ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS

CREACIÓN DE UN PROTOTIPO DE MONITOREO DE TEMPERATURA DEL AIRE UTILIZANDO EQUIPOS DE DESARROLLO DE BAJO COSTO Y COMUNICACIÓN LORAWAN.

Implementación física de un prototipo de monitoreo de temperatura del aire utilizando equipos de desarrollo de bajo costo y comunicación LoRaWAN.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN CIENCIAS DE LA COMPUTACIÓN

VICKY NAHOMI BONILLA NAZARENO

vicky.bonilla@epn.edu.ec

DIRECTOR: Ph.D SANG GUUN YOO.

sang.yoo@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, VICKY NAHOMI BONILLA NAZARENO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

VICKY NAHOMI BONILLA NAZARENO

Certifico que el presente trabajo de integración curricular fue desarrollado por VICKY NAHOMI BONILLA NAZARENO, bajo mi supervisión.

SANG GUUN YOO, Ph.D.
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

VICKY NAHOMI BONILLA NAZARENO

SANG GUUN YOO

BRANDON MARTÍN CAMPOVERDE RODRÍGUEZ

DEDICATORIA

A mi madre y mi padre que me enseñaron a seguir adelante a pesar de todo.

AGRADECIMIENTO

A mi madre, Guerdy. Por impulsarme a seguir adelante y por todo el amor y sacrificio que le han significado mis cuidados y el apoyarme en mis estudios. Espero poder retribuir, aunque sea una parte de todo lo que ha hecho por mí.

A mi padre, Víctor. Por su amor, por creer en mí incluso cuando yo mismo no lo hacía y por ser mi cimiento para comenzar de nuevo.

A mis hermanos: Josué porque admiro su capacidad que me motiva a estudiar e intentar ser como él; Jason por su cariño y lecciones; y Adrián por su ejemplo de disciplina, por haber estado pendiente de mí siempre y ser alguien con quien sé que puedo contar.

Al Profe Sang, por haberme abierto las puertas del laboratorio de investigación Smart Lab y darme su voto de confianza con proyectos de investigación, además de su guía y apoyo para cumplir mis objetivos.

A Brandon, por su apoyo incondicional. No pude haber elegido a nadie mejor como compañero no solo de este trabajo sino para seguir adelante esta etapa de mi vida.

A mi familia y amigos que me dan ánimos para hacer lo que me proponga.

ÍNDICE DE CONTENIDO

RESUMEN

LoRaWAN ha ganado importancia en el área de comunicaciones IoT a nivel mundial debido a que es una tecnología económica, de largo alcance y bajo costo energético que permite realizar una gran variedad de aplicaciones. Sin embargo, en Ecuador no se ha extendido su adopción. El presente trabajo muestra el diseño y creación de un prototipo de bajo costo para la medición de temperatura ambiental utilizando LoRaWAN dentro de la Escuela Politécnica Nacional. Para ello se utilizó la metodología PPDIOO propuesta por Cisco, dividiendo el proceso en 6 etapas. Preparación, planificación, diseño, implementación, operación y optimización. El prototipo se implementó ensamblando el dispositivo final, luego conectando este al Gateway LoRaWAN. Posteriormente se procesaron los datos dentro del Gateway haciendo uso de Node-RED. Luego se estableció un servidor MQTT que permitió el acceso a los datos por parte de un cliente MQTT. Tras poner la red en operación, se optimizó y se realizaron pruebas de cobertura, consumo energético y calidad de la señal. Entre los principales resultados se encuentra una cobertura de 2.66km que cubre el perímetro de la Escuela politécnica Nacional. Además, el tiempo de funcionamiento, enviando datos cada 3 minutos con una batería de 1000mAh, alcanza las 20 horas. Para la calidad de la señal se obtuvo una RSSI promedio de -28dBm y una SNR de 9dB. Todo esto con un costo que no supera los $70 por dispositivo, logrando un avance en el ámbito de la supervisión ambiental inalámbrica de bajo coste en el entorno de la Escuela Politécnica Nacional y expandiendo las posibilidades de futuras aplicaciones utilizando la tecnología LoRaWAN.

PALABRAS CLAVE: LoRaWAN, sensores, monitoreo de temperatura.

ABSTRACT

LoRaWAN has gained significance in the field of IoT communications globally due to its cost-effectiveness, long-range capabilities, and low energy consumption, enabling a wide range of applications. However, its adoption in Ecuador is not widespread. The purpose of this study is the design and creation of an affordable prototype for ambient temperature measurement using LoRaWAN within the environment of Escuela Politécnica Nacional. The PPDIOO methodology proposed by Cisco was employed, dividing the process into six stages: Preparation, Planning, Design, Implementation, Operation, and Optimization. To implement the prototype, the end-device was assembled, followed by connecting it to the LoRaWAN Gateway. Subsequently, data processing occurred within the Gateway using Node-RED. A MQTT server was established to grant data access to an MQTT client. After deploying the network, optimization was conducted along with tests for coverage, energy consumption, and signal quality. The results include a coverage of 2.66 km encompassing the perimeter of the Escuela Politécnica Nacional. Moreover, the operational time, sending data every 3 minutes with a 1000mAh battery, extends to 20 hours. Signal quality revealed an average RSSI of -28dBm and a SNR of 9dB. All of this is achieved at a cost not exceeding $70 per device, representing a significant advancement in the realm of economical wireless environmental monitoring within the setting of Escuela Politécnica Nacional's environment while also broadening prospects for future applications using LoRaWAN technology.

KEYWORDS: LoRaWAN, sensors, temperature monitoring.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El componente desarrollado es un prototipo que permite la medición, transmisión y almacenamiento temporal de datos de temperatura ambiental. El prototipo se compone de diversas partes fundamentales, cada una de ellas desempeñando un papel crucial en el funcionamiento eficiente del sistema. La arquitectura del prototipo puede apreciarse en la Figura 1, donde los elementos correspondientes al componente desarrollado se pueden ver dentro del recuadro con línea entrecortada.

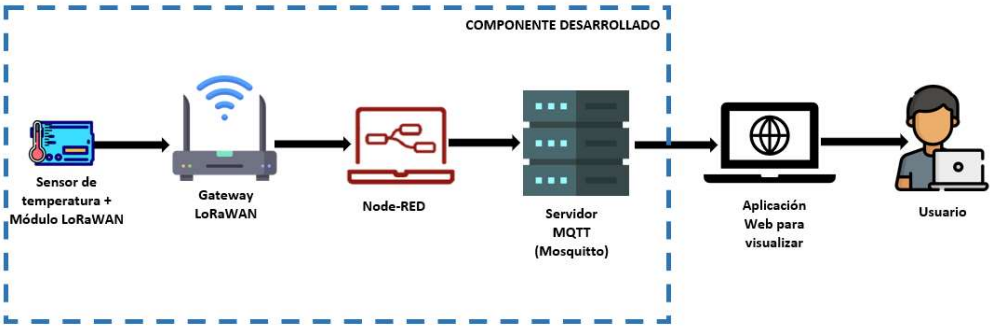

Figura 1. Arquitectura del prototipo de monitoreo de temperatura del aire.

En primer lugar, se contó con un sensor de la marca DHT11 que permitió receptar la temperatura ambiental relativa en el rango de 0 a 50°C.

Luego, el sensor de temperatura ambiental DHT11 fue conectado a un módulo de LoRaWAN de modelo WiFi LoRa 32 V1 el cual es el responsable de la transmisión inalámbrica de los datos recolectados por el sensor hacia el Gateway LoRaWAN. El módulo WiFi LoRa 32 V1 es una placa de desarrollo IoT que puede ser empleada en diversas aplicaciones IoT. La programación del sensor en conjunto con el módulo de LoRaWAN fue realizada en el IDE de Arduino debido a su capacidad de añadir las librerías de código abierto necesarias para este componente.

En cuanto al Gateway LoRaWAN, se optó por utilizar el modelo MultiTech Conduit® IP67 Base Station. Este Gateway es una solución que proporciona conectividad de red de área amplia, de bajo consumo y bajo consumo de energía en aplicaciones IoT.

Posteriormente, se hizo uso de Node-RED, el cual permitió la recepción y decodificación de datos además de la extracción de información relevante de los sensores, incluyendo sus identificadores.

Finalmente, la información proporcionada por Node-RED fue extraída haciendo uso del servidor Mosquitto, que es un servidor de mensajería eficiente y ligero que permite procesar los mensajes recibidos del Gateway LoRaWAN gracias al uso del protocolo MQTT.

Al combinar las tecnologías de sensores, IoT, comunicación LoRaWAN y el servidor MQTT, el sistema ofrece una solución completa y efectiva para la recolección y posterior análisis de datos de temperatura. Este componente sienta las bases para futuras implementaciones más amplias en el ámbito de la gestión ambiental en la Escuela Politécnica Nacional, representando así un importante avance en el campo de la monitorización ambiental inalámbrica y de bajo costo dentro de la Institución.

1.1 Objetivo general

Desarrollar un prototipo de sistema de monitoreo de la temperatura ambiental utilizando LoRaWAN y sensores de bajo costo que permita el acceso a los datos a través de un servidor MQTT.

1.2 Objetivos específicos

1. Generar una arquitectura del sistema de monitoreo de la temperatura ambiental.
2. Desarrollar un prototipo del componente físico del sistema de monitoreo de la temperatura ambiental.
3. Desarrollar la infraestructura de comunicación utilizando LoRaWAN que permita tomar los datos obtenidos mediante el uso de un servidor MQTT.
4. Realizar pruebas del componente físico y de la infraestructura de comunicación del sistema de monitoreo de la temperatura ambiental.

1.3 Alcance

En este componente, se diseñó y desarrolló el prototipo del sistema de medición de temperatura ambiental definiendo los requerimientos necesarios del sistema conforme la arquitectura propuesta. También, se desarrolló el componente físico del sistema, es decir el dispositivo de monitoreo. Posteriormente se configuró la infraestructura de comunicación LoRaWAN. Luego, se configuró un servidor MQTT para enviar los datos obtenidos. Finalmente, se desarrollaron las pruebas que permitieron comprobar el funcionamiento del dispositivo y su conexión con la red LoRaWAN.

1.4 Marco teórico

En esta sección, se exponen los conceptos más importantes necesarios para comprender el desarrollo y funcionamiento del prototipo de monitoreo de temperatura del aire utilizando equipos de desarrollo de bajo costo y comunicación LoRaWAN.

Tecnologías LPWAN

Las tecnologías LPWAN (Low Power Wide Area Network) son aquellas que como su nombre indica hacen uso un bajo consumo energético y poseen una amplia cobertura. Existen algunas tecnologías LPWAN de entre las que destacan LoRaWAN, Sigfox, NB-IoT y LTE-M.

LoRaWAN

LoRaWAN es actualmente una de las tecnologías LPWAN más utilizadas en IoT [1]. Esta tecnología funciona en frecuencias no licenciadas y utiliza una modulación CSS (chirp spread specturm), utiliza las bandas 868 en Europa y 915MHz en Norteamérica junto con Sigfox. Está estandarizada por la LoRa-Alliance y se caracteriza por sus bajos niveles de ruido y dificultad para interceptar mensajes. El factor de esparcimiento (SF por sus siglas en inglés) va desde el SF7 hasta el SF12 para adaptarse a la velocidad de datos (300bps-50kbps). Mientras más alto sea el SF, se tendrá mayor rango de comunicación, pero menor velocidad de datos [2].

Los dispositivos finales pueden pertenecer a una de tres clases: A, B o C. Estas clases determinan cómo se gestionan las comunicaciones entre los dispositivos finales y la red.

La clase A está compuesta por dispositivos de comunicación bidireccional asimétrico, donde después de enviar los datos al servidor de red (uplink), el dispositivo abre dos periodos cortos para esperar posibles mensajes de retorno desde el servidor (downlink). Una vez terminado el periodo de espera, el dispositivo vuelve a un estado de bajo consumo [2]. Este modo de comunicación permite un bajo consumo energético. Un esquema de la clase A puede verse en la Figura 2.

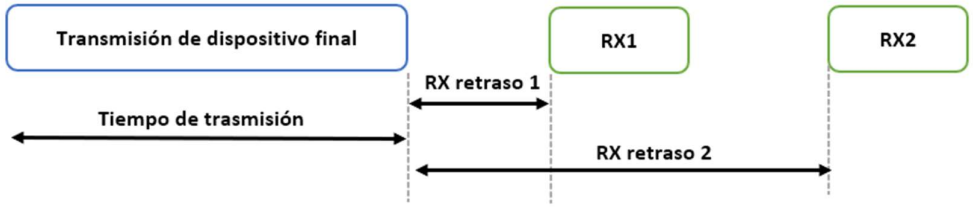

Figura 2. Esquema de funcionamiento de dispositivos de clase A.

La clase B además de recibir mensajes aleatorios, puede recibir mensajes en intervalos programados gracias a una señal sincronizada con el tiempo de la estación base. Esto hace que se consuma más energía que si operara como en la clase A.

En la clase C los dispositivos pueden recibir mensajes continuamente, excepto cuando estos transmiten mensajes. Esta clase consume más energía que las dos mencionadas anteriormente [3]

Sigfox

Es una tecnología propietaria LPWAN que ofrece conectividad punto a punto. Esta tecnología utiliza modulación de BPSK (Binary Phase-Shift Keying), además utiliza frecuencias no licenciadas como la de 433 MHz en Asia y comparte las bandas de 868 MHz en Europa y de 915MHz en Norteamérica con Sigfox [3]. El ancho de banda en el que Sigfox opera (100Hz) permite experimentar una comunicación con bajos niveles de ruido, dando como resultado un bajo consumo de energía y una sensibilidad más alta de las antenas a un bajo costo. Sin embargo, el estrecho ancho de banda limita el rendimiento (throughput) a 100bps.

NB-IoT

Esta es una tecnología que puede coexistir en LTE o GSM además de trabajar tanto en frecuencias licenciadas como no licenciadas. Ocupa un ancho de banda de 200KHz y dentro de las frecuencias licenciadas tiene tres modos de operación que son [2]:

- Operaciones Stand alone: Que ocupa frecuencias GSM ya en uso
- Operaciones Guard Band: Se implementa dentro de la banda de guarda de LTE.
- Operaciones In-band: Se lo usa dentro de la portadora LTE.

Un esquema de estas operaciones puede verse en la Figura 3.

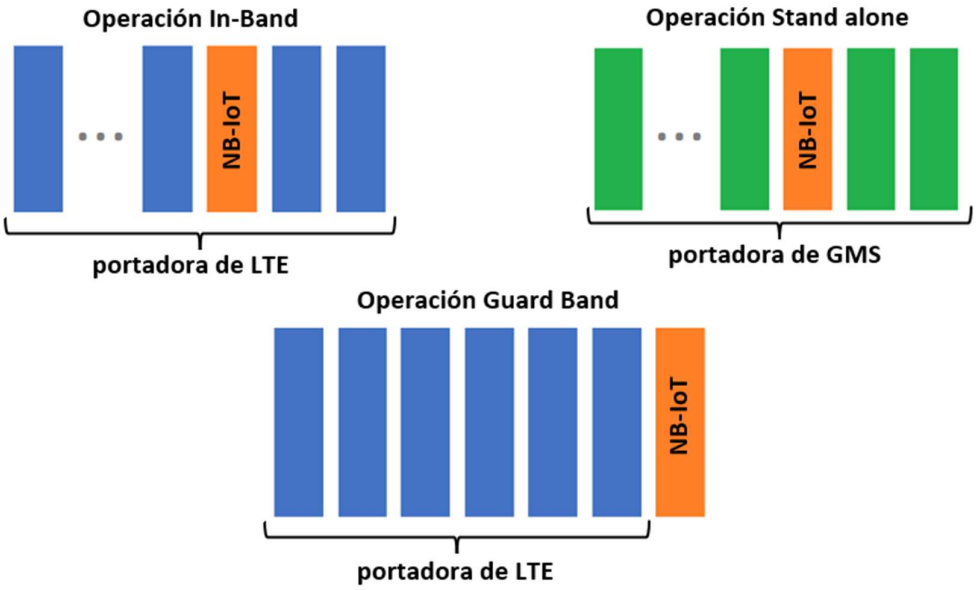

Figura 3. Esquema de operaciones NB-IoT.

Las tecnologías LPWAN mencionadas cuentan con diversos campos de aplicación y además son frecuentemente usadas dentro de aplicaciones IoT. Un resumen de las características principales se puede apreciar en Tabla 1

Tabla 1. Características de las principales tecnologías LPWAN [4].

	LoRaWAN	**Sigfox**	**NB-IoT**
Capacidad de dispositivos	10000	1000000	40000
Modulación	FSS/CSS	D-BPSK	OFDMA
Datos de downlink (bytes)	14	8	125
Datos de uplink (bytes)	51	12	125
Velocidad de datos (bps)	1760	100	50000
Banda de frecuencia (MHz)	868	868	868
Espectro (kHz)	1175	200	180
Ancho de banda (kHz)	125	0.1	15

Node-RED

Node-RED es una herramienta de programación desarrollada por IBM y ahora es parte de la OpenJS Foundation. Esta herramienta utiliza la programación asada en flujos que es una forma de describir el comportamiento de una red de “nodos”. Cada nodo tiene el propósito de recibir datos, hacer alguna operación sobre esos datos y posteriormente transmitirlos. Node-RED está compuesto de un entorno de ejecución basado en Node.js que apunta a un navegador web para acceder al editor de flujos [5].

MQTT

MQTT (OASIS Message Queuing Telemetry Transport) es un protocolo de transporte de mensajería abierto basado en el modelo de publicación y suscripción, creado con el propósito de facilitar el intercambio eficaz de información en tiempo real entre dispositivos móviles y sensores [6]. Mosquitto es una implementación de MQTT que está destinada a ser utilizada en todas las situaciones en las que se requiere mensajería liviana, especialmente en dispositivos limitados por recursos. El proyecto Mosquitto es miembro de la Fundación Eclipse [7].

MQTT consta de varios componentes: Un mensaje de aplicación que es la información llevada a través de la red para la aplicación. Cuando los Mensajes de Aplicación se transmiten mediante MQTT, están asociados a una Calidad de Servicio y un Nombre de tema [8]. Un Bróker es un servidor central dentro de una red MQTT que puede publicar mensajes de aplicación que podrían interesar a otros clientes. Los clientes deberán conectarse a este bróker para poder suscribirse y eliminar su suscripción a temas. Los temas dentro del protocolo MQTT son una cadena de identificación para los mensajes.

Existen tipos de clientes distintos, un cliente puede ser publicador (envía mensajes) como suscriptor (recibe mensajes). La conexión establecida entre el cliente y el bróker (sesión MQTT) puede mantenerse después de una desconexión temporal o puede durar sólo cuando esté conectada. El protocolo se ejecuta sobre TCP/IP, o sobre otros protocolos de red que proporcionan conexiones bidireccionales ordenadas y sin pérdida [8]

Un esquema del protocolo MQTT se puede apreciar en la Figura 4.

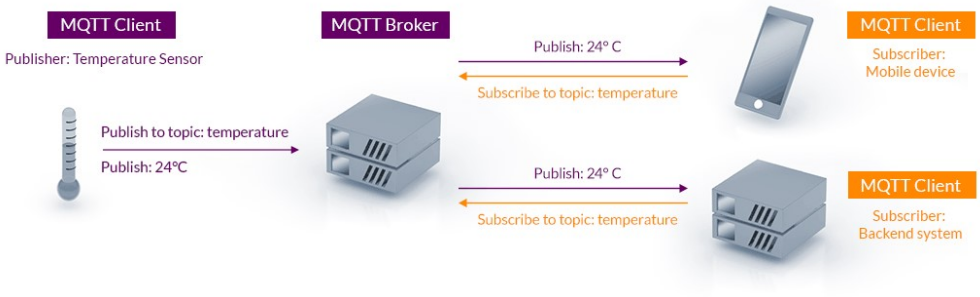

Figura 4. Esquema del protocolo MQTT [9].

2 METODOLOGÍA

En esta sección, se detalla la metodología utilizada para el desarrollo del componente físico del prototipo de monitoreo de temperatura del aire. La metodología seleccionada fue PPDIOO debido que define un ciclo de vida continuo que permite la mejora del prototipo, una entrega de soluciones a tiempo y reduce el coste total de la propiedad de la red [10].

2.1 Metodología PPDIOO

PPDIOO significa Preparar, Planificar, Diseñar, Implementar, Operar y Optimizar. Esta es una metodología de Cisco que plantea un ciclo de vida continuo de las redes como se puede ver en la Figura 5. A continuación, se detalla cada una de las respectivas etapas

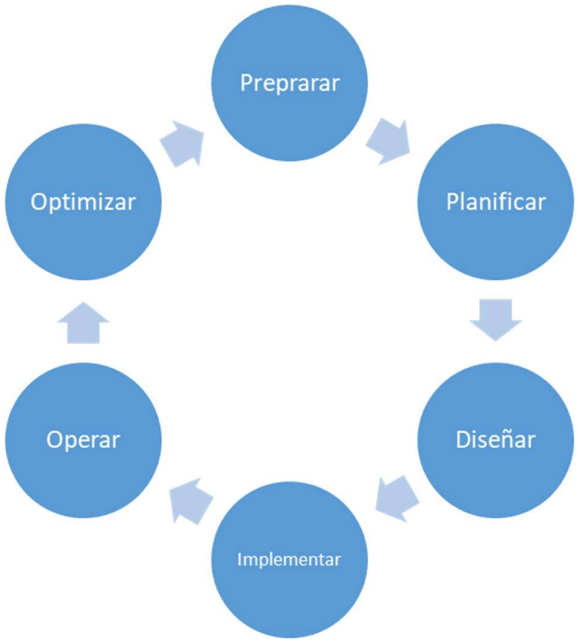

Figura 5. Ciclo de metodología PPDIOO.

- Preparar: Involucra establecer los requerimientos y proponer una arquitectura de alto nivel que permita identificar las tecnologías más útiles que se acoplen a la solución pensada [11].
- Planificar: Se deben identificar los requerimientos iniciales de la red, caracterizar los recursos existentes y se realiza un plan que permite conocer las tareas que habrán de realizarse en la fase de implementación [11].
- Diseñar: En función de las etapas anteriores se genera un diseño técnico de la red incluyendo las configuraciones y especificaciones de seguridad [11].

- Implementar: Se lleva a cabo la implementación física y lógica de la red en conformidad con el diseño. Se realizan configuraciones de dispositivos, conexiones y pruebas [11].
- Operar: en esta etapa se supervisa la implementación y se gestionan los problemas que puedan existir con la implementación [11].
- Optimizar: Con los datos obtenidos en la etapa de operación se puede tener una administración proactiva de la red, resolviendo los problemas encontrados previamente [11]. En el caso de que existiese una gran cantidad de problemas con la red establecida, se podría rediseñar toda la red conforme se puede ver en el esquema de la Figura 5.

En las etapas de preparación, planeación y diseño se tiene una ventaja principal que es la reducción de eventuales rediseños. Por otra parte, en la etapa de implementación se tienen beneficios como la disminución del tiempo de implementación y la mitigación de riesgos [12] .

2.2 Desarrollo

Preparación

En esta etapa, se analizaron las características de los equipos y las tecnologías necesarias para desarrollar el componente con la arquitectura planteada (ver Figura 6). En dicha arquitectura, se puede apreciar la necesidad de un sensor de temperatura, un módulo de LoRaWAN, un Gateway LoRaWAN, herramienta Node-RED y el uso de MQTT como servidor.

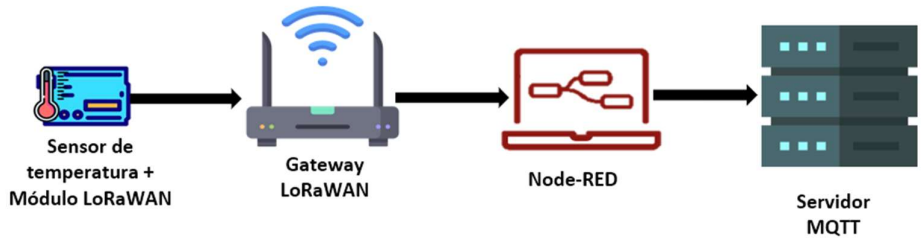

Figura 6. Arquitectura del componente desarrollado.

Sin embargo, además de estos requerimientos iniciales tomados de la arquitectura del prototipo, se vio la necesidad de programar el sensor. Para esto, se utilizó el Arduino IDE con sus respectivas librerías. Finalmente, otro elemento que no se puede ver dentro de la

arquitectura, pero resultó necesario fue una fuente de energía para abastecer al módulo de LoRaWAN

Planificación

En esta sección, a partir de las tecnologías y características descritas en la etapa de preparación, se realizó el plan que determinó las tareas correspondientes a la etapa de implementación. Este plan se elaboró en base a lo descrito en las recomendaciones de Cisco como se muestra en la Tabla 2 [11].

Tabla 2. Plan de tareas a realizar.

Fase	Fecha propuesta de culminación	Descripción	Implementación
Fase 1	30/05/2023	Recepción de datos de temperatura del entorno	Implementación de la Fase 1
Paso 1		Ensamblaje del dispositivo final	
Paso 2		Programación del sensor de temperatura	
Paso 3		Verificación de datos de temperatura	
Fase 2	04/07/2023	Recepción de datos por parte del Gateway	Implementación de la Fase 2
Paso 1		Configuración de Gateway	
Paso 2		Configuración del dispositivo final modificando la programación del paso 2 de la fase 1.	
Paso 3		Verificación de conexión entre Gateway y dispositivo final	
Fase 3	30/07/2023	Procesamiento de datos en Node-RED y envío hacia MQTT	Implementación de la Fase 3
Paso 1		Programación de flujo de datos en Node-RED	
Paso 2		Configuración de servidor MQTT	
Paso 3		Suscripción Node-RED - MQTT	
Paso 4		Verificación de suscripción	

Esta etapa también permitió conocer de manera más específica los elementos de hardware y software que facilitaron el desarrollo del componente.

Sensor de temperatura DHT

Debido a su costo y facilidad de conexión, se utilizó un sensor de la marca DHT11 que permite receptar la temperatura ambiental relativa en el rango de 0 a 50°C con una precisión de ±2.0°C [13] (ver Figura 7).

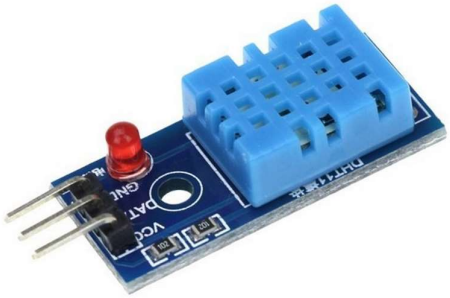

Figura 7. Sensor DHT11 [13].

Módulo Heltec WiFi LoRa 32 V1

Dado que la tecnología LPWAN permite la transmisión de datos sin agotar rápidamente sus energías, es una opción atractiva en el campo de IoT. Entre las tecnologías LPWAN, se optó por utilizar LoRaWAN debido a su bajo costo. Al ser una tecnología no propietaria, tiene una mayor flexibilidad en la adquisición de dispositivos y por consiguiente la posibilidad de reducir costos. Además, es una alternativa costo eficiente tanto en áreas rurales y urbanas cuando la densidad de dispositivos y su actividad es baja [4]. Este módulo cuenta con un procesador ESP32, un wifi de 2,4GHz y una antena LoRa con una frecuencia de 915MHz. Además, Heltec ofrece una librería Arduino para el protocolo LoRaWAN + ESP32 [14]. Un diagrama del módulo puede apreciarse en la Figura 8.

