ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

REDES DE SENSORES INALÁMBRICOS PARA IOT MONITOREO DEL NIVEL DE ENERGÍA DE LA SEÑAL RECIBIDA EN UN NODO LORAWAN

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

ELVIS ALEXANDER BONILLA CADENA
elvis.bonilla@epn.edu.ec

DIRECTOR:

CARLOS ROBERTO EGAS ACOSTA, MSc.
carlos.egas@epn.edu.ec

DMQ, abril 2023

CERTIFICACIONES

Yo, ELVIS ALEXANDER BONILLA CADENA 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.

ELVIS ALEXANDER BONILLA CADENA

Certifico que el presente trabajo de integración curricular fue desarrollado por ELVIS ALEXANDER BONILLA CADENA, bajo mi supervisión.

CARLOS ROBERTO EGAS ACOSTA, MSc.

DIRECTOR DE PROYECTO

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.

ELVIS ALEXANDER BONILLA CADENA
CARLOS ROBERTO EGAS ACOSTA. MSc.

DEDICATORIA

Este trabajo de integración curricular está dedicado a las personas más importantes en mi vida, quienes han sido mi pilar y sostén en cada paso que he dado.

En primer lugar, a mi padre Hugo Bonilla, cuyo amor, apoyo incondicional y sabios consejos han sido fundamentales para que yo pueda alcanzar mis metas y sueños. Gracias por ser mi guía en este camino y por enseñarme a nunca rendirme ante las adversidades.

A mi madre y mi hermana por estar siempre presentes con sus palabras de apoyo y su eterna compañía. Gracias por ser parte del sostén de mi vida y por impulsarme en cada momento.

A mi familia, mi razón de ser, mi motivo de lucha y mi fuente de inspiración. Gracias por brindarme su amor y su apoyo en cada una de mis decisiones y por ser mi gran ejemplo a seguir.

A mis amigos, por estar siempre a mi lado, apoyándome y acompañándome en cada momento. Gracias por sus risas, sus bromas y sus palabras de aliento, que me han ayudado a mantenerme firme en momentos difíciles.

Este trabajo no solo es el resultado de mi esfuerzo y dedicación, sino también es una muestra de mi gratitud hacia ustedes. Gracias por ser mi motivación, mi inspiración y mi razón de ser.

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a todas aquellas personas que han sido parte fundamental en la realización de este trabajo de integración curricular.

En primer lugar, a mi familia, por su apoyo incondicional en cada paso que he dado. Gracias por ser mi motivación y mi inspiración en momentos difíciles. Su amor, paciencia y comprensión han sido fundamentales para alcanzar mis metas y sueños.

A mi novia Lorena Tepud, por ser mi compañera de aventuras viajes y mucho más, por brindarme su ayuda y su apoyo constante. Gracias por ser mi cómplice y mi gran motivación en este proceso.

A Erick Tobar y Alexander Lara, por su amistad y por estar siempre presentes en los momentos más importantes de mi vida. Gracias por sus consejos, su apoyo y su compañía.

A Francesca Briones, por ser una gran amiga y por brindarme su ayuda y apoyo incondicional. Gracias por su paciencia y su disposición en cada momento.

Al club de robótica de la EPN, por enseñarme las bases y los fundamentos necesarios que me han ayudado a lo largo de mi carrera. Gracias por su dedicación y por brindarme su apoyo en cada uno de mis proyectos.

A la rama estudiantil IEEE, por todo lo aprendido durante mis años de voluntariado. Gracias por brindarme la oportunidad de crecer como persona y como profesional, y por enseñarme los valores de la solidaridad, el trabajo en equipo y el compromiso.

También quiero extender mi agradecimiento a un grupo muy especial de personas que han sido parte importante en mi vida durante estos años de carrera. A mis compañeros de clase y aventuras, Michael, Axel, Nubia, Lorena, Jhonatan y Dayana, por ser una fuente inagotable de alegrías y compañía irremplazable.

Gracias por compartir conmigo momentos inolvidables de risas, de esfuerzo, de estudio y de celebración. Gracias por su amistad, su apoyo y su ánimo, que han sido fundamentales para superar los momentos difíciles y alcanzar nuestras metas juntos.

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO VI
RESUMEN VII
ABSTRACT VIII

1 INTRODUCCIÓN 1
1.1 Objetivo general 2
1.2 Objetivos específicos 2
1.3 Alcance 3
1.4 Marco teórico 3
1.4.1 LoRa 3
1.4.2 LoRaWAN 5
1.4.3 Wio-E5 Development kit 9
1.4.4 Arduino Nano 10
1.4.5 Sentrius RG191 12
1.4.6 The Things Network (TTN) 13
1.4.7 Node-Red 14

2 METODOLOGÍA 15
2.1 Problema a resolver 15
2.2 Análisis de requisitos de la red 15
2.2.1 Banda de frecuencias 15
2.2.2 Factor de Spreading 16
2.2.3 Potencia de transmisión del nodo 16
2.2.4 Sensibilidad 17
2.2.5 Distancia máxima de cobertura 17
2.2.6 Red de sensores inalámbricos 20
2.3 Justificación de componentes y plataformas 20
2.3.1 Arduino Nano 20

2.3.2 Wio-E5 Development Kit 21
2.3.3 Sentrius RG191 22
2.3.4 The Things of Network 22
2.3.5 Node-Red 23
2.4 Implementación 23
2.4.1 Comandos AT Wio-E5 24
2.4.2 Consola The Things of Networks 26
2.4.3 Aurdino 34
2.4.4 Dashboard en Node-Red 42
2.4.5 Diseño electrónico del prototipo 47

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES 49
3.1 Resultados 49
3.1.1 Implementación del prototipo 49
3.1.2 Funcionamiento del sensor y monitoreo por el puerto serial 51
3.1.3 Funcionamiento y precisión del nodo sensor 52
3.1.4 Funcionamiento de la aplicación de monitoreo 56
3.1.5 Iconos dinámicos del RSSI 57
3.1.6 Sistema de alerta 59
3.2 Trabajos Futuros 59
3.3 Conclusiones 59
3.4 Recomendaciones 61

4 REFERENCIAS BIBLIOGRÁFICAS 62

5 ANEXOS 65

RESUMEN

El Internet de las Cosas (IoT) ha generado un crecimiento exponencial en el uso de tecnologías, lo que ha creado nuevas problemáticas, una de ellas es la necesidad de transmitir datos a largas distancias con un bajo consumo de energía. Para dar solución a este problema, se han desarrollado tecnologías como LPWAN (Low Power Wide Area Networks), en particular LoRaWAN (Long Range Wide Area Networks), que destaca por su bajo consumo de energía y largas distancias de transmisión alcanzadas por su técnica de modulación basada en espectro ensanchado. Es importante destacar que LoRa y LoRaWAN no son lo mismo, ya que el primero es un protocolo de capa física y el segundo es un protocolo de comunicación que hace uso de las tecnologías LoRa. El objetivo de este trabajo de integración curricular es diseñar e implementar un prototipo que permita monitorear el nivel de energía de la señal recibida en un nodo LoRaWAN conectado a un Gateway LoRa. En la primera sección se explicará detalladamente la teoría sobre LoRaWAN, mientras que en la segunda sección se planteará el problema a resolver y se realizará el diseño de la red y el nodo. En la tercera sección se llevará a cabo la implementación del prototipo y se obtendrán los resultados correspondientes, detallando las ventajas y limitaciones del prototipo desarrollado. Este prototipo puede resultar de gran utilidad para diferentes aplicaciones, desde la industria hasta la domótica y la salud.

Pablabras clave: IoT, LoRaWAN, monitorear, nivel de energía, prototipo.

ABSTRACT

The Internet of Things (IoT) has generated exponential growth in the use of technologies, creating new challenges, one of which is the need to transmit data over long distances with low energy consumption. To address this problem, technologies such as LPWAN (Low Power Wide Area Networks) have been developed, particularly LoRaWAN (Long Range Wide Area Networks), which stands out for its low energy consumption and long transmission distances achieved through its modulation technique based on spread spectrum. It is important to note that LoRa and LoRaWAN are not the same, as the former is a physical layer protocol and the latter is a communication protocol that uses LoRa technologies. The objective of this curricular integration work is to design and implement a prototype that allows monitoring the energy level of the signal received at a LoRaWAN node connected to a LoRa Gateway. The first section will explain in detail the theory behind LoRaWAN, while the second section will present the problem to be solved and the design of the network and node. The third section will carry out the implementation of the prototype and obtain the corresponding results, detailing the advantages and limitations of the developed prototype. This prototype can be useful for various applications, from industry to home automation and health.

Keywords: IoT, LoRaWAN, monitoring, energy level, prototype.

1 INTRODUCCIÓN

El Internet de las Cosas (IoT, Internet of Things) ha sido una de las áreas de investigación y aplicación más prometedoras en los últimos años, gracias al constante desarrollo de las redes de sensores. Este concepto implica un mundo donde los objetos sean inteligentes y estén interconectados, lo que ha generado un crecimiento exponencial en el uso de estas tecnologías [1]. Sin embargo, este aumento ha traído nuevas problemáticas, una de ellas es la necesidad de transmitir datos a largas distancias con un consumo mínimo de batería.

Para dar solución a este problema, se han desarrollado tecnologías como LPWAN (Low Power Wide Area Networks), que permite la transmisión de datos a larga distancia con un bajo consumo de energía. En este sentido, LoRaWAN (Long Range Wide Area Networks) es uno de los protocolos de comunicación para redes LPWAN que ha recibido una atención significativa por parte de la comunidad investigadora en los últimos años [2].

Este protocolo destaca por su bajo consumo de energía y las largas distancias de transmisión alcanzadas por su técnica de modulación [1]. LoRa es un protocolo de capa física en el que se destaca principalmente su largo alcance debido a la técnica de modulación basada en espectro ensanchado, lo que hace que sea robusto ante el ruido blanco y se vuelve independiente de las implementaciones de capa superior.

Es importante destacar que LoRa y LoRaWAN no son lo mismo, ya que el primero es un protocolo de capa física que implementa un esquema de modulación propio, y el segundo es un protocolo de comunicación que hace uso de las tecnologías LoRa, es decir, el primero se convierte en la herramienta usada por el segundo para efectuar el proceso de comunicación [1].

El objetivo de este trabajo de integración curricular es diseñar e implementar un prototipo que permita monitorear el nivel de energía de la señal recibida en un nodo LoRaWAN conectado a un Gateway LoRa. Este prototipo utilizará una interfaz vinculada a un servidor para mostrar los valores obtenidos de manera clara y ordenada, lo que permitirá una fácil interpretación de resultados.

En la primera sección de este trabajo se llevará a cabo una revisión de la teoría sobre LoRaWAN, explicando detalladamente su funcionamiento, características y aplicaciones. En la segunda sección se planteará el problema a resolver, que consiste en la necesidad

de monitorear el nivel de energía de la señal recibida en un nodo LoRaWAN, y se realizará el diseño de la red y el nodo, incluyendo los componentes necesarios y la configuración correspondiente.

En la tercera sección se llevará a cabo la implementación del prototipo y se obtendrán los resultados correspondientes. En esta sección se explicará detalladamente la configuración de los dispositivos, la programación y la conexión a la interfaz del servidor. También se incluirá un análisis de los resultados obtenidos, detallando las ventajas y limitaciones del prototipo desarrollado.

En resumen, este trabajo de integración curricular tiene como objetivo diseñar e implementar un prototipo que permita monitorear el nivel de energía de la señal recibida en un nodo LoRaWAN conectado a un Gateway LoRa. El monitoreo del nivel de energía de la señal en un nodo LoRaWAN es una tarea fundamental para garantizar el correcto funcionamiento de la red y la optimización de su rendimiento, por lo que la implementación de un prototipo como el propuesto puede resultar de gran utilidad para diferentes aplicaciones, desde la industria hasta la domótica, la agricultura, etc.

1.1 OBJETIVO GENERAL

Monitorear el nivel de energía de la señal recibida en un nodo LoRaWAN en el contexto de una red punto-punto.

1.2 OBJETIVOS ESPECÍFICOS

1. Revisar el fundamento teórico de las tecnologías LoRa y LoRaWAN para su uso dentro de las redes de sensores inalámbricos.

2. Diseñar una red de sensores inalámbricos para el monitoreo del nivel de energía de la señal en un nodo LoRaWAN.

3. Implementar un sistema de monitoreo de nivel de energía de la señal recibida desde un nodo terminal.

4. Presentar los resultados obtenidos tras la implementación del prototipo.

1.3 ALCANCE

El presente Trabajo de Integración Curricular tiene como objetivo implementar un sistema de monitoreo de nivel de energía de la señal en un nodo LoRaWAN, para lo que se comenzará por realizar el análisis a través de una fase teórica en donde se estudiarán las necesidades que se deben solventar para soluciones LoRaWAN, como lo son: bajo consumo de potencia y que los dispositivos finales puedan comunicarse de forma inalámbrica a grandes distancias, después del estudio se seleccionó el nodo LoRa E5 en combinación con un Arduino Nano, que permiten obtener el nivel de RSSI del nodo y que a través de su configuración pueden obtener y mostrar los valores contemplados.

Se realizará un estudio del uso de las herramientas necesarias para la configuración, programación y despliegue de la aplicación que permitirán la implementación del sistema de monitoreo. Una vez realizado el estudio del nodo seleccionado se definirá las características mínimas del sistema que permitan monitorear el nivel de energía de la señal en el nodo receptor.

Seguido se desarrollará el código que permita la obtención del nivel de energía de la señal, así como la comunicación serial con la computadora donde se visualizará en modo texto de forma clara y ordenada, de esta manera se comprobará el funcionamiento del prototipo, a la vez se diseñará la interfaz gráfica en Node-Red que permitirá mostrar la información del nodo LoRaWAN y presentarlo en pantalla. La implementación de servidores, nos permitirá la conexión remota con los nodos, la obtención de datos y la presentación de los mismos a través de la interfaz desarrollada.

Finalmente, se entrará en una fase de análisis de resultados, donde se podrá visualizar los datos arrojados por el sistema y determinar cómo está funcionando el nodo, comprobando así la validez del sistema implementado para monitorear el nivel de energía de la señal.

1.4 MARCO TEÓRICO

1.4.1 LoRa

LoRa (Long Range) es un protocolo de capa física y una modulación inalámbrica patentado por la empresa Semtech [1], este protocolo es utilizado en tecnologías LPWAN y tiene

grandes ventajas para los proyectos de IoT que utilizan dispositivos alimentados por baterías simples, ya que [2].

La modulación LoRa se basa en la combinación de dos técnicas, la primera es la Modulación por Desplazamiento de Frecuencia (Frequency Shifting Keying - FSK) y la siguiente es una variación del espectro ensanchado chirp (Chirp Spread Spectrum - CSS), de esta manera se tiene las características de baja potencia de FSK, y se consigue un amplio rango de comunicación por CSS; también hace uso de corrección de errores (Forward Error Correction - FEC) [1].

Opera a distintas frecuencias en la banda ISM, como se muestra en la Tabla 1.1, dependiendo de la zona donde se realice la implementación y los equipos usados para la misma.

Tabla 1.1: Rango de frecuencias de operación de LoRa para diferentes regiones [3]

Europa	Norteamérica	China	Japón	India
863-870 MHz	902-928 MHz	470-510 MHz	920-925 MHz	865-867 MHz

Además, el factor de Spreading puede tomar valores entre 7 y 12, según la aplicación a usarse, las distancias requeridas y la velocidad de transmisión que se desee [2]; a continuación se presentan las especificaciones según el SF en la tabla 1.2.

Tabla 1.2: Especificaciones de transmisión [2][3]

Modulación	Factor de Spreading	AB (kHz)	Vtx (bps)	Sensibilidad (dBm)
LoRa	12	125	250	-137
LoRa	11	125	440	-134.5
LoRa	10	125	980	-132
LoRa	9	125	1760	-129
LoRa	8	125	3125	-126
LoRa	7	125	5470	-123

1.4.1.1 Frequency Shifting Keying (FSK)

Es un método de transmisión digital que usa señales discretas que está compuesto por estados binarios, 0 lógico (bajo) y 1 lógico (alto).

El 0 lógico está representado por una onda a una frecuencia específica, y el 1 lógico está representado por una onda a una frecuencia diferente, donde la distancia entre el 0 y el 1 es la desviación o punto de cambio [4].

1.4.1.2 Chirp Spread Spectrum (CSS)

La técnica de modulación CSS o Chirp Spread Spectrum utiliza pulsos “chirp” modulados en una frecuencia lineal de banda ancha que permite codificar la información para realizar transmisiones en ráfagas [3], el salto entre las frecuencias se da en una secuencia pseudo-aleatoria.
Un chirp es la variación de frecuencias donde se incrementa (up-chirp) o decrementa (down-chirp) en un lapso de tiempo y su ancho de banda es equivalente al ancho de banda espectral de la señal [3].

1.4.1.3 Forward Error Correction (FEC)

Es una técnica de codificación de canal utilizada para minimizar los errores en la transmisión de datos sin utilizar retransmisión de paquetes.
La fuente (transmisor) envía datos redundantes y el destino (receptor) reconoce solo la parte de los datos que no contiene errores aparentes [5].

1.4.2 LoRaWAN

Es un protocolo de control de medio de acceso (Medium Access Control, MAC) que ha sido estandarizado por LoRa Alliance, el mismo se encarga de definir tanto la arquitectura de la red, Figura 1.1, como el protocolo de comunicación que usará el sistema [6], de esta forma se puede garantizar aspectos de suma importancia como lo son la vida útil de la batería de los nodos terminales, la capacidad de la red, la seguridad, la calidad de servicio (QoS), etc. [1][2].

El ancho de banda garantizado por LoRaWAN es suficiente para aplicaciones de comunicación Machine-to-Machine (M2M), donde se informen resultados obtenidos por sensores, sin embargo, aplicaciones de transmisión de datos en tiempo real como imágenes, video o audio streaming no son viables para esta tecnología debido a que está diseñada para trabajar a bajas velocidades con el fin de aumentar la vida útil de la batería [1][6].

Las características más importantes de LoRaWAN son [7]:

- ❒ Se puede usar en redes privadas y públicas.

- ❒ Se usa en administración de dispositivos.
- ❒ Posee topología estrella.
- ❒ Alcance de 10 a 15km en línea de vista.
- ❒ Soporte para 3 clases de nodos.
- ❒ Redes de largo alcance y bajo consumo.
- ❒ Usa encriptación AES 128.
- ❒ Baja tasa de transferencia de datos.
- ❒ Es un protocolo full dúplex.
- ❒ Cuenta con encriptación de extremo a extremo.
- ❒ Permite el registro en el aire de los nodos finales, con capacidad de multidifusión.

1.4.2.1 Arquitectura LoRaWAN

Por lo general, las redes LoRaWAN tienen topología estrella punto-multipunto. La arquitectura está conformada por las puertas de enlace (Gateways), los nodos terminales (LoRa Things), servidor de red(the things of network), servidor de aplicación (Dashboard Node Red) como se muestra en la Figura 1.1 [1].
En las redes LoRaWAN, los nodos no necesariamente deben estar asociados con un solo Gateway en específico, por esta razón, los datos transmitidos por un nodo pueden ser recibidos por múltiples Gateways. Cada Gateway reenvía el paquete recibido al servidor, a través de una red de Backhaul [1][2].

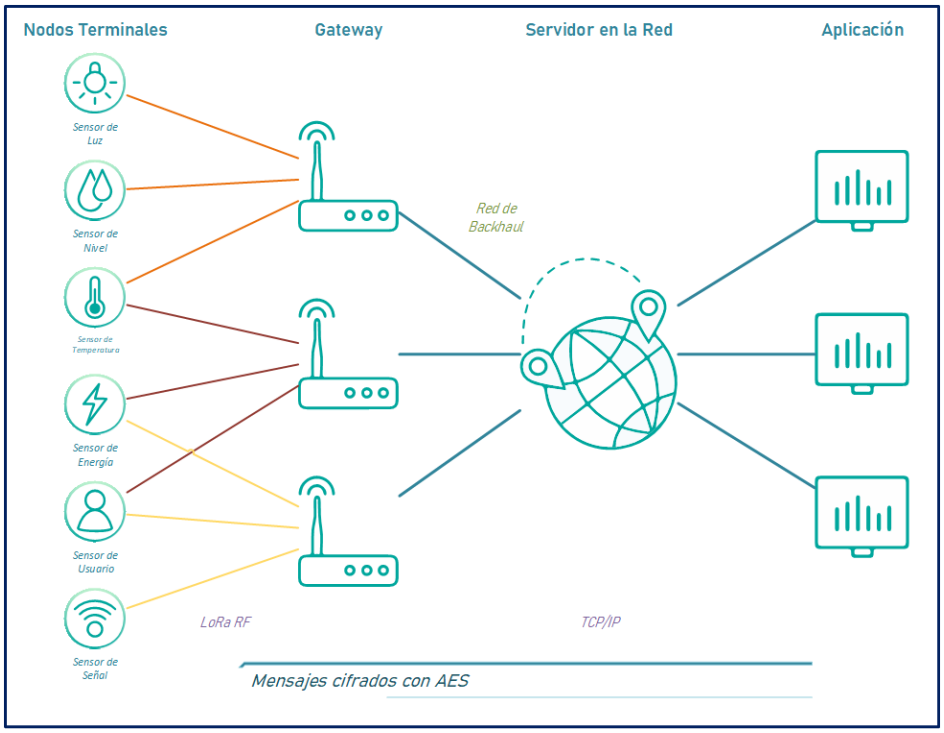

Figura 1.1: Topología de red LoRaWAN [1]

Finalmente, el servidor de red ayuda en la activación segura de los dispositivos, el almacenamiento de claves raíz y la generación de claves de sesión. Se procesa el mensaje de solicitud de unión, genera claves de sesión y transfiere NwkSKey y AppSKey al servidor de red y al servidor de aplicaciones, respectivamente. Siempre y cuando se use Over The Air Activation(OTAA) [8].

El servidor de aplicación procesa los mensajes específicos de la aplicación recibidos de los nodos terminales. También genera todos los paquetes del enlace descendente y las envía a los dispositivos finales conectados al servidor de red [8]. Los datos recopilados se pueden presentar de distintas maneras, entre ellas están los dashboards gráficos donde se indican distintos datos enviados por los nodos sensores.

1.4.2.2 Clases LoRaWAN

Los nodos terminales pueden ser usados en distintas aplicaciones que demandan distintos requerimientos [2]. LoRaWAN usa diferentes clases de nodos para dar mayor calidad de servicio según la aplicación destinada. De esta forma se definen tres clases de nodos [1]:

- ❒ Clase A: Su comunicación es dúplex, después de la transmisión de un paquete (Uplink), abren dos ventanas de recepción (Downlink) para aceptar un acuse de recibo (ACK) o aceptar datos del gateway, después se mantiene en modo inactivo hasta la siguiente transmisión. Es la clase más eficiente tomando en cuenta el consumo de energía, pero tiene el mayor tiempo de latencia [1].

- ❒ Clase B: Su comunicación es dúplex y, además de las ventanas de recepción de la clase A, pueden crear ventanas de recepción en horas programadas, ya que no necesitan enviar un paquete Uplink para tener poder recibir datos en el dispositivo final (Downlink) [1].

- ❒ Clase C: Su comunicación es dúplex, y su ventana de recepción (Downlink) se mantiene escuchando de manera continua. Pueden recibir datos los nodos terminales casi todo el tiempo, excepto cuando estos transmiten (Uplink). Los tiempos de latencia son menores, pero implica un mayor consumo de energía con respecto a las clases A y B [1].

1.4.2.3 Seguridad LoRaWAN

LoRa utiliza dos capas de seguridad, la primera se encarga de asegurar la capa física y la siguiente se encarga de dar seguridad a la capa aplicación. Para esta última se utiliza el cifrado AES con intercambio de claves, utilizando un identificador IEEE EUI64 [3].

Para LoRaWAN se han definido 3 claves de seguridad de 128 bits, las mismas tienen las siguientes características [2]:

- ❒ Clave de aplicación AppKey: Se utiliza en el proceso de activación cuando el nodo se une a la red, esta es conocida únicamente por el nodo y por la aplicación. Tras el proceso de activación se generan dos llaves:

- ❒ Clave de sesión de aplicación AppSKey: Es usada para encriptar y desencriptar la carga útil.

- ❒ Clave de sesión de red NwkSKey: Es usada entre el nodo y el servidor, se encarga de verificar que los mensajes sean válidos.

Si en el proceso de activación se usa ABP (Activation by Personalization), las claves son estáticas y deben ser cambiadas por el usuario, mientras que si se usa OTAA (Over-the-air Activation), estas se regeneran cada activación[3].

1.4.3 Wio-E5 Development kit

La Wio-E5 es una placa de evaluación para el módulo Seeed Studio LoRa-E5 STM32WLE5JC. Incluye un módulo de soporte de radiofrecuencia compatible con varios protocolos LPWAN en las bandas de frecuencia de 868/915 MHz, incluido LoRa, con capacidad de salida de 20,8 dBm a 3,3 V [9].

Cuenta con interfaces como Grove y RS-485, y los varios pines GPIO. La placa también incluye un conector de batería JST [9].

Basado en el chip de desarrollo SX126X multimodo de alto rendimiento, el módulo LoRa-E5 admite el modo (G) FSK y el ancho de banda LoRa de 62,5 kHz, 125 kHz, 250 kHz y 500 kHz [10].

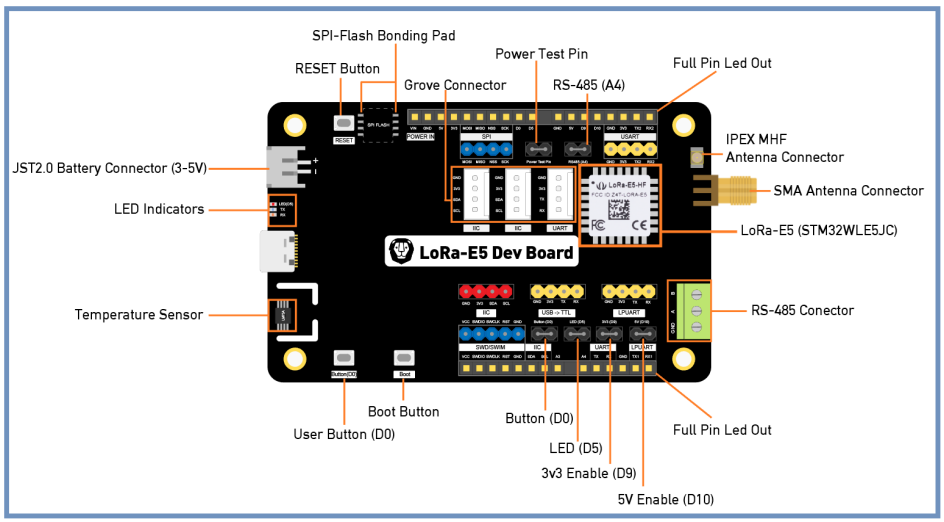

Figura 1.2: Especificaciones de Hardware de la placa de desarrollo Wio-E5 [10]

La placa de desarrollo Wio-E5 tiene un rango de transmisión de larga distancia de hasta 10 km en áreas abiertas. La corriente de reposo puede ser tan baja, tan baja como 2,1 uA (modo WOR). Su temperatura de trabajo va desde los -40 ℃ a los 85 ℃, cuenta con alta sensibilidad entre -116,5 dBm -136 dBm [10].

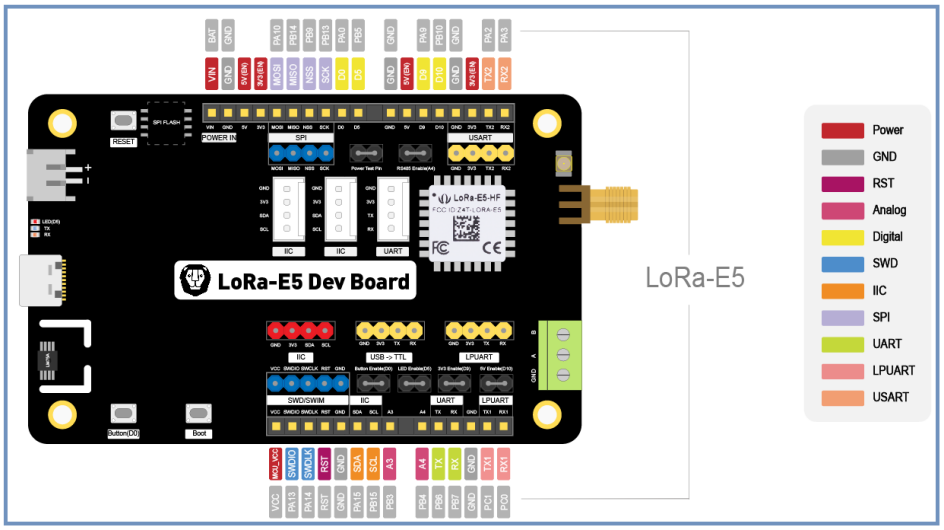

Figura 1.3: Pinout de la placa de desarrollo Wio-E5 [10]

1.4.3.1 Especificaciones

Tabla 1.3: Especificaciones placa de desarrollo Wio-E5 [10]

Parámetros	Especificaciones
Tamaño	85,6*54mm
Voltaje de entrada	3-5V (batería) / 5V (USB tipo C)
Voltaje de salida	ES 3V3 / 5V
Salida de potencia	Hasta +20,8 dBm a 3,3V
Frecuencia	UE868/US915/AU915/AS923/KR920/IN865
Protocolo	LoRaWAN®
Sensibilidad	-116.5 dBm ∼-136 dBm
Interfaces	USB tipo C / JST2.0 / Grove*3(IIC*2/UART*1) / RS485 / SMA-K / IPEX
Modulación	LoRa®, (G)FSK, (G)MSK, BPSK
Temperatura de trabajo	-40 ℃ ∼85 ℃
Corriente mínima	2.1 uA (modo WOR)

1.4.4 Arduino Nano

Es una placa de desarrollo de tamaño reducido, completamente compatible con proto-boards, basada en el microcontrolador ATmega328P. Tiene 14 pines de entrada y salida digital, de estos, 6 pueden ser usando con PWM, además cuenta con 6 entradas analógi-

cas, un cristal de 16 MHz, interfaz Mini-USB, puerto para conexión ICSP y un botón de reset [11].

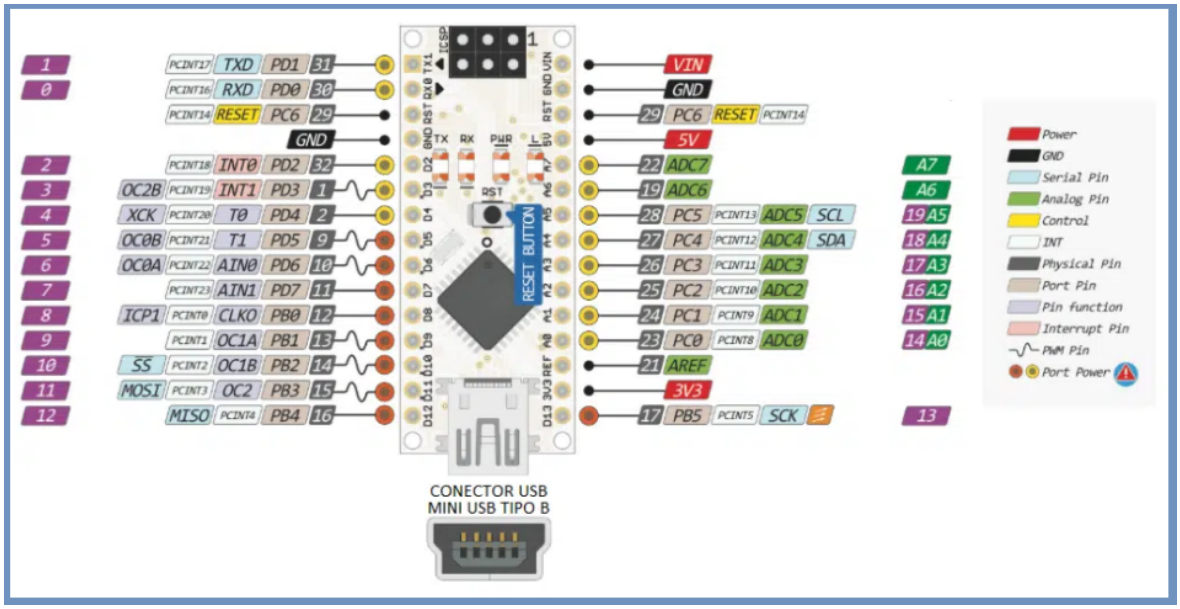

Figura 1.4: Pinout de la placa de desarrollo Arduino Nano [11]

Es bastante similar a un Arduino UNO, tanto en la capacidad del microcontrolador como en comunicación, reduce su tamaño al usar un conector mini-USB tipo B, no cuenta con un conector jack de alimentación y los pines cambian un formato de header [11].

El puerto mini-USB se utiliza tanto para la programación, para la comunicación serial y la alimentación. Una de las características más importantes del Arduino Nano es que puede elegir la fuente de energía más fuerte por su voltaje [12].

1.4.4.1 Especificaciones

- ❒ Microcontrolador: ATmega328p
- ❒ Arquitectura: AVR
- ❒ Voltaje de funcionamiento: 5 V
- ❒ Memoria flash: 32 kB
- ❒ SRAM: 2 kB
- ❒ Clock: 16 MHz

- ❒ Entradas analógicas: 8
- ❒ EEPROM: 1 kB
- ❒ Entradas y salidas digitales: 22
- ❒ Salidas PWM: 6
- ❒ Consumo de corriente: 19 mA

1.4.5 Sentrius RG191

Los gateways LoRa, Sentrius RG1xx de Laird Connectivity, son una solución segura, escalable y robusta para el control extremo a extremo de redes privadas LoRaWAN. cuentan con el bridge inalámbrico WB50NBT, soportan Wi-Fi empresarial de doble banda, BLUETOOTH v4.0 (BLE y Classic) y conectividad Ethernet [13].

Figura 1.5: Gateway Sentrius RG191 [14]

1.4.5.1 Especificaciones

- ❒ Sistema operativo Linux completo [13]
  - ✧ Kernel v4.x en Atmel-A5 Core @ 536MHz.

- ❒ Múltiples interfaces [13]
  - ✧ LoRaWAN: de 863 MHz a 870 MHz (UE) y de 902 MHz a 928 MHz (EE. UU.)
  - ✧ Wi-Fi: 802.11a/b/g/n, 2,4 GHz y 5 GHz
  - ✧ BLUETOOTH v4.0
  - ✧ Ethernet
- ❒ Compatibilidad con LoRaWAN de 8 canales
- ❒ Potencia de transmisión máxima de hasta +27 dBm
- ❒ Seguridad de nivel empresarial
- ❒ Certificaciones y aprobaciones para FCC/IC (RG191), (RG186), ASNZS y NCC
- ❒ Temperatura de trabajo de -30 ℃ a +70 ℃

Cuenta con una fuente de alimentación de 12v a 2A. La antena LoRa debe conectarse al puerto 868/915MHz y las dos antenas Wi-Fi deben conectarse a los puertos 2.4/5.5GHz [14].

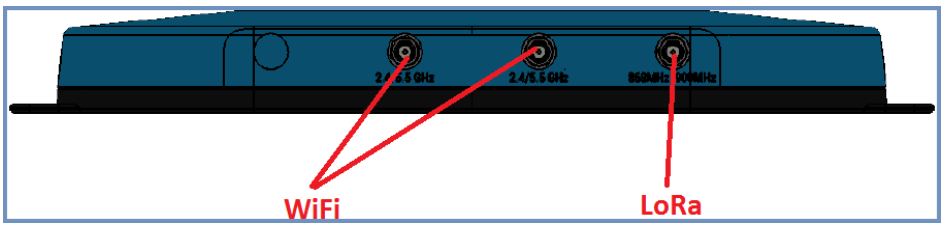

Figura 1.6: Conexión de gateway Sentrius RG191[14]

1.4.6 The Things Network (TTN)

TTN es escalable, seguