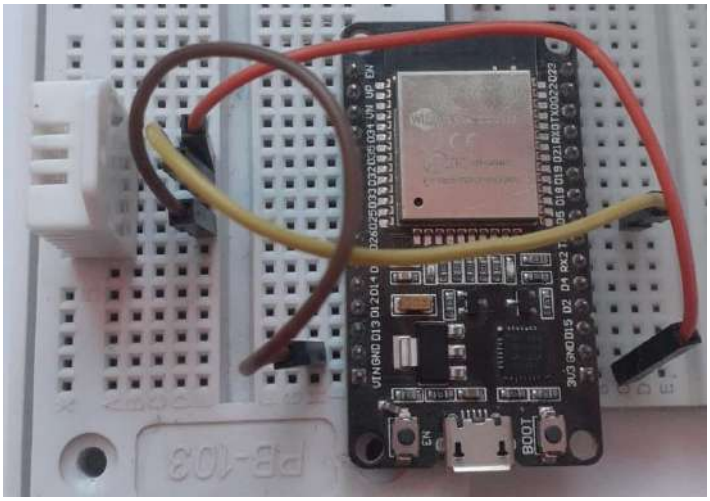


Figura 27. Módulo ESP-WROOM-32 como Nodo Esclavo o Maestro-Esclavo

De igual manera, en la Figura 28 se presenta el nodo esclavo o maestro-esclavo, el cual es una tarjeta desarrollada específicamente para este proyecto, como se detalló en la sección de desarrollo de la tesis. En esta figura, se puede observar la tarjeta PCB conectada con el sensor DHT y una batería de 3.7V con una capacidad de 000 mAh. Esta configuración permite que cada nodo esclavo recopile y transmita datos ambientales al nodo maestro, facilitando así la implementación y gestión de la red ad hoc en su totalidad.

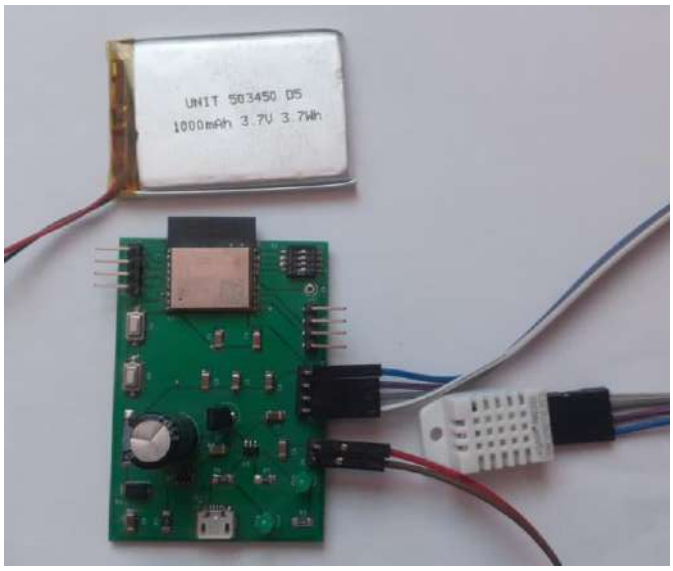


Figura 28. Nodo esclavo con la Tarjeta PCB desarrollada.

Una vez que los datos son enviados por los nodos, son recibidos y procesados por la plataforma MQTTx. Esta plataforma proporciona una interfaz intuitiva y versátil para la gestión de mensajes MQTT observados en la figura 26. Los datos son recibidos en formato JSON, lo que facilita su procesamiento y análisis posterior. Además, la plataforma MQTTx permite la visualización de los datos de manera clara y concisa, lo que facilita su interpretación y comprensión.

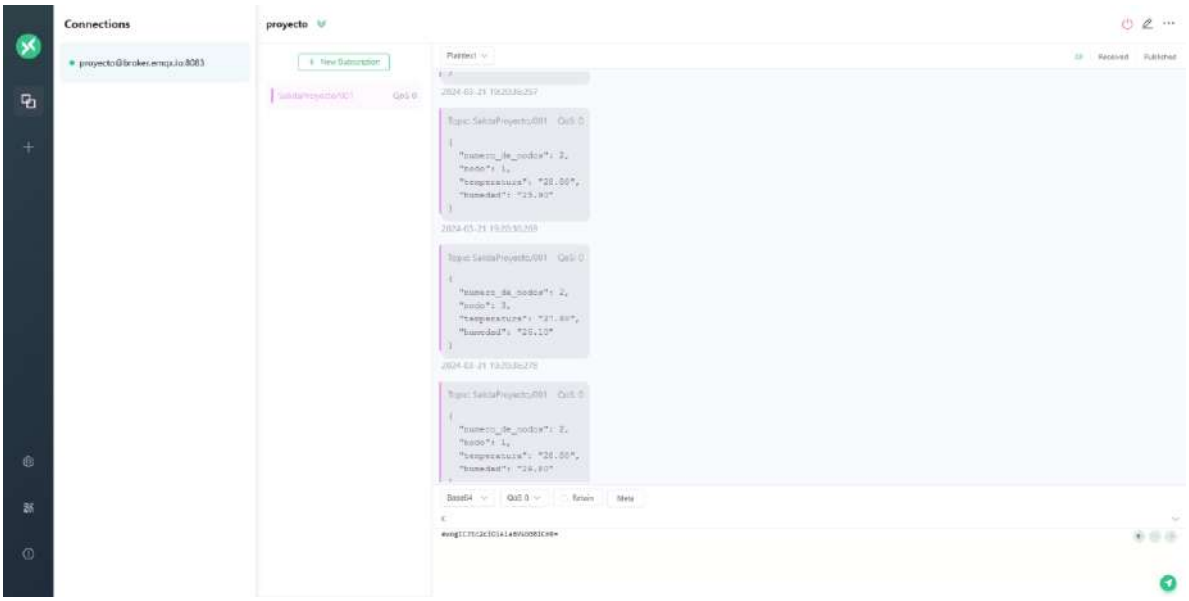


Figura 29. Interfaz de gestión de mensajes en la plataforma MQTTx.

La figura 27 muestra un ejemplo de los datos recibidos y procesados por la plataforma MQTTx. En este mensaje, se puede observar información detallada, como el número de nodos conectados a la red mesh, así como el número de nodo que envía la temperatura y la humedad en ese momento específico. Esta información es crucial para realizar un monitoreo efectivo del entorno y tomar decisiones informadas en tiempo real.

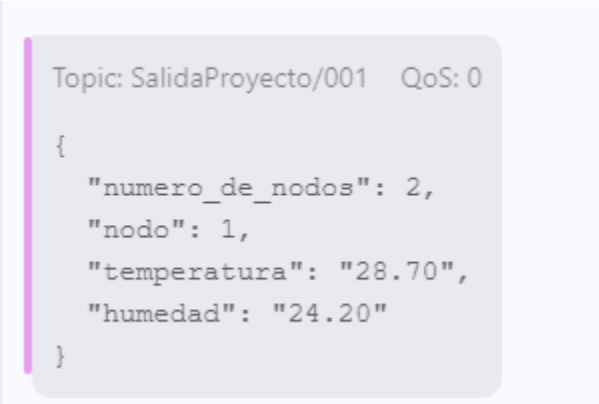


Figura 30. Datos recibidos y procesados por la plataforma MQTTx

Al añadir o retirar un nodo de la red ad hoc o al transcurrir cierto tiempo es posible recibir un mensaje similar tal y como se observa en la imagen 28, debido a la sincronización en los tiempos de envió de datos provenientes de todos los nodos y el tiempo en el que recibe la información la página mqttx, este error se soluciona al concluir el mensaje.

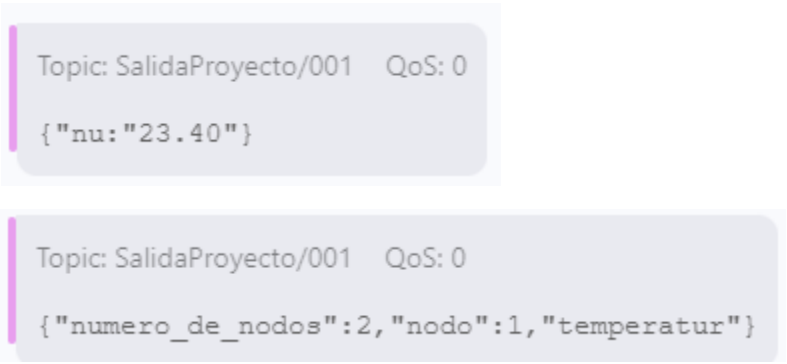


Figura 31. Errores de mensaje en la plataforma MQTTx

Capítulo 6

Conclusiones

Tras el desarrollo e implementación del sistema descrito en este trabajo, se pueden extraer varias conclusiones importantes. El enfoque principal del sistema se basó en la utilización del chip ESP32 y la creación de una red ad hoc mediante ESP Mesh para establecer la comunicación entre los nodos.

- ✓ Ventajas de las redes ad hoc con ESP Mesh:

El uso de redes ad hoc con ESP Mesh presenta varias ventajas significativas en comparación con las redes convencionales. Al establecer una red ad hoc, se elimina la necesidad de una infraestructura de red preexistente, lo que la hace ideal para entornos donde no se dispone de una infraestructura establecida o en situaciones de emergencia. Las redes ad hoc con ESP Mesh son autoorganizadas y autónomas, lo que permite una rápida configuración y adaptabilidad a cambios en la topología de la red.

- ✓ Potencial del chip ESP32:

El chip ESP32 utilizado en este proyecto ha demostrado ser una solución para la implementación de redes ad hoc con ESP Mesh. Con su capacidad de conectividad Wi-Fi y Bluetooth, así como su amplio conjunto de funciones y periféricos, el ESP32 ofrece un rendimiento y una amplia gama de aplicaciones. Además, su bajo consumo de energía lo hace adecuado para aplicaciones de monitoreo y control a largo plazo.

- ✓ Importancia de la estructura de red ad hoc con ESP Mesh:

La elección de la estructura de red ad hoc con ESP Mesh demostró ser acertada, ya que permitió una comunicación eficiente y confiable entre los nodos de la red. Esto se traduce en una transmisión de datos fluida y una capacidad de respuesta óptima. La utilización de los sensores de temperatura y humedad DHT11 también contribuyó a la obtención de datos precisos y confiables para el monitoreo ambiental.

- ✓ Ventajas de la plataforma MQTTx y el protocolo MQTT:

La integración de la plataforma MQTTx y el protocolo MQTT resultó ser altamente conveniente para la visualización y análisis de los datos recopilados. La interfaz gráfica proporcionada por MQTTx permite una fácil comprensión de los resultados, brindando una visión clara y accesible del monitoreo de temperatura y humedad en tiempo real. Además, el protocolo MQTT asegura una comunicación confiable y segura entre los nodos y la plataforma.

Mejoras futuras del proyecto:

A medida que el proyecto avance, existen diversas mejoras que pueden implementarse para optimizar el sistema y ampliar sus capacidades. Algunas posibles mejoras incluyen:

- Implementación de algoritmos de enrutamiento más avanzados: Para mejorar la comunicación en la red ad hoc, se pueden explorar algoritmos de enrutamiento más sofisticados y adaptativos.
- Integración de otros tipos de sensores: Además de los sensores de temperatura y humedad, se pueden incorporar otros tipos de sensores, como sensores de calidad del aire, de luminosidad o de presencia, para obtener una visión más completa del entorno monitoreado.
- Desarrollo de funciones de control y actuación: En lugar de solo monitorear los datos ambientales, se puede considerar la integración de actuadores para permitir el control remoto de dispositivos, como sistemas de climatización o sistemas de riego automatizados, en base a los datos recopilados.
- Implementación de medidas de seguridad adicionales: Para garantizar la integridad y la confidencialidad de los datos transmitidos, se pueden aplicar técnicas de seguridad adicionales, como el cifrado de datos o la autenticación de nodos en la red.

Con la implementación de este sistema basado en el chip ESP32 y la red ad hoc con ESP Mesh, se logró un monitoreo de la temperatura y humedad. Las conclusiones obtenidas indican que el sistema es capaz de recopilar datos precisos y confiables, permitiendo tomar decisiones informadas y realizar ajustes cuando sea necesario. La estructura de red ad hoc con ESP Mesh demostró ser flexible y escalable, lo que brinda la posibilidad de expandir el sistema en el futuro para adaptarse a nuevas necesidades y áreas de monitoreo. La integración con la plataforma MQTTx y el uso del protocolo MQTT ofrecen una interfaz amigable y una comunicación segura. Además, se mencionaron posibles mejoras como la implementación de algoritmos de enrutamiento avanzados, la integración de otros sensores, el desarrollo de funciones de control y actuación y la implementación de medidas de seguridad. Estas mejoras permitirían ampliar las capacidades del sistema y mejorar su eficiencia en la recolección y análisis de datos ambientales.

Referencias

[1] M. Kumari, D. V. Kumar y M. D. Chandra Sati, «Establishing a Wireless-Local-Area-Network (WLAN) Connectivity between Multiple Nodes using ESP-Mesh Network Topology for IoT Applications, » International Research Journal of Engineering and Technology (IRJET), vol. 7, nº 8, 2020.

[2] I.V. Haro Vilaña, "Implementación de un sistema de monitoreo y control automático de riego para invernaderos mediante tecnología lora con ESP32". tesis, Universidad Tecnológica Israel, Quito, 2019.

[3] K. Suseenthiran, A. Shukur Ja'afar, K. Wei Heng, M. Z. Abidin Abd Aziz, A. Awang Md Isa y S. Huzaimah Husin, «Indoor positioning utilizing bluetooth low energy RSSI on LoRa system,» Indonesian Journal of Electrical Engineering and Computer Science, vol. 23, nº 2, p. 937, 2021.

[4] C. R. P. V. Pravalika, «Internet of Things Based Home Monitoring and Device Control Using Esp32,» International Journal of Recent Technology and Engineering (IJRTE), vol. 8, p. 62, 2019.

[5] A. Umarov, M. Kunelbayev, M. Satymbekov, G. Turken, A. Bagila, K. Imanzhanova y L. Duisembayeva, «Microclimate Monitoring System for a Home Greenhouse as Part of ESP32,» *TEST Engineering & Management,* vol. 82, p. 10, 2020.

[6] J. González Álvarez, Posicionamiento de una cabina de ascensor mediante un ESP32 y sensores, Castellón de la Plana: Universitat Jaume, 2021.

[7] M. . P. J. V., B. J. Olivieri de Souza, T. de Souza Lamenza y M. E. , «PRACTICAL CHALLENGES AND PITFALLS OF BLUETOOTH MESH DATA COLLECTION EXPERIMENTS WITH ESP-32 MICROCONTROLLERS,» Computer Science Networking and Internet Architecture, vol. 1, nº 22, p. 12, 2022.

[8] J. I. Vega Luna, V. N. Tapia Vargas, M. A. Lagos Acosta, G. Salgado Guzmán, F. J. Sánchez Rangel y J. F. Cosme Aceves, «Sistema Detector de Alacranes Usando IOT con BLE,» Congreso Internacional de Ingeniería Electrónica, vol. 42, p. 118, 2020.

[9] M. García Gómez, «Sistema de control de casa inteligente utilizando el protocolo ESP-NOW: Smart-Home control system using the ESP-NOW protocol», INCAING, vol. 4, n.º 24, pp. 1–6, jun. 2021.

[10] J. Peris Martinez, Sistema de Monitorización Inalámbrica de Temperatura Mediante Sensor de Infrarrojos y Microcontrolador ESP32, Valencia : Universidad Politécnica de Valencia, 2020.

[11] Devu Jayaraj, P. Aneesh R , S. Sooraj P , S. Arya Lekshmi y P. Deenath , «Smart Agro: IOT Based Rice Plant Health Monitoring System,» International Journal of Innovations & Implementations in Engineering, vol. 1, nº 19, 2020.

[12] Hermansyah, Kasim y. Lin Karmila , «Solar Panel Remote Monitoring and Control System on Miniature Weather Stations Based on Web Server and ESP32,» International Journal of Recent Technology and Applied Science, vol. 2, nº 1, p. 24, 2020.

[13] M. F. Boné Andrade, J. D. Rodriguez Vizuete, A. P. Mora Olivero y S. M. Sosa Calero, "Sistema de Monitoreo de temperatura y humedad en el hogar aplicando IoT de bajo costo", G-ner@ndo, vol. 3, n.º 2, p. 88, 2022

[14] I. Firman Ashari, M. Darma Satria y I. Mohamad, «Parking System Optimization Based on IoT using Face and Vehicle Plat Recognition via Amazon Web Service and ESP-32 CAM (Case Study: Institut Teknologi Sumatera),» Computer Engineering and Applications, vol. 11, nº 2, p. 17, 2022.

[15] A. Muhammad, H. G. Hafiz Ali, B. A. Muhammad y S. Moeed , «IoT Based Real Time Warehouse Monitoring using Sparkfun ESP8266 Thing Dev and Cayenne,» University of Swabi Journal (USJ), vol. 2, nº 2, p. 20, 2018.

[16] M. Á. PÉREZ DÍAZ, Invernadero inteligente basado en ESP-MESH y AWS Smart greenhouse using ESP-MESH and AWS, Madrid: UNIVERSIDAD COMPLUTENSE DE MADRID, 2020.

[17] F. A. Arturo Asistimbay, "Diseño y desarrollo de un prototipo inalámbrico para un sistema de casilleros universitarios utilizando dispositivos iot y ubidots", Proyecto de titulación, Universidad de Guayaquil, Ecuador, 2021.

[18] A. M. Yachimba Guaitapi, "Diseño e implementación de un prototipo para un sistema de medición, análisis y purificador de gases contaminantes en el aire utilizando Arduino y ubidots ot", Trabajo de titulación, Universidad Politécnica Salesiana, Guayaquil, 2022.

[19] D. Patiño Vélez, «Diseño de un dispositivo wearable para el monitoreo de la oxigenación y ritmo cardiaco,» Sociedad Mexicana de Ingeniería Biomédica, vol. 7, nº 1, pp. 485-492, 2020

[20] F. Aji Purnomo, N. Maulana Yoeseph, S. A. Tri Bawono y R. Hartono, «Development of air temperature and soil moisture monitoring systems with LoRA technology,» Journal of Physics: International Conference on Physics and Its Applications, vol. 1825, nº 012029, 2020.

[21] A. Álvarez Carulla, «Comunicación de un módulo ESP32 con Ubidots mediante MQTT,» Creative Commons, 2021.

[22] S. N. Pineida Parra, ANÁLISIS DE VULNERABILIDADES EN LA SEGURIDAD DE LA RED WLAN CON IOT EN RESIDENCIAS QUE UTILIZAN DOMÓTICA Y CONTROL REMOTO, Quito: Universidad Politécnica Salesiana Ecuador, 2020.

[23] N. Cameron, Electronics Projects with the ESP8266 and ESP32, Edinburg: Apress, 2021.

[24] R. Santos y S. Santos, MycroPython ESP32 and ESP8266, 2020.

[25] J. Andreu, Servicios en red. Madrid: Editex, 2010.

[26] Espressif Systems. (2021). "ESP32 Datasheet." Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_es.pdf.

[27] Espressif Systems. (2014). "ESP8266EX Datasheet." Disponible en: https://www.espressif.com/sites/default/files/documentation/0aesp8266ex_datasheet_es.pdf

[28] Raspberry Pi. (2021). "Raspberry Pi Pico Datasheet." Disponible en: https://datasheets.raspberrypi.org/pico/pico-datasheet-es.pdf.

[29] Arduino. (2021). "Arduino UNO Datasheet." Disponible en: https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-02-TH.pdf.

[30] STMicroelectronics. (2021). "STM32F103xC, STM32F103xD, STM32F103xE Datasheet." Disponible en: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf.

[31] Adafruit Industries, "DHT11 Temperature & Humidity Sensor Datasheet". Disponible en: https://www.mouser.com/datasheet/2/758/DHT11-Technical-DataSheetTranslatedVersion-1143054.pdf

[32] Adafruit Industries, "DHT22 Temperature & Humidity Sensor Datasheet". Disponible en: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[33] Maxim Integrated, "DS18B20 Datasheet". Disponible en: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[34] Bosch Sensortec, "BME280 Datasheet". Disponible en: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf

[35] Sensirion, "SHT3x-DIS Humidity and Temperature Sensor Datasheet". Disponible en: https://html.alldatasheet.com/html-pdf/897975/ETC2/SHT31/394/4/SHT31.html

[36] N. Fernando, et al., "Lithium Polymer Battery Characteristics and Performance Evaluation for Wireless Sensor Nodes," IEEE Transactions on Power Electronics, vol. 28, no. 12, pp. 5763-5774, Dec. 2013.

[37] S. D. Silva, et al., "Battery Management System for LiPo Batteries in a Wireless Sensor Node," IEEE Transactions on Industrial Electronics, vol. 64, no. 7, pp. 5612-5621, Jul. 2017.

[38] A. Sharma, et al., "Performance Analysis of Li-Ion Batteries for Wireless Sensor Nodes in IoT Applications," IEEE Transactions on Sustainable Energy, vol. 9, no. 1, pp. 412-421, Jan. 2018.

[39]| J. C. Lee, et al., "Design and Evaluation of Battery Management System for Li-Ion Battery Packs in IoT Devices," IEEE Transactions on Power Electronics, vol. 31, no. 5, pp. 3794-3804, May 2016.

[40] M. J. Kim, et al., "NiMH Battery Modeling and Performance Evaluation in Low-Power IoT Devices," IEEE Transactions on Power Electronics, vol. 29, no. 8, pp. 4398-4407, Aug. 2014.

[41] S. R. Gupta, et al., "A Comparative Study of NiMH and Li-ion Batteries for Energy Harvesting Wireless Sensor Nodes," IEEE Sensors Journal, vol. 18, no. 6, pp. 2471-2480, Mar. 2018.

[42] R. S. Patel, et al., "NiCd Battery Characteristics and Their Suitability for ESP32-Based IoT Applications," IEEE Transactions on Power Electronics, vol. 33, no. 9, pp. 7765-7775, Sep. 2018.

[43] A. K. Das, et al., "Evaluation of NiCd Battery Performance in Low-Power Wireless Sensor Nodes," IEEE Sensors Letters, vol. 2, no. 3, 2500504, Sep. 2018.

Cronograma de actividades

Seminario de Tesis I

	Sep	Oct	Nov	Dic
Planteamiento del capítulo 1	x			
Redacción de los objetivos	x			
Investigación sobre trabajos relacionados		x		
Inv. Redes ad hoc			x	
Inv. Microcontrolador ESP32			x	
Inv. Creación de redes ad hoc con el chip ESP32				x
Inv. Protocolos de comunicación para establecer comunicación entre un chip ESP32 y una página web				x

Seminario de Tesis II

	Ene	Feb	Mar	Abr
Evaluación de materiales para el desarrollo del proyecto	x			
Análisis de los materiales utilizados en el desarrollo del proyecto	x			
Diseño del circuito eléctrico		x		
Diseño de la tarjeta PCB			x	
Soldar los componentes en la PCB				x

Seminario de Tesis III

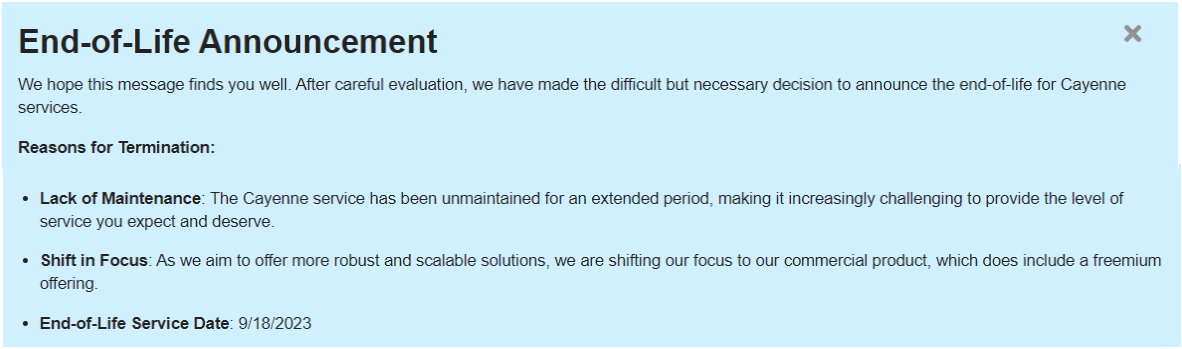
	May	Jun	Jul	Ago
Programacion de los chips ESP32-C3-WROOM-02	x			
Configuración de la página MQTTx		x		
Resultados		x		
Correcciones del proyecto			x	
Concluciones			x	
Correcciones del documeno				x
Presentacion Final				x

Codigo para los nodos esclavos y maestro-esclavo

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include "painlessMesh.h"
#include <Arduino_JSON.h>
#define DHT_PIN 5
#define DHT_TYPE DHT22
#define MESH_PREFIX "RNTMESH"
#define MESH_PASSWORD "MESHpassword"
#define MESH_PORT 5555
int nodeNumber = 4;
String readings;
Scheduler userScheduler;
painlessMesh mesh;
void sendMessage();
String getReadings();
Task taskSendMessage(TASK_SECOND * 5, TASK_FOREVER, &sendMessage);
DHT dht(DHT_PIN, DHT_TYPE);
void setup() {
  Serial.begin(115200);
  dht.begin();
  mesh.setDebugMsgTypes(ERROR | STARTUP);
  mesh.init(MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT);
  mesh.onReceive([](uint32_t from, String &msg) {
    Serial.printf("Received from %u msg=%s\n", from, msg.c_str());
    JSONVar myObject = JSON.parse(msg.c_str());
    int node = myObject["node"];
    double temp = myObject["temp"];
    double hum = myObject["hum"];
    Serial.print("Node: ");
    Serial.println(node);
    Serial.print("Temperature: ");
    Serial.print(temp);
    Serial.println(" C");
    Serial.print("Humidity: ");
    Serial.print(hum);
    Serial.println(" %");
  });
  mesh.onNewConnection([](uint32_t nodeId) {
    Serial.printf("New Connection, nodeId = %u\n", nodeId);
  });
  mesh.onChangedConnections([]() {
    Serial.printf("Changed connections\n");
  });
```

```
  mesh.onNodeTimeAdjusted([](int32_t offset) {
    Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(),
offset);
  });
  userScheduler.addTask(taskSendMessage);
  taskSendMessage.enable();
}
void loop() {
  mesh.update();
}
String getReadings() {
  JSONVar jsonReadings;
  jsonReadings["node"] = nodeNumber;
  jsonReadings["temp"] = dht.readTemperature();
  jsonReadings["hum"] = dht.readHumidity();
  readings = JSON.stringify(jsonReadings);
  return readings;
}
void sendMessage() {
  String msg = getReadings();
  mesh.sendBroadcast(msg);
}
```

Inicialmente, teníamos la intención de trabajar en la plataforma Cayenne MyDevices. Sin embargo, durante el desarrollo del proyecto, la página decidió cesar sus operaciones, deshabilitando así su acceso. Como resultado, al intentar acceder a la página oficial (https://community.mydevices.com), nos encontramos con el siguiente mensaje informativo.



End-of-Life Announcement

We hope this message finds you well. After careful evaluation, we have made the difficult but necessary decision to announce the end-of-life for Cayenne services.

Reasons for Termination:

- **Lack of Maintenance**: The Cayenne service has been unmaintained for an extended period, making it increasingly challenging to provide the level of service you expect and deserve.
- **Shift in Focus**: As we aim to offer more robust and scalable solutions, we are shifting our focus to our commercial product, which does include a freemium offering.
- **End-of-Life Service Date**: 9/18/2023