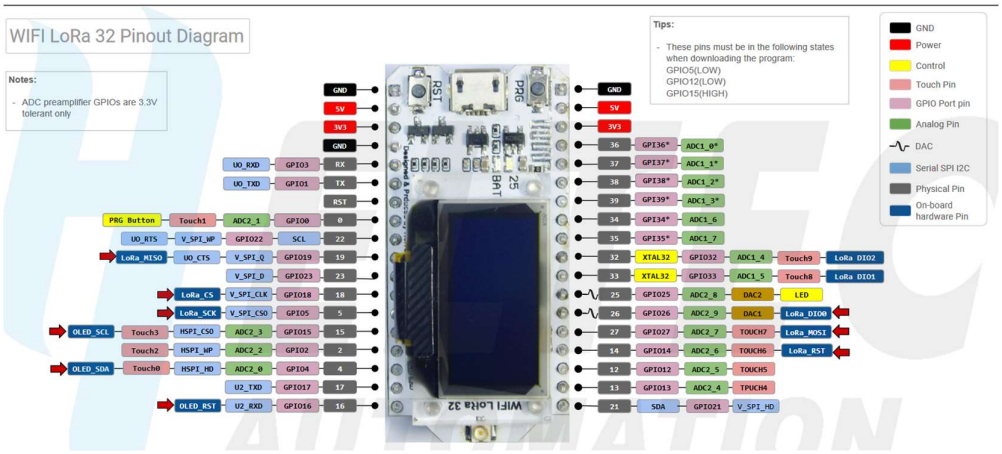

Figura 8. Diagrama del módulo Heltec WiFi LoRa 32 V1 [15].

Power Bank Hoco

El Power Bank seleccionado fue el B20-10000 Mige Mobile de la marca Hoco, el cual se utilizó debido a su costo y a que el material del cual está hecho es retardante de llamas, lo que proporciona seguridad física para el módulo de LoRaWAN en caso de algún fallo dentro de la batería. Además, posee un display del porcentaje batería que resultó útil para realizar pruebas de consumo energético. En la Tabla 3, se pueden apreciar sus principales características.

Tabla 3. Características de Power Bank Hoco [16].

Capacidad	10000mAh
Batería	Batería de litio 18650
Tamaño	94mm65mm24mm (LWH).
Entrada	5V/2A
Salida	Doble USB: 5V/2.1A
Material	Material retardante de llama ABS+PC, patrón de superficie de fibra de carbono.
Display	Indicador digital de batería.

Gateway MultiTech Conduit® IP67 Base Station

Este es un componente con el que ya se contaba de antemano dentro del Smart Lab de la Escuela Politécnica Nacional. Este dispositivo ofrece un enfoque de consumo energético reducido y es utilizado en aplicaciones IoT. Una descripción de algunas de las características de hardware más relevantes se puede ver en la Tabla 4.

Tabla 4. Características del Gateway MultiTech Conduit® IP67 Base Station [17].

Hardware	
Procesador y memoria	ARM9 con 32-Bit ARM & 16-Bit Thumb instruction sets 400 MHz, 16K Data Cache, 16K Instruction Cache, 128X16 MB DDR RAM, 256 MB Flash Memory
Modo de Red	Público o privado
Especificaciones LoRa	
Frequency de Banda	915 MHz
Canal	US915
Capac dad de canal	8 (half-duplex)
LoRa Power Output	27 dBm maximum ERP (before LoRa antenna)
Connectors	
E-NET RJ45	Ethernet jack (10/100 port) (PoE)
Antennas	Cellular, GPS, LoRa: N-Type Female
Perfiles	
Perfil de red	Soporta Clase A, B y C. Se usa case A
Perfil de dispositivo	LW102-OTA-US915: LoRaWAN v1.0.2, OTAA, Plan del Canal US915

Arduino IDE 2.1.0

En base a las características del sensor y placa seleccionada, así como por el objetivo del presente trabajo de utilizar dispositivos de bajos costos, se utilizó el lenguaje de programación de Arduino dentro del Arduino IDE 2.1.0 para la elaboración del componente. Esta esta es una herramienta flexible y de código abierto que permitió el uso de diferentes librerías [18]. Para la elaboración del componente se utilizaron las siguientes librerías:

- Adafruit Unified Sensor Driver 1.1.9: Esta librería se utilizó para estandarizar la variable de temperatura dentro de este prototipo. La librería simplifica todos los datos a un único tipo de 'sensors_event_t' y establece unidades SI estándar específicas para cada familia de sensores. Esto garantiza que los mismos tipos de sensores proporcionen valores comparables con otros sensores similares lo que permite abordar los desafíos y dificultades relacionados con la disponibilidad de sensores y la reutilización de código [19].
- Arduino LoRaWAN library: Se utilizó debido a que admite implementaciones de Clase A y Clase C de LoRaWAN que operan en las bandas EU-868, AS-923, US-

915 y AU-915. Permite funcionalidades como: Enviar paquetes (uplink), teniendo en cuenta el ciclo de trabajo activo. Recibir paquetes (downlink) en la ventana RX1 (US-915) y en la ventana RX2. Activación por aire (OTAA / unión) (US-915). Esta librería también permite mapear los pines I/O y se ha probado que funciona con dispositivos ESP32 como lo es el módulo Heltec WiFi LoRa 32 V1 [20].

- Heltec_ESP32 Library: Se utilizó esta librería ya que permite manejar el ESP32 al proporcionar funciones y utilidades definidas para el microcontrolador. Esta librería es desarrollada por Heltec Automation y es compatible con el módulo seleccionado para el componente a desarrollarse [21].
- Arduino-LMIC library ("MCCI LoRaWAN LMIC Library"): Se utilizó esta librería ya que permite utilizar los transceptores SX1272, SX1276 y módulos compatibles, enviar paquetes (uplink), teniendo en cuenta el ciclo de trabajo, cifrado y verificación de integridad de mensajes, configuración de frecuencias personalizadas y tasas de datos, activación por aire (OTAA / unión), recibir paquetes (downlink) en las ventanas RX1 y RX2. Procesamiento de comandos MAC [22].

Node-RED v0.15.3

Esta herramienta se encuentra integrada en el Gateway MultiTech Conduit® IP67 y se la ocupó debido a su utilidad para gestionar los datos recibidos desde el sensor DHT11. Al poseer capacidades de decodificado y filtrado de datos que permite captar la información relevante para el componente desarrollado [23]. La versión utilizada de Node-RED para este prototipo es la versión v0.15.3 y funciona en el puerto 1880.

Mosquitto como bróker y cliente MQTT

Mosquitto es un bróker de mensajes de código abierto que puede implementar las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT [24]. Este sistema es liviano y se adapta a su uso en una amplia gama de dispositivos, desde computadoras de placa única con baja potencia hasta servidores completos, por lo que es ideal para la implementación junto con LoRaWAN debido a su bajo uso de recursos. La versión utilizada para el desarrollo de este prototipo es la 2.0.15. y se utilizó en el puerto 8883.

Diseño

En esta etapa, se diseñó la arquitectura lógica de la red, teniendo como referencia base los elementos de la etapa de diseño como se puede ver en la Figura 9.

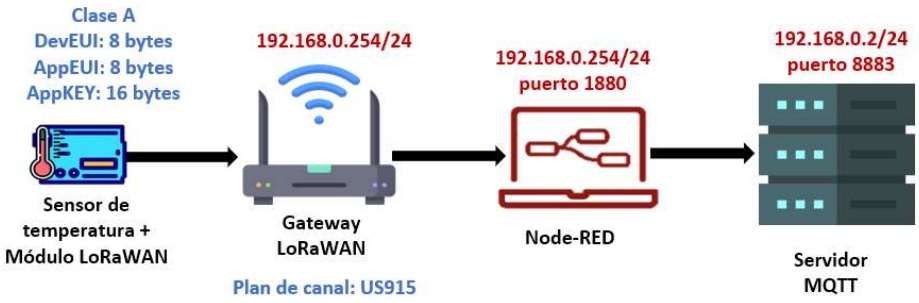

Figura 9. Arquitectura lógica de la red.

De este esquema, se obtuvieron los requerimientos que se dieron satisfacción en la fase de implementación. Se determinó que el dispositivo final debía contar con las características de la Tabla 5.

Tabla 5. Características del dispositivo final.

Perfil de la Red	LW102-OTA-US915
Perfil del dispositivo	DEFAULT-CLASS-A
DevEUI (8 bytes)	4e-4f-4d-42-52-45-31-32
AppEUI (8 bytes)	70-b3-d5-7e-d0-02-d4-e7
AppKey (16 bytes)	63ae0b8b8f5cc637e645939ec52c12

Donde se puede apreciar que el dispositivo pertenece a la clase A de LoRaWAN, dado que esta es una clase de bajo consumo energético. Las configuraciones del Gateway se hicieron en base las características ya descritas del dispositivo final. El plan de canal fue configurado como US915 y se trabajó en la clase A.

Otro punto para tratar fue la seguridad dentro de LoRaWAN, para lo cual se utilizó las medidas de protección por defecto de su protocolo de comunicación. En esta tecnología, la autenticación mutua se establece entre un dispositivo final LoRaWAN y la red LoRaWAN como parte del procedimiento de incorporación a la red. Cada dispositivo LoRaWAN se personaliza con una clave única AES de 128 bits (AppKey) y un identificador globalmente único (DevEUI basado en EUI-64), los cuales se utilizan durante el proceso de autenticación del dispositivo. También usa la llave de sesión de red (NwkSkey) y la llave de aplicación de sesión (AppSKey) [25].

En este sentido, también es importante indicar que LoRaWAN provee dos tipos posibles de autenticación i.e., APB (Activación By Personalization) y OTAA (Over-The-Air Activation).

En la autenticación APB, se suministran al dispositivo final dos claves de sesión (llamadas Clave de Sesión de Aplicación - AppSKey, y Clave de Sesión de Red - NwkSKey). Del lado de la red, la NwkSKey se suministra en el Servidor de Red y la AppSKey se suministra en el Servidor de Aplicaciones [26].

Por otro lado, en la autenticación OTAA, tanto el dispositivo final como la red tienen conocimiento de la AppKey. Luego se derivan dos claves de sesión, una para proporcionar protección de integridad y cifrado de los comandos MAC de LoRaWAN y la carga útil de la aplicación (NwkSKey), y otra para el cifrado de extremo a extremo de la carga útil de la aplicación (AppSKey). La NwkSKey se distribuye a la red LoRaWAN para demostrar/verificar la autenticidad e integridad de los paquetes. La AppSKey se distribuye al servidor de aplicaciones para cifrar/descifrar la carga útil de la aplicación. Tanto la AppKey como la AppSKey pueden ocultarse al operador de red para que no pueda descifrar las cargas útiles de la aplicación [25].

Para el desarrollo del prototipo, se seleccionó la autenticación OTAA debido a que permite un intercambio seguro de claves de sesión entre el dispositivo y la red y ofrece una solución más dinámica. Este esquema de seguridad de LoRaWAN 1.0.x se puede apreciarse en la Figura 10.

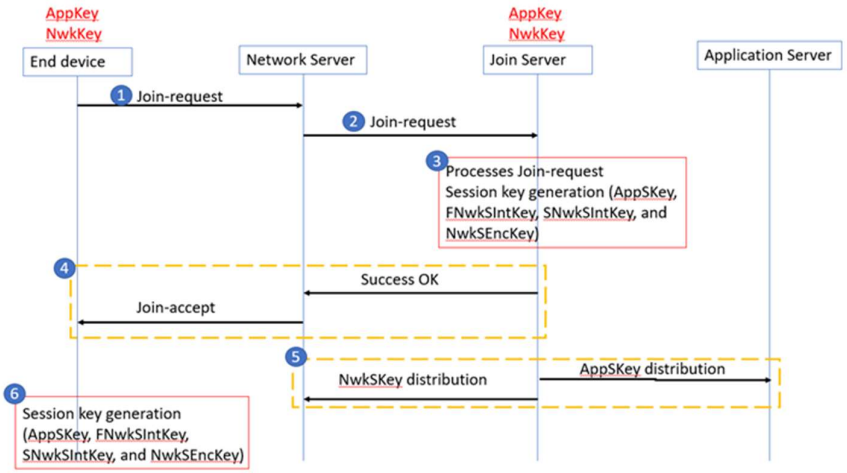

Figura 10. Esquema de seguridad de LoRaWAN 1.0.x [27].

Implementación

En base a la información desarrollada en etapa de diseño, el dispositivo que hace las veces de nodo final fue integrado a la red, para lo cual se realizaron las tres fases propuestas en la etapa de planificación.

Fase 1: Recepción de datos de temperatura del entorno

- **Paso 1: Ensamblaje del dispositivo final**

Como se aprecia en la Figura 11, se armó el dispositivo final que permitió medir la temperatura en base a las especificaciones de la etapa de planificación. Para realizar las conexiones del módulo LoRaWAN con el sensor se tomó como referencia el diagrama de la Figura 8. El cable negro corresponde a la conexión a tierra (GND), el cable amarillo es el voltaje de 3.3 voltios (3V3) y el cable café es el cable de datos que fue conectado en el pin 13. Para la alimentación eléctrica se utilizó el Power Bank de la Marca Hoco. El resultado puede verse en la Figura 11.

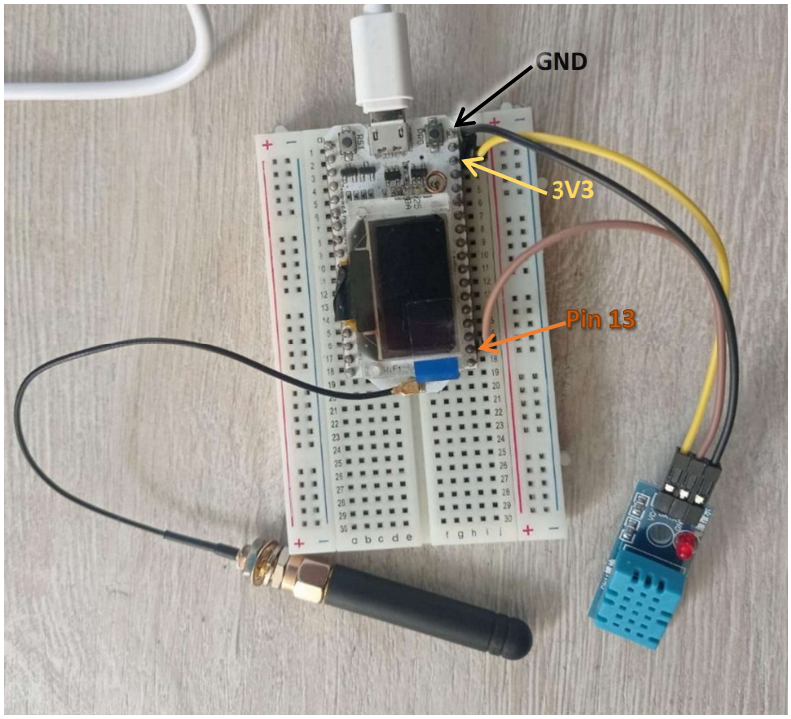

Figura 11. Prototipo versión 1.

- **Paso 2: Programación del sensor de temperatura**

Se utilizó un programa de prueba que permitiría comprobar el funcionamiento del sensor. El programa consistió de una función **loop()** que permitió tomar la temperatura, definiendo la variable de temperatura y utilizando la función **delay()** para tomar registros cada 2 segundos (ver Figura 12).

temperatura_Heltec.ino

```
#include <DHT.h>

#define DHTPIN 13     // Pin donde está conectado el sensor DHT11
#define DHTTYPE DHT11 // Tipo de sensor DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600); // Iniciar el monitor serial
  dht.begin();        // Iniciar el sensor DHT11
}

void loop() {
  // Leer la temperatura y la humedad del sensor
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Mostrar los valores en el monitor serial
  Serial.print("Temperatura: ");
  Serial.print(temperature);
  Serial.print(" °C\t");

  delay(2000); // Esperar 2 segundos para tomar otra lectura
}

```

Figura 12. Primer programa de medición de temperatura.

El código fuente de este programa puede encontrarse en el ANEXO I.

- **Paso 3: Verificación de datos de temperatura**

Una vez compilado el código, se pueden ver los resultados obtenidos en el monitor serial que provee Arduino IDE (ver Figura 13).

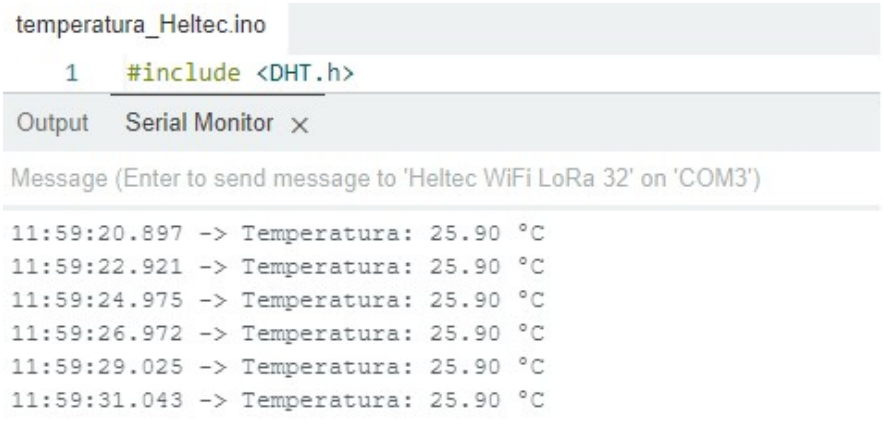

Figura 13. Resultados obtenidos en el monitor serial.

Fase 2: Recepción de datos por parte del Gateway

- **Paso 1: Configuración de Gateway**

Dado que al Gateway le fue asignado la dirección IP 192.168.0.254 conforme a la etapa de diseño, se configuró una red local que permitió establecer conexiones con el Gateway a través de un cable LAN. La configuración de esta red se puede apreciar en la Figura 14.

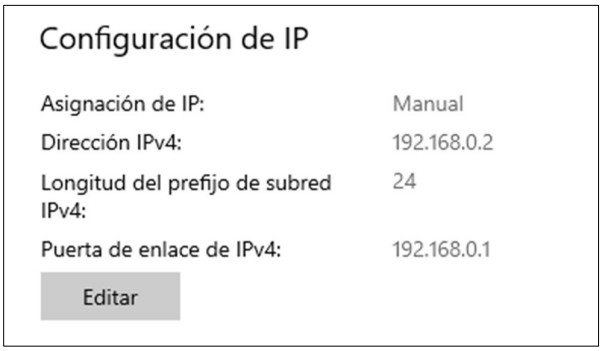

Figura 14. Configuración de red local.

Las configuraciones más importantes del Gateway realizadas en base a los parámetros regionales definidos por LoRa Alliance [28] son: (1) el plan de canal US915, (2) Min Datarate SF10BW125 y (3) Max Datarate. Ver la Figura 15.

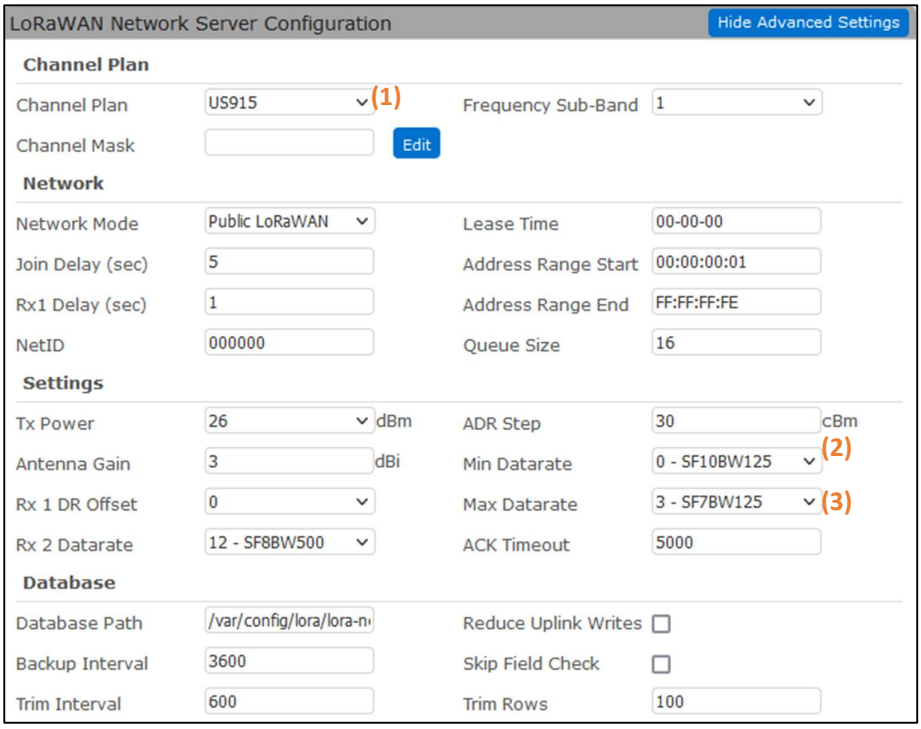

Figura 15. Configuraciones del Gateway.

Dentro del Gateway, fue necesario registrar el dispositivo final con las llaves propuestas en la etapa de diseño, como se puede ver en la Figura 16.

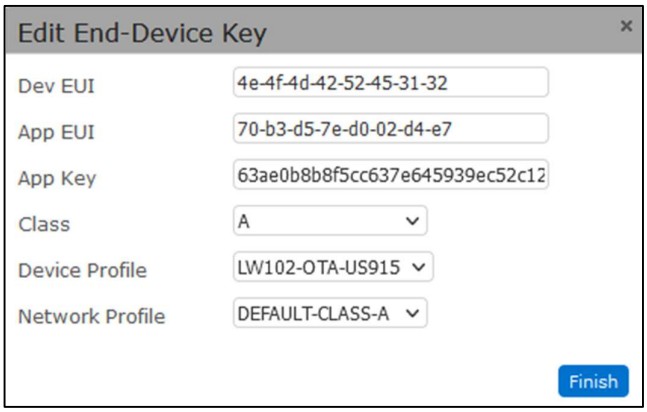

Figura 16. Registro del dispositivo final.

- **Paso 2: Configuración del dispositivo final modificando la programación de la fase 1**

Se modificó el código para que este permita la conexión con el Gateway de LoRaWAN. Primero, se definió el tipo de sensor usado con la definición DHTTYPE y cuál fue el pin de

datos con la definición DHTPIN. En este caso, el sensor es un DHT11 y conforme a lo estipulado en el diseño se seleccionó el pin 13 como se ve en la Figura 17.

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <heltec.h>
#include <DHT.h>

#define DHTPIN 13
#define DHTTYPE DHT11 // Tipo de sensor DHT11

DHT dht(DHTPIN, DHTTYPE);
//
```

Figura 17. Modificación de código para conexión con Gateway de LoRaWAN.

Luego se establecieron las llaves del dispositivo conforme a las características de la Tabla 5. Las llaves AppEUI y DevEUI se ingresaron en formato little-endian mientras que el AppKey en formato big-endian. El formato big-endian y little-endian hacen referencia a la manera en que se introducen los bits.

En little endian, el bit menos significativo es almacenado en la menor dirección de memoria, mientras que en big endian, el bit más significativo es almacenado en la menor dirección de memoria [29].

Las llaves ingresadas se pueden apreciar en la Figura 18.

```
// Esta llave en little-endian
static const u1_t PROGMEM APPEUI[8]={ 0xe7,0xd4, 0x02,0xd0,0x7e,0xd5,0xb3,0x70 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// Esta llave en little endian
static const u1_t PROGMEM DEVEUI[8]={ 0x32,0x31, 0x45,0x52,0x42,0x4d,0x4f,0x4e };
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

//Esta llave en big endian
static const u1_t PROGMEM APPKEY[16] = { 0x63,0xae,0x0b,0x8b,0x8f,0x5c,0xc6,0x37,0xe6,0x45,0x93,0x9e,0xc5,0x2c,0x12,0x51
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
```

Figura 18. AppEUI, DevEUI y Appkey.

El tiempo entre cada transmisión, es decir, cada cuánto se envían los mensajes, se asignó mediante la constante **TX_INTERVAL** y su valor fue configurado en 15 segundos, como se muestra en la Figura 19.

```
const unsigned TX_INTERVAL = 15;
```

Figura 19. Tiempo entre cada transmisión.

Otra sección que se destaca del nuevo código es la preparación de los datos, donde la variable **strData** almacena la información recolectada del sensor, la cual posteriormente se

envía a través de la función **LMIC_setTxData2** que permite enviar el mensaje en un formato que sea aceptado por el Gateway (ver Figura 20).

```
void do_send(osjob_t* j){
    // Chequear si existe algún trabajo TX/RX en proceso
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        //se prepara los datos para la transmisión
        temperature = dht.readTemperature();
        delay(5000);
        humidity = dht.readHumidity();
        delay(5000);
        //String strData="Temperatura="+String(temperature)+"°C Humedad="+humidity+"%";
        String strData="Temperatura="+String(temperature)+"°C";
        if (isnan(temperature) || isnan(humidity))
        {
          Serial.println("Failed to read from DHT sensor!");
        }else{
            Serial.print(strData);
        }
        char charMensaje[36];
        strData.toCharArray(charMensaje, strData.length()+1);
        LMIC_setTxData2(1,(uint8_t *)charMensaje, sizeof(charMensaje)-1, 0);
        Serial.println(F("Packet queued"));
    }
    // Próxima TX luego de el evento TX_COMPLETE
}
```

Figura 20. Función de preparación de datos.

Como se muestra en la Figura 21, en esta sección de código, se realizó el proceso de unión a la red y se mostró información importante como el identificador de red, la dirección del dispositivo y las claves de sesión. También se deshabilitó la comprobación de enlace, dado que esta puede interferir con la capacidad de transmitir datos.

```
case EV_JOINING:
    Serial.println(F("EV_JOINING"));
    break;
case EV_JOINED:
    Serial.println(F("EV_JOINED"));
    {
      u4_t netid = 0;
      devaddr_t devaddr = 0;
      u1_t nwkKey[16];
      u1_t artKey[16];
      LMIC_getSessionKeys(&netid, &devaddr, nwkKey, artKey);
      Serial.print("netid: ");
      Serial.println(netid, DEC);
      Serial.print("devaddr: ");
      Serial.println(devaddr, HEX);
      Serial.print("AppSKey: ");
      for (size_t i=0; i<sizeof(artKey); ++i) {
        if (i != 0)
          Serial.print("-");
        printHex2(artKey[i]);
      }
      Serial.println("");
      Serial.print("NwkSKey: ");
      for (size_t i=0; i<sizeof(nwkKey); ++i) {
              if (i != 0)
                      Serial.print("-");
              printHex2(nwkKey[i]);
      }
      Serial.println();
    }
    LMIC_setLinkCheckMode(0);
    break;
```

Figura 21. Proceso de unión a la red.

Otra adición importante al código fue que, cuando se termine una transmisión, se agende una nueva. Esto se lo realizó mediante la función **os_setTimedCallback** como se puede ver en la Figura 22.

```
case EV_TXCOMPLETE:
    Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
    if (LMIC.txrxFlags & TXRX_ACK)
      Serial.println(F("Received ack"));
    if (LMIC.dataLen) {
      Serial.print(F("Received "));
      Serial.print(LMIC.dataLen);
      Serial.println(F(" bytes of payload"));
    }
    // agendar la siguiente transmisión
    os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
    break;
```

Figura 22. Función para agendar una nueva transmisión.

Finalmente se tienen las funciones **setup()** y **loop()** que permiten definir las configuraciones, así como inicializar el sistema operativo e iniciar con la seguridad OTAA, como se puede ver en la Figura 23.

```
void setup() {
    Serial.begin(9600);
    dht.begin();
    Serial.println(F("Starting"));
    os_init();
    // Resetear el estado MAC state. Se descartan las sessiones y los datos pendientes de envío.
    LMIC_reset();
    // Inicia el trabajo (automáticamente inicia OTAA)
    do_send(&sendjob);
    //delay(1000);
}

void loop() {

    os_runloop_once();
}
```

Figura 23. Funciones setup() y loop().

El código completo de la implementación puede encontrarse en el ANEXO I.

- **Paso 3: Verificación de conexión entre Gateway y dispositivo final**

En el monitor serial se puede en ver los siguientes mensajes: (1) el sistema inicia, (2) un paquete se pone en la cola para enviarlo cuando ya se establezca la conexión entre el dispositivo y el Gateway, (3) comienza el proceso de unión a la red, (4) intentos fallidos de conexión, y (5) conexión exitosa. (Ver la Figura 24).

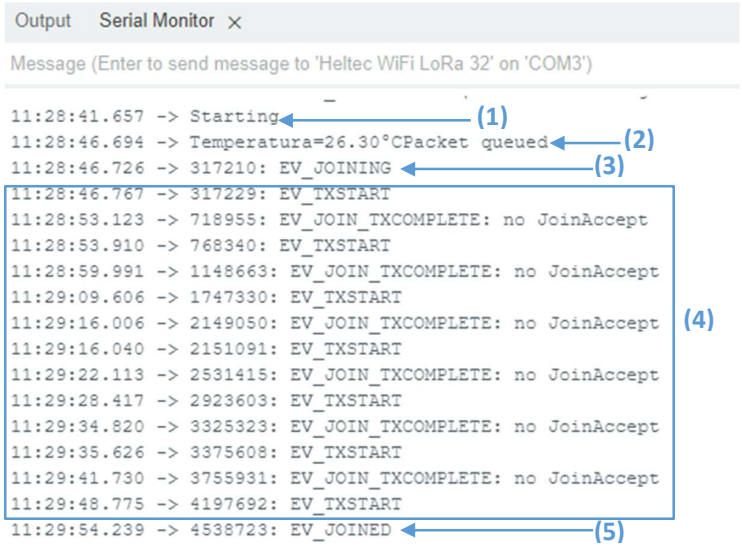

Figura 24. Dispositivo estableciendo conexión.

Posteriormente, el dispositivo empezó a enviar los datos con normalidad hacia el Gateway como se muestra en la Figura 25.

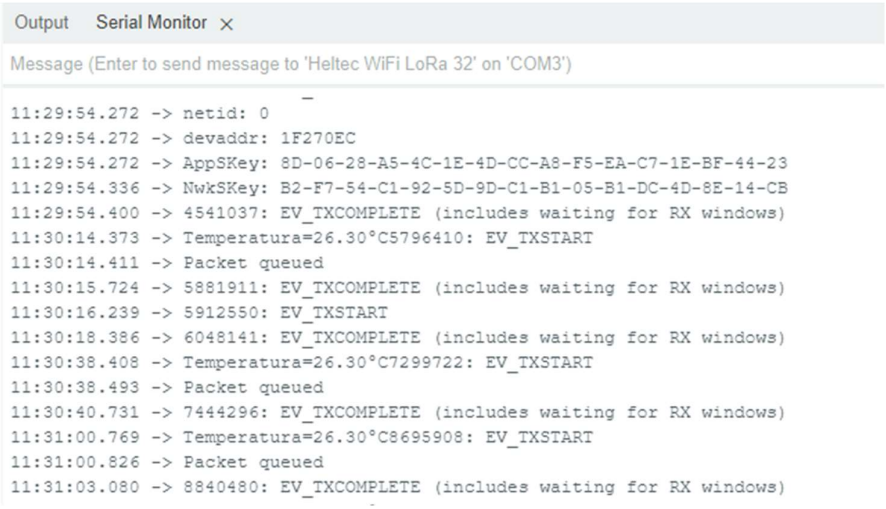

Figura 25. Envío de datos por parte del dispositivo al Gateway.

Por otra parte, en el Gateway se mostró la conexión exitosa como se aprecia en la Figura 26.

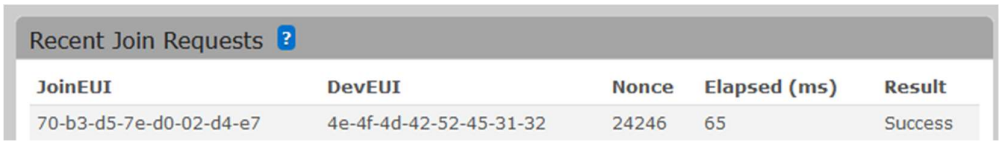

Figura 26. Conexión exitosa al Gateway.

Los detalles de la sesión se pueden ver en la Figura 27.

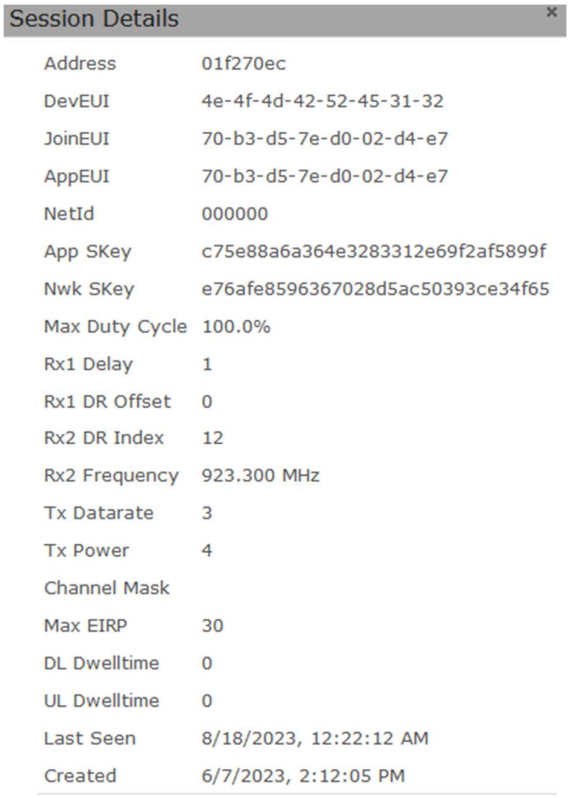

Figura 27. Detalles de sesión.

Además, se pueden apreciar los paquetes recibidos en la Figura 28.

Recent Rx Packets

Time	Freq	Datarate	CRC	SNR	RSSI	Size	Type	Data	Details
721348003	902.500	SF7BW125	OK	8.2	-25	15	UpUnc	QOxw8gGADAAASbKAdm...	
722899443	902.900	SF7BW125	OK	8.8	-26	15	UpUnc	QOxw8gGADQAAu+8+Sk...	
724450884	903.700	SF7BW125	OK	8.8	-25	15	UpUnc	QOxw8gGADgAAoSvIlz...	
726002324	903.300	SF7BW125	OK	8.2	-26	15	UpUnc	QOxw8gGADwAANfz0Xt...	
727553739	902.700	SF7BW125	OK	7.2	-30	15	UpUnc	QOxw8gGAEAAAknrZQN...	
728908292	903.700	SF7BW125	ERR	-11.2	-109	204	Unknown	bncz7T0UdRCO558Ct/...	
729105156	903.300	SF7BW125	OK	8.5	-27	15	UpUnc	QOxw8gGAEQAAi7mRwL...	
730656587	902.500	SF7BW125	OK	9.5	-30	15	UpUnc	QOxw8gGAEgAAleEa1T...	
732208019	902.300	SF7BW125	OK	9.5	-30	15	UpUnc	QOxw8gGAEwAAmLBgEv...	
733759452	903.500	SF7BW125	OK	7	-25	15	UpUnc	QOxw8gGAFAAA34ajvc...	
745526843	903.100	SF7BW125	ERR	-10.8	-109	86	Unknown	tVe3aDyoOsFachuu+x...	
755887532	903.700	SF7BW125	OK	9.8	-28	48	UpUnc	QOxw8gGAFQABeP+S1/...	
778015891	902.900	SF7BW125	OK	8.5	-27	48	UpUnc	QOxw8gGAFgABH1i14s...	
799925667	903.100	SF7BW125	ERR	-11.2	-109	239	Unknown	52RMJiJMxrxhxuYjOb...	

Figura 28. Paquetes recibidos.

A través de estos procesos, se demostró la correcta conexión y recepción de datos por parte del Gateway.

Fase 3: Procesamiento de datos en Node-RED y envío hacia MQTT

- **Paso 1: Programación de flujo de datos en Node-RED**

En primer lugar, fue necesario iniciar la aplicación de Node-RED desde el Gateway como se ve en la Figura 29.

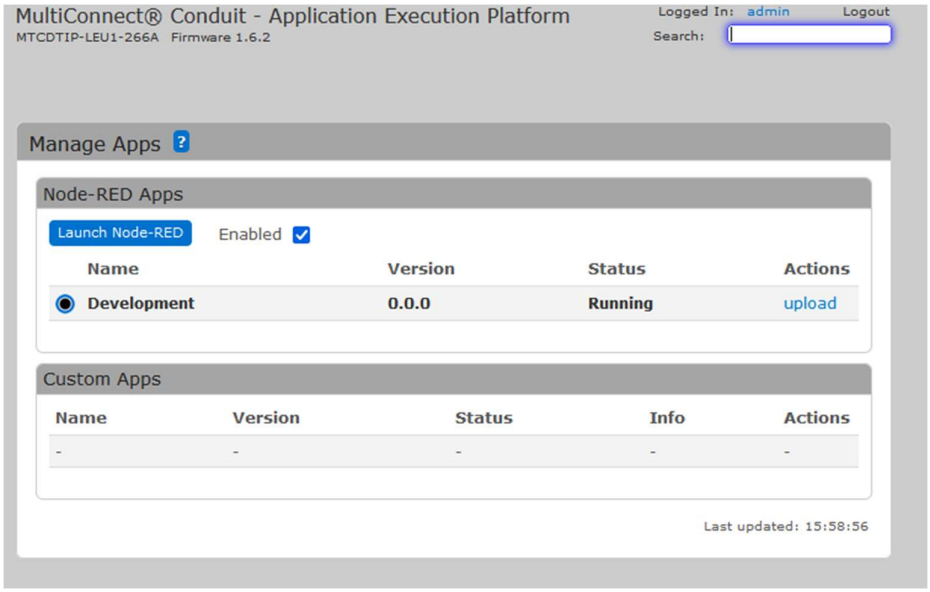

Figura 29. Iniciación de la aplicación Node-RED.

Posteriormente, se creó un flujo que permite recibir los datos desde el módulo de lora y seleccionar los datos relevantes que serán transmitidos mediante el servidor MQTT hacia un cliente suscriptor MQTT. El flujo puede verse en la Figura 30 donde **msg** contiene todo el objeto JSON y mensaje tiene el payload que se envió al servidor MQTT.

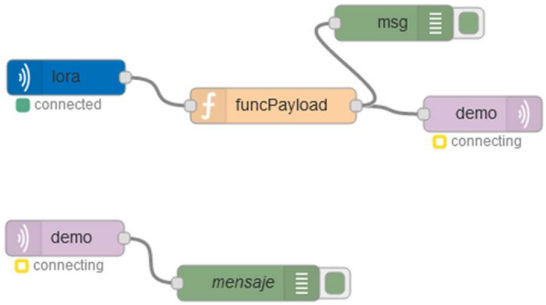

Figura 30. Flujo de Node-RED.

La función **funcPayload** permitió definir los elementos que se desean transmitir a un cliente suscriptor MQTT. Esta función contiene la identificación del dispositivo, los datos tomados del sensor y el tiempo, que es la fecha y hora. Véase la Figura 31.

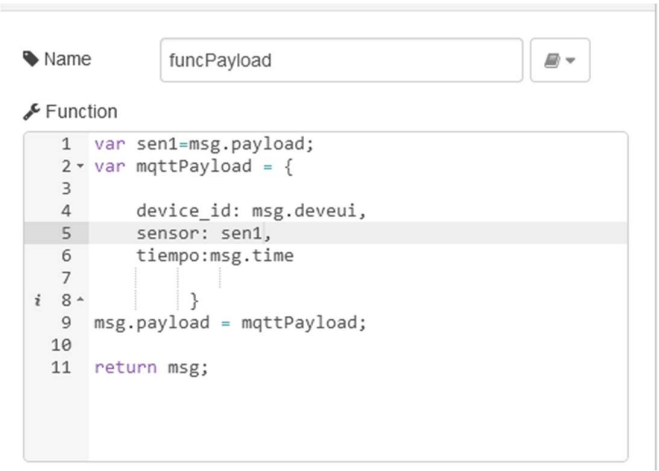

Figura 31. Función funcPayload.

El resultado del objeto **msg,** se puede ver en la Figura 32, donde la sección encerrada correspondería al payload generado por **funcPayload.**

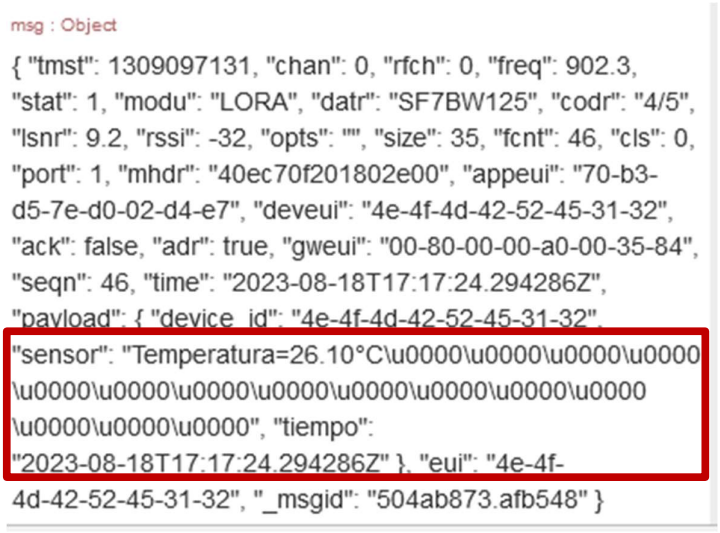

Figura 32. Objeto msg.

- **Paso 2: Configuración de servidor MQTT**

Para configurar un servidor MQTT con certificados autogenerados, se instaló OpenSSL y se procedió a colocar el comando **openssl genrsa -des3 -out ca.key 2048** para generar

una llave para la Autoridad Certificadora (CA). En primera instancia ocurrió un error como se muestra en la Figura 33.

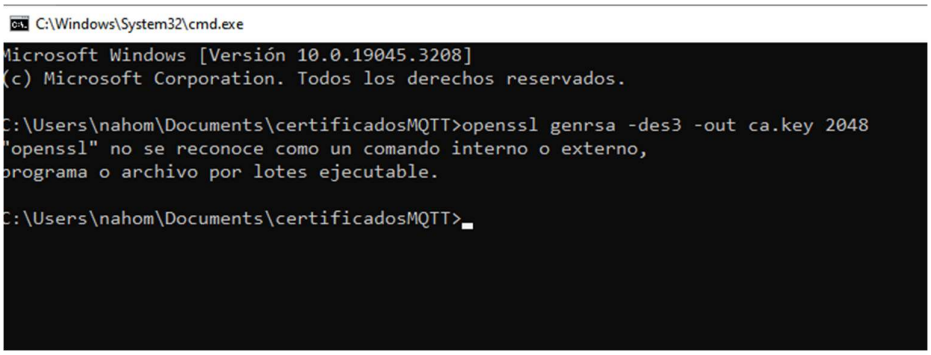

Figura 33. Error de generación de una llave para la autoridad certificadora.

Por ello fue necesario agregar a OpenSSL como variable de entorno, tal y como se muestra en la Figura 34.

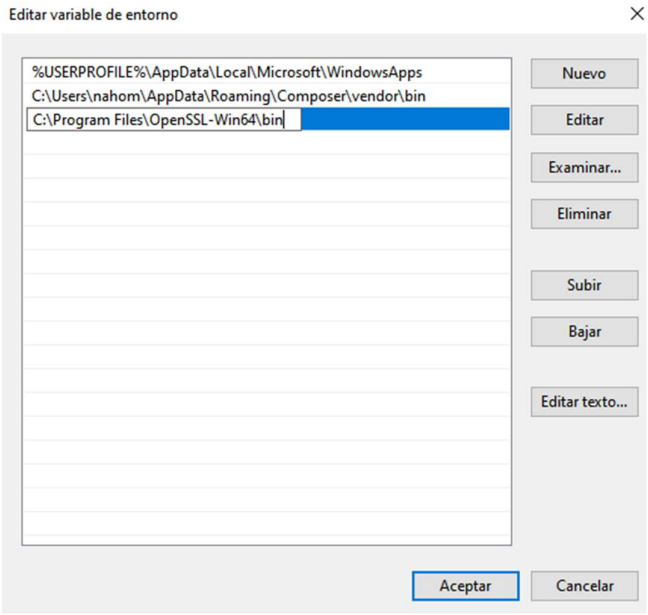

Figura 34. Agregar OpenSSL a las variables de entorno.

Una vez agregada como variable de entorno, se pudo crear la llave para CA como se ve en la Figura 35. Note que se generó una frase de seguridad que se debió utilizar en el resto del proceso.

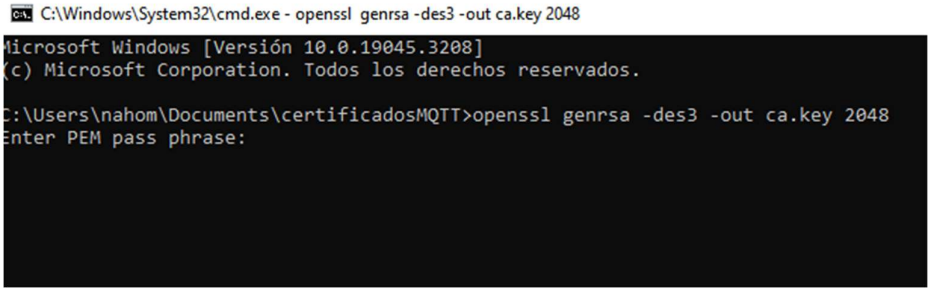

Figura 35. Generación de una llave para la autoridad certificadora.

Posteriormente, se creó un certificado para la CA, usando la llave creada anteriormente mediante y la frase de seguridad mediante el comando **openssl req -new -x509 -days 1826 -key ca.key -out ca.crt** como se muestra en la Figura 36.

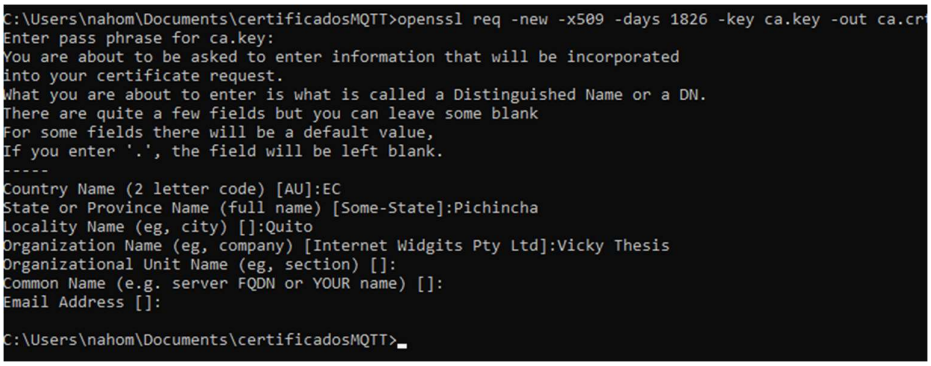

Figura 36. Creación de certificado para la autoridad certificadora.

Luego, se generó una llave para el servidor mediante el comando **openssl genrsa -out server.key 2048** tal y como se muestra en la Figura 37.

Figura 37. Generación de llave para el servidor.

Después, se generó una solicitud de certificado .csr. Al completar el formulario, se ingresó el nombre común, que generalmente corresponde al nombre de dominio del servidor mediante el comando **openssl req -new -out server.csr -key server.key**. Véase la Figura 38.

```
C:\Users\nahom\Documents\certificadosMQTT>openssl req -new -out server.csr -key server.key
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:EC
State or Province Name (full name) [Some-State]:Pichincha
Locality Name (eg, city) []:Quito
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Vicky Thesis
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:192.168.0.2
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

C:\Users\nahom\Documents\certificadosMQTT>
```

Figura 38. Solicitud de certificado csr.

Luego, se creó una clave para el servidor con el comando **openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 360**. Esto crea un certificado para el servidor utilizando el algoritmo RSA con una longitud de 2048 bits. Como se ve en la Figura 39.

```
C:\Users\nahom\Documents\certificadosMQTT>openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.k
ey -CAcreateserial -out server.crt -days 360
Certificate request self-signature ok
subject=C = EC, ST = Pichincha, L = Quito, O = Vicky Thesis, OU = 192.168.0.2, CN = 192.168.0.2
Enter pass phrase for ca.key:
```

Figura 39. Creación del certificado para el servidor.

Posteriormente, se movieron los archivos ca.crt, server.crt y server.key a una carpeta dentro de Mosquitto llamada “certificados” como se puede ver en la Figura 40.

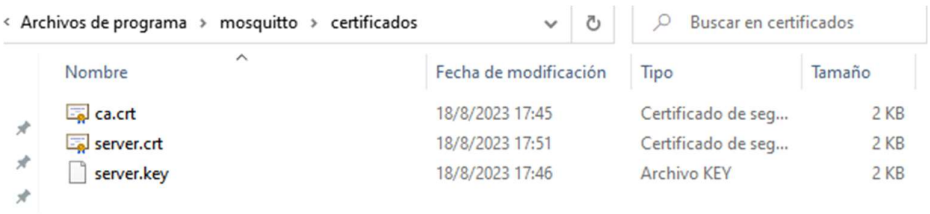

Figura 40. Carpeta “certificados”.

Luego, en el archivo de configuración, se especificaron las rutas hacia los archivos ca.crt, server.crt y server.key, además de la versión del protocolo tls (transport layer security), como se ve en la Figura 41.

```
cafile C:\Program Files\mosquitto\certificados\ca.crt
keyfile C:\Program Files\mosquitto\certificados\server.key
certfile C:\Program Files\mosquitto\certificados\server.crt
tls_version tlsv1.2
```

Figura 41. Configuración de TLS, rutas a certificados y llaves.

Luego, se creó en la carpeta de Mosquitto un archivo de usuarios (ver Figura 42 y Figura 43).

```
C:\Program Files>cd mosquitto

C:\Program Files\mosquitto>mosquitto_passwd -c usuarios vicky
Password:
Reenter password:
```

Figura 42. Creación de archivo “usuarios” y usuario “vicky”.

Figura 43. Archivo “usuarios”.

También, se agregó la línea **auth_anonymus false** dentro del archivo de configuración de Mosquitto, ya que esto obliga a ingresar un usuario al momento de suscribirse, como se ve en la Figura 44.

```
allow_anonymous false
password_file C:\Program Files\mosquitto\usuarios
```

Figura 44. Configuración para autenticación en Mosquitto.

Posteriormente, se habilitó el puerto seguro 8883 conforme a los requerimientos del diseño. Véase la Figura 45.

```
# ================================================
# Listeners
# ================================================
listener 8883
```

Figura 45. Configuración de puerto seguro de Mosquitto.

Finalmente, mediante el comando **mosquitto -c mosquitto.conf -v**. se pudo comprobar el funcionamiento del servidor Mosquitto (ver Figura 46).

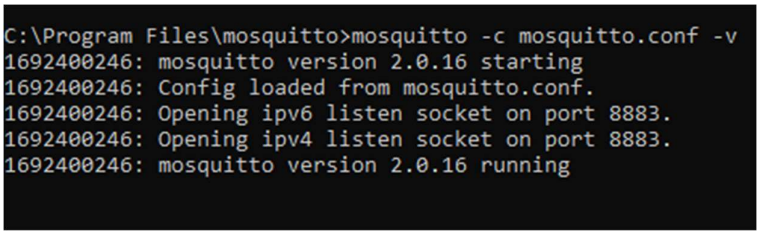

Figura 46. Servidor Mosquitto en funcionamiento.

- **Paso 3: Conexión Node-RED – MQTT (Mosquitto)**

Node-RED trabajó como cliente y fue asignado el tema “demo” que permitió identificar a los datos provenientes del sensor de temperatura. Véase la Figura 47.

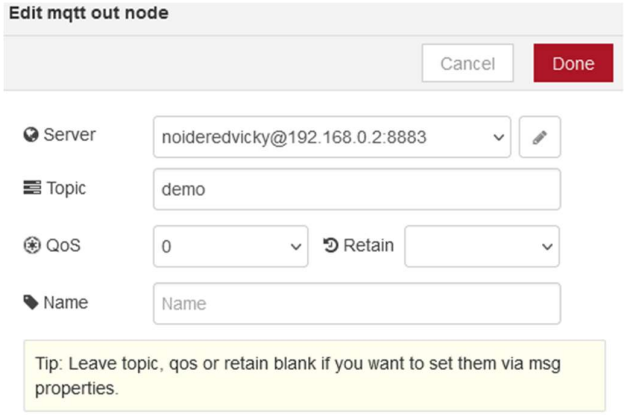

Figura 47. Configuración del tema y QoS de Node-RED.

En la Figura 48 se puede apreciar el nodo que se conecta con el servidor MQTT y que contiene el mensaje. Este funciona en la dirección IP y puertos especificados en la fase de diseño

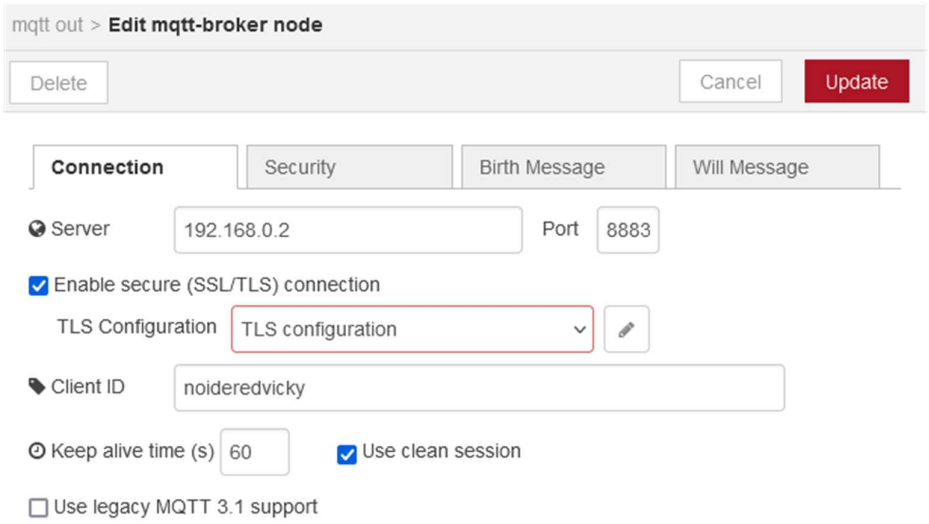

Figura 48. Configuración de la conexión MQTT en Node-RED.

- **Paso 4: Verificación de suscripción con cliente MQTT**

Se recibieron los datos desde Node-RED con el cliente **noideredvicky** como se ve en la Figura 49.

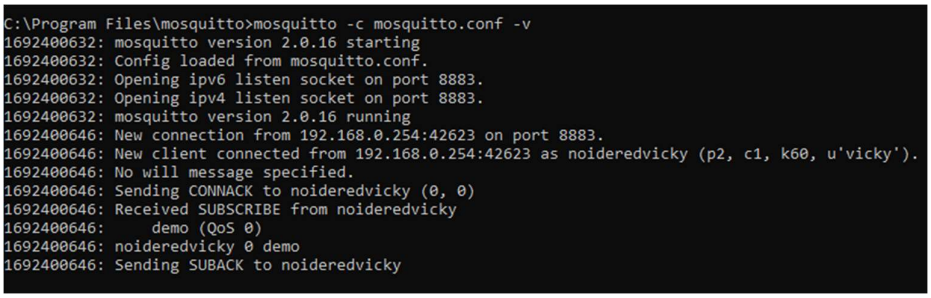

Figura 49. Verificación de cliente Node-RED a Mosquitto.

Para verificar que se puedan acceder a los datos del servidor MQTT, se creó un cliente MQTT. Sin embargo, este cliente no tenía definido ni usuario ni contraseña, por lo que se producía un error como se muestra en la Figura 50.

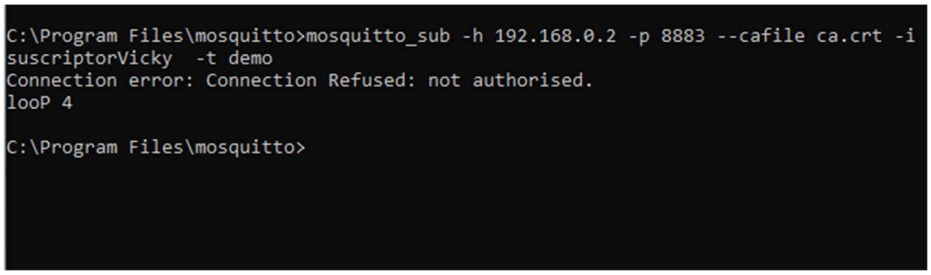

Figura 50. Cliente MQTT suscriptor sin definir datos de autenticación.

Sin embargo, cuando se especificó tanto un usuario como contraseña definida anteriormente, se pudo conectar sin problemas, como se ve en la Figura 51 y la Figura 52, lo que demuestra el uso exitoso del proceso de autenticación.

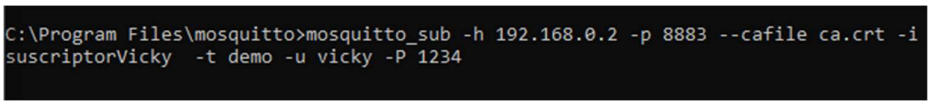

Figura 51. Cliente MQTT suscriptor con datos de autenticación definidos.

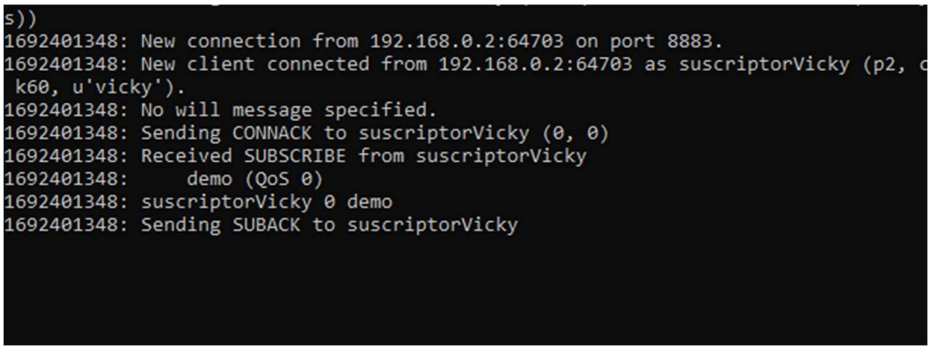

Figura 52. Verificación de suscripción de cliente MQTT en el servidor Mosquitto.

Los datos recibidos por este suscriptor pueden apreciarse en la Figura 53

Figura 53. Mensajes recibidos por el cliente MQTT.

Operación

Dentro de la operación, se realizó un cambio del mapeo de pines para que correspondan con los pines descritos en la fase de planificación. Esto permitió posteriores adaptaciones de otros elementos en la placa. El mapeo se puede apreciar en la Figura 54.

```
const lmic_pinmap lmic_pins = {
  .nss = 18,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 14,
  .dio = {26, 33, 32}
};
```

Figura 54. Mapeo de pines.

En la Figura 55, se pueden apreciar los valores mínimos, máximos y promedios tanto de RSSI (Indicador de fuerza de la señal recibida) como SNR (proporción de la señal a ruido). Al tener un RSSI promedio de -39 y un SNR promedio de 9.6 se puede decir que se tienen buenos niveles de comunicación [30].

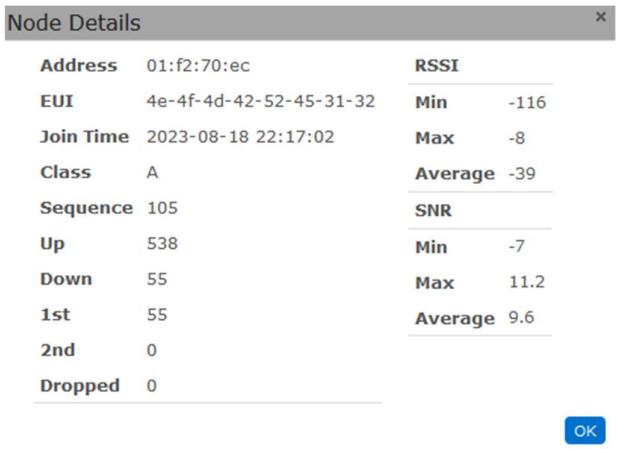

Figura 55. Promedios SRI y SNR.

Optimización

En la fase de operación, se pudo evidenciar que existían caracteres basura concatenados con los datos de temperatura recolectados. De ahí se determinó que el tamaño de la variable **charMensaje** era demasiado grande para los datos enviados. Por esto, se redujo el tamaño de **charMensaje[35] a charMensaje[21].** El cambio de código se puede apreciar en la Figura 56.

```
    char charMensaje[21];
    strData.toCharArray(charMensaje, strData.length()+1);
    LMIC_setTxData2(1,(uint8_t *)charMensaje, sizeof(charMensaje)-1, 0);
    Serial.println(F("Packet queued"));
}
```

Figura 56. Función de envío de datos por un char definido.

Los resultados de este cambio se pueden apreciar en la Figura 57, done ya no existen datos concatenados a la medición de la temperatura. De esta forma se eliminó el uso innecesario de recursos.

Figura 57. Nuevo object msg.

El payload de este objeto puede verse en la Figura 58.

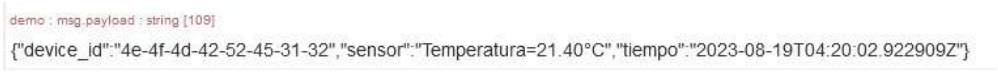

Figura 58. Nuevo payload.

Adicionalmente, se cambió el intervalo de transmisión a 180 segundos ya que las medidas cada 15 segundos suponían un mayor gasto energético. El intervalo no es mayor a los tres minutos dado que la temperatura ambiental puede cambiar mucho en ese tiempo, por ejemplo, en Quito el rango de temperaturas que se pueden apreciar en un solo día va de los 7 a los 20 °C [31] El cambio realizado se puede apreciar en la Figura 59.

```
const unsigned TX_INTERVAL = 180;
```

Figura 59. Modificación del tiempo entre transmisión.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Prototipo final

Los elementos del componente físico fueron colocados organizadamente dentro de un recipiente que permite protegerlo del polvo y mantiene los componentes unidos como se muestra en la Figura 60 y la Figura 61.

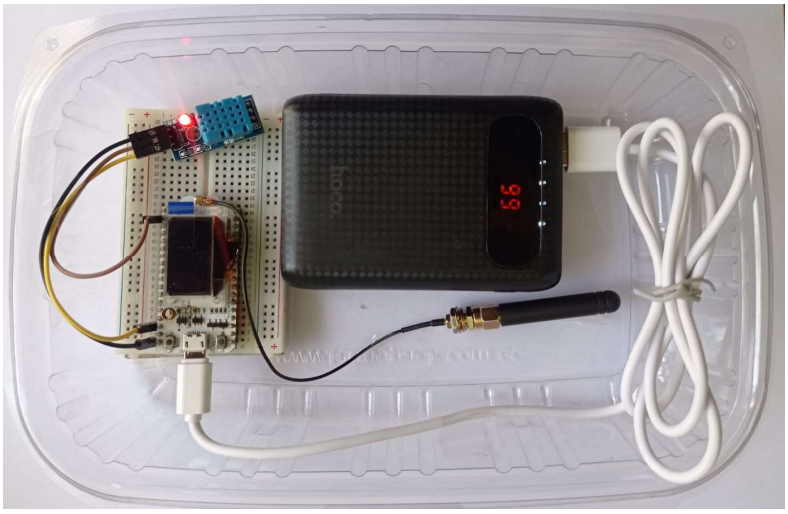

Figura 60. contenido del prototipo final ensamblado

Figura 61. Prototipo cerrado

Calidad de la señal

El RSSI promedio ha disminuido, lo que indica una mejora en la calidad de la señal, mientras que el SNR disminuyó en 0.5dB, pero todavía se puede considerar esta como una buena medida. Véase la Figura 62.

Figura 62. Promedio de RSSI y SNR del prototipo final.

Cobertura

Dado que la cobertura de LoRaWAN es radial, los cálculos para medir la cobertura se realizaron en base a la distancia máxima a la que se pudo colocar el dispositivo final sin perder la señal. El Gateway se colocó dentro de la Facultad de Ingeniería en Sistemas (FIS) de la Escuela Politécnica Nacional, concretamente en el laboratorio Smart Lab, ubicado en las coordenadas -0.21006, -78.48907. Para ello realizaron mediciones en 5 lugares donde se obtuvieron diferentes valores de RSSI como se ve en la Tabla 6.

Tabla 6. Tabla de cobertura.

Coordenadas	Distancia (km)	RSSI
-0.20807, -78.49001	0. 313	-105
-0.20699, -78.48674	0.433	-108
-0.20644, -78.48563	0.558	-110
-0.20530, -78.48385	0.789	-110
-0.20502, -78.48341	0.847	-111

Una descripción más detallada de los paquetes obtenidos se puede ver en el ANEXO II.

El último lugar donde se pudo obtener una señal fue a 0.847 km del Gateway, por lo que el área de cobertura viene a ser de 2.661 km^2. El radio de cobertura se puede apreciar en la Figura 63.

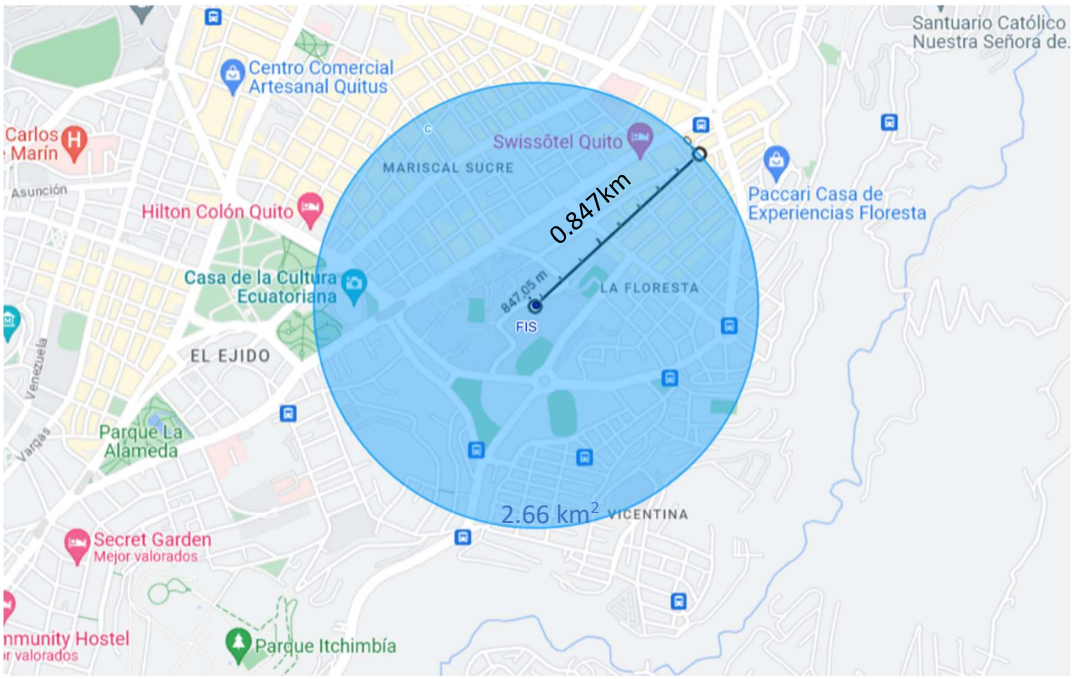

Figura 63. Cobertura estimada

Consumo energético

Se realizaron mediciones del porcentaje de batería del dispositivo final cada hora durante para comprobar la efectividad del cambio del intervalo de envío datos. Los resultados de la Figura 64 indican que para las medidas tomadas cada 15 segundos la batería se agotó al cabo de aproximadamente 18 horas mientras que para las medidas tomadas cada 180 segundos la batería tuvo una duración de 20 horas.

La vida del dispositivo aumentó en 2 horas al aumentar el tiempo de envío de datos, indicando la efectividad del aumento en el intervalo para reducir el consumo energético del prototipo.

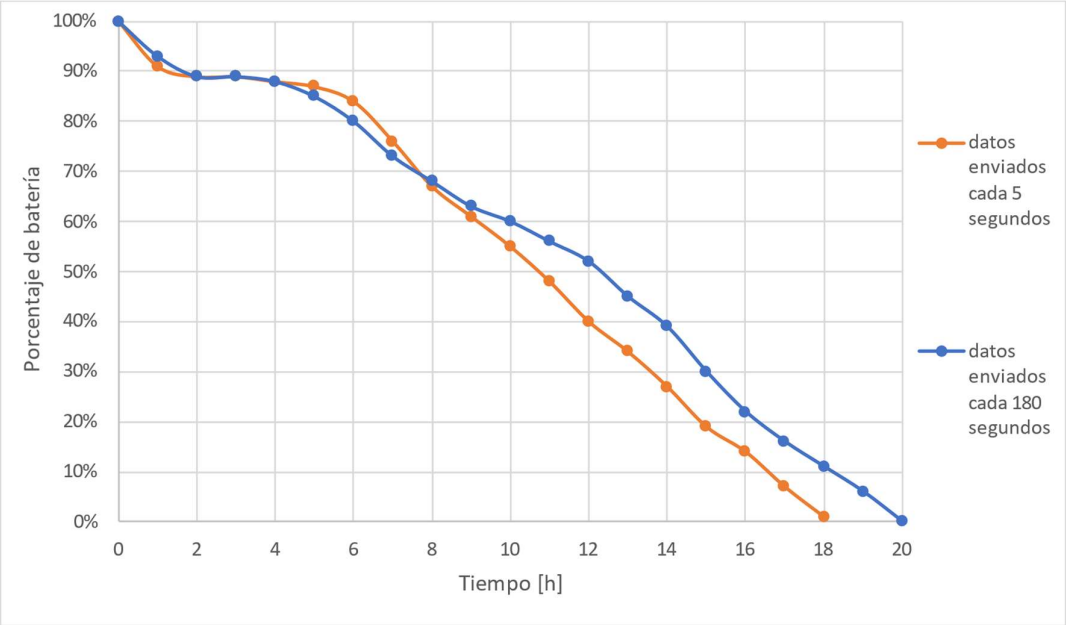

Figura 64. Porcentaje de batería vs tiempo

Costo del prototipo

El costo estimado del prototipo fue de $68.24 y su desglose puede verse en la Tabla 7. Cabe notar que el costo descrito no incluyó el Gateway LoRaWAN ya que de antemano se contaba con este dispositivo como parte del laboratorio SmartLab.

Tabla 7. Costo estimado del prototipo.

Dispositivo	Costo ($)
Módulo Heltec WiFi LoRa 32 V1	38.99 [32]
Power Bank Hoco	20.00 [33]
Protoboard	2.50 [34]
Cables de pines	1.75 [35]
Cable de carga Micro USB	5.00 [36]
Total	68.24

3.2 Conclusiones

Se concluye que se logró generar una arquitectura y diseño del sistema de monitoreo de la temperatura ambiental utilizando la tecnología LoRaWAN con cobertura de 2.661 km^2 que cubre toda el área del campus principal de la EPN.

Se logró desarrollar un prototipo del componente físico del sistema de monitoreo de la temperatura ambiental con elementos de bajo costo y desarrollar la infraestructura de comunicación utilizando LoRaWAN que dio acceso a los datos obtenidos a un cliente MQTT mediante el uso de un servidor MQTT.

Se concluye que las pruebas tanto del componente físico y de la infraestructura de comunicación del sistema de monitoreo de la temperatura ambiental fueron satisfactorias en tanto en la cobertura como la calidad de la señal obtenida para un solo dispositivo.

Es necesario establecer un tiempo razonable para el envío de datos desde el sensor para así optimizar la duración de la batería del dispositivo final. Además, otra forma en la que se redujo el consumo energético fue con el cambio del tamaño de payload, lo que también permitió una mejor lectura de los datos obtenidos.

3.3 Recomendaciones

Se recomienda cerciorarse de la compatibilidad entre las diferentes versiones de librerías y del módulo de LoRaWAN ya que esto puede afectar o incluso impedir la toma de datos y la conexión de los dispositivos finales con el Gateway.

Para futuros trabajos, se recomienda probar el prototipo en un ambiente real y aumentar número de nodos para poder estudiar el tráfico de paquetes y las posibles colisiones que se puedan presentar.

A partir de los resultados obtenidos se recomienda colocar el Gateway en una zona alta y despejada para mejorar la cobertura de la red LoRaWAN ya que en esta ocasión se obtuvieron medidas desde el interior de un edificio.

También es recomendable añadir más tipos de sensores que permitan conocer diferentes aspectos del ambiente, ya sea su temperatura, niveles de gases como el CO2 y la radiación ultravioleta, todo esto en conjunto con sensores GPS que permitan conocer las ubicaciones de los nodos finales.

Se recomienda mejorar el diseño físico del dispositivo final para que pueda ser probado en la intemperie sin correr riesgos de daño por viento, polvo, lluvia o sol como lo realizó el autor de [37]. Además, es necesario considerar la opción de una fuente alimentación de energía permanente.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] A. Ikpehai, B. Adebisi, K. Rabie, K. Anoh, R. Ande, M. Hammoudeh, H. Gacanin y U. Mbanaso, «Low-Power Wide Area Network Technologies for Internet-of-Things: A Comparative Review,» *IEEE Internet of Things Journal,* vol. 6, nº 2, pp. 2225-2240, 2019.

[2] K. Mekki, E. Bajic, F. Chaxel y F. Meyer, «Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT,» *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops),* pp. 197-202, 2018.

[3] K. Mekki, E. Bajic, F. Chaxel y F. Meyer, «A comparative study of LPWAN technologies for large-scale IoT deployment,» *ICT Express,* vol. 5, nº 1, pp. 1-7, 2019.

[4] M. I. Hossain y J. Markendahl, «Comparison of LPWAN Technologies: Cost Structure and Scalability,» *Wireless Personal Communications,* vol. 121, pp. 887-903, 2021.

[5] Node-RED, «Interacting with Arduino,» [En línea]. Available: https://nodered.org/docs/faq/interacting-with-arduino. [Último acceso: 12 Agosto 2023].

[6] IBM, «MQTT messaging,» 07 Mayo 2021. [En línea]. Available: https://www.ibm.com/docs/en/mapms/1_cloud?topic=reference-mqtt-messaging. [Último acceso: 12 Agosto 2023].

[7] R. Ligth, «Mosquitto: server and client implementation of the MQTT protocol,» *The Journal of Open Source Software,* p. 265, 2017.

[8] OASIS, «MQTT Version 3.1.1 Plus Errata 01,» 10 Diciembre 2015. [En línea]. Available: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html#_Toc442180823. [Último acceso: 12 Agosto 2023].

[9] MQTT, «MQTT: The Standard for IoT Messaging,» [En línea]. Available: https://mqtt.org/. [Último acceso: 12 Agosto 2023].

[10] K. Mindo, S. Karume y M. Thiga, «Designing a Fused Machine Learning Model for the Provision of Smart Health Care in MANETS,» *Saudi Journal of Engineering and Technology,* pp. 363-370, 2019.

[11] Cisco Press, «Analyzing the Cisco Enterprise Campus Architecture,» 15 Julio 2010. [En línea]. Available: https://www.ciscopress.com/articles/article.asp?p=1608131&seqNum=3. [Último acceso: 12 Agosto 2023].

[12] Cisco, «Nuevas tecologías simplificadas,» [En línea]. Available: https://www.cisco.com/c/dam/global/es_mx/products/servicios/docs/LCS_Brochure_Enterprise_Spanish_062006.pdf. [Último acceso: 12 Agosto 2023].

[13] TECmikro, «DHT11 Sensor de temperatura y humedad,» Agosto 2023. [En línea]. Available: https://tecmikro.com/sensores/141-dht11-sensor-de-temperatura-y-humedad.html. [Último acceso: 12 Agosto 2023].

[14] Heltec Automation, «WiFi LoRa 32 (V3),» [En línea]. Available: https://heltec.org/project/wifi-lora-32-v3/. [Último acceso: 12 Agosto 2023].

[15] Heltec Automation, «Hardware Update Logs,» [En línea]. Available: https://docs.heltec.org/en/node/esp32/dev-board/hardware_update_log.html#v1. [Último acceso: 12 Agosto 2023].

[16] Hoco., «Power bank «B20-10000 Mige» 10000 mAh with LED display,» [En línea]. Available: https://hocotech.com/product/power/portable-chargers/power-banks/power-bank-b20-10000-mige-10000-mah-with-led-display/. [Último acceso: 12 Agosto 2023].

[17] Multitech, «MultiTech Conduit IP67 Base Station,» [En línea]. Available: https://www.multitech.com/documents/publications/data-sheets/86002221.pdf. [Último acceso: 11 Agosto 2023].

[18] Arduino, «What is Arduino?,» 05 Febrero 2018. [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction. [Último acceso: 13 Agosto 2023].

[19] Adafruit, «Adafruit_Sensor,» 12 Mayo 2023. [En línea]. Available: https://github.com/adafruit/Adafruit_Sensor. [Último acceso: 13 Agosto 2023].

[20] ElectronicCats, «Beelan-LoRaWAN,» 11 Noviembre 2022. [En línea]. Available: https://github.com/ElectronicCats/Beelan-LoRaWAN. [Último acceso: 13 Agosto 2023].

[21] Heltec Automation, «Heltec_ESP32,» 26 Febrero 2023. [En línea]. Available: https://github.com/HelTecAutomation/Heltec_ESP32#how-to-use-this-library. [Último acceso: 13 Agosto 2023].

[22] mcci-catena, «arduino-lmic,» 09 Noviembre 2022. [En línea]. Available: https://github.com/mcci-catena/arduino-lmic. [Último acceso: 13 Agosto 2023].

[23] Node-RED, «Node-RED,» [En línea]. Available: https://nodered.org. [Último acceso: 14 Agosto 2023].

[24] Mosquitto, «Eclipse Mosquitto,» [En línea]. Available: https://mosquitto.org. [Último acceso: 14 Agosto 2023].

[25] GEMALTO, ACTILITY and SEMTECH, «FULL END–TO–END ENCRYPTION FOR IoT APPLICATION PROVIDERS,» Febrero 2017. [En línea]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf. [Último acceso: 14 Agosto 2023].

[26] LoRa Alliance, «FREQUENTLY ASKED QUESTIONS,» Febrero 2020. [En línea]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/la_faq_security_0220_v1.2_0.pdf. [Último acceso: 14 Agosto 2023].

[27] The Things Network, «End Device Activation,» 2023. [En línea]. Available: https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/. [Último acceso: 06 08 203].

[28] LoRa Alliance, «LoRaWAN™ 1.1 Regional Parameters,» 2017. [En línea]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan-regional-parameters-v1.1ra.pdf. [Último acceso: 17 08 2023].

[29] Princeton University, «Assembly Language: Part 1,» [En línea]. Available: https://www.cs.princeton.edu/courses/archive/spr18/cos217/lectures/13_Assembly1.pdf. [Último acceso: 15 Agosto 2023].

[30] LoRa® Developer Portal, «Predicting LoRaWAN Capacity,» [En línea]. Available: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/predicting-lorawan-capacity/. [Último acceso: 07 08 2023].

[31] Weather Spark, «El clima y el tiempo promedio en todo el año en Quito,» [En línea]. Available: https://es.weatherspark.com/y/20030/Clima-promedio-en-Quito-Ecuador-durante-todo-el-a%C3%B1o. [Último acceso: 16 Agosto 2023].

[32] MCIElectronics, «Modulo Wifi LoRa 32 Heltec,» [En línea]. Available: https://mcielectronics.cl/shop/product/modulo-wifi-lora-32-heltec-27575/. [Último acceso: 08 08 2023].

[33] Smart Mart, «B20-10000 Mige Mobile Power Bank 10000mAh dual USB 2.1A charging output carbon fiber pattern digital power indicator dual LED light for flashlight,» [En línea]. Available: https://www.carousell.sg/p/b20-10000-mige-mobile-power-bank-10000mah-dual-usb-2-1a-charging-output-carbon-fiber-pattern-digital-power-indicator-dual-led-light-for-flashlight-283296480/. [Último acceso: 15 08 20023].

[34] Mercado Libre, «Protoboard De 2 Canales Pequeño,» [En línea]. Available: https://articulo.mercadolibre.com.ec/MEC-520397495-protoboard-de-2-canales-pequeno-_JM#position=1&search_layout=stack&type=item&tracking_id=66d4f8fd-318a-4524-83bb-089cc4fca365. [Último acceso: 10 080 2023].

[35] Mercado Libre, «Cable Jumper 40 Pines 20cm Para Protoboard Arduino Dupont,» [En línea]. Available: https://articulo.mercadolibre.com.ec/MEC-517615926-cable-jumper-40-pines-20cm-para-protoboard-arduino-dupont-_JM#reco_item_pos=2&reco_backend=machinalis-v2p&reco_backend_type=low_level&reco_client=vip-v2p&reco_id=59e1dd03-b4c1-4371-a1f8-395359a4695d. [Último acceso: 02 08 2023].

[36] MiniSo, [En línea]. Available: https://www.miniso.com.ec/cable-de-datos-micro-usb-negro/p. [Último acceso: 09 08 2023].

[37] J. Miño, «DISEÑO E IMPLEMENTACIÓN DE SENSORES DE MEDICIÓN DE CONTAMINACIÓN DEL AIRE BASADO EN UNA RED LORAWAN EN LA ESCUELA POLITÉCNICA NACIONAL,» Escuela Politécnica Nacional, Quito, 2021.

5 ANEXOS

ANEXO I. Repositorio de GitHub

La programación realizada se encuentra en el siguiente repositorio de GitHub: https://github.com/BriennedeTarth/TIC-LoRaWAN-Temperature.git

ANEXO II. Mediciones de Cobertura

Para las coordenadas -0.20807, -78.49001 se obtuvo el objeto JSON de la Figura 65.

msg : Object

{ "tmst": 456583772, "chan": 6, "rfch": 1, "freq": 903.5, "stat": 1, "modu": "LORA", "datr": "SF7BW125", "codr": "4/5", "lsnr": -7.8, "rssi": -105, "opts": "", "size": 35, "fcnt": 39, "cls": 0, "port": 1, "mhdr": "40ec70f201802700", "appeui": "70-b3-d5-7e-d0-02-d4-e7", "deveui": "4e-4f-4d-42-52-45-31-32", "ack": false, "adr": true, "gweui": "00-80-00-00-a0-00-35-84", "seqn": 39, "time": "2023-08-18T18:14:46.706424Z", "payload": { "device_id": "4e-4f-4d-42-52-45-31-32", "sensor": "Temperatura=32.40°C\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000", "tiempo": "2023-08-18T18:14:46.706424Z" }, "eui": "4e-4f-4d-42-52-45-31-32", "_msgid": "b1d78e32.4e287" }

Figura 65. Object msg para -0.20807, -78.49001.

Para las coordenadas -0.20699, -78.48674 se obtuvo el objeto JSON de la Figura 66.

msg : Object

{ "tmst": 2194409371, "chan": 4, "rfch": 1, "freq": 903.1, "stat": 1, "modu": "LORA", "datr": "SF7BW125", "codr": "4/5", "lsnr": -9, "rssi": -108, "opts": "", "size": 35, "fcnt": 103, "cls": 0, "port": 1, "mhdr": "40ec70f201806700", "appeui": "70-b3-d5-7e-d0-02-d4-e7", "deveui": "4e-4f-4d-42-52-45-31-32", "ack": false, "adr": true, "gweui": "00-80-00-00-a0-00-35-84", "seqn": 103, "time": "2023-08-18T18:43:44.519155Z", "payload": { "device_id": "4e-4f-4d-42-52-45-31-32", "sensor": "Temperatura=33.40°C\u0000\u0000\u0000\u0000^\u0013\r\u001f?\u0000\u0000\u0000", "tiempo": "2023-08-18T18:43:44.519155Z" }, "eui": "4e-4f-4d-42-52-45-31-32", "_msgid": "72accd23.8d5334" }

Figura 66. Object msg para -0.20699, -78.48674.

Para las coordenadas -0.20644, -78.48563 se obtuvo el objeto JSON de la Figura 67.

msg : Object
{ "tmst": 1705645939, "chan": 0, "rfch": 0, "freq": 902.3, "stat": 1, "modu":
"LORA", "datr": "SF7BW125", "codr": "4/5", "lsnr": -3.8, "rssi": -110, "opts": "",
"size": 35, "fcnt": 85, "cls": 0, "port": 1, "mhdr": "40ec70f201805500", "appeui"
"70-b3-d5-7e-d0-02-d4-e7", "deveui": "4e-4f-4d-42-52-45-31-32", "ack": fals
"adr": true, "gweui": "00-80-00-00-a0-00-35-84", "seqn": 85, "time":
"2023-08-18T18:35:35.753436Z", "payload": { "device_id": "4e-4f-
4d-42-52-45-31-32", "sensor": "Temperatura=40.70°C\u0000\u0000\u0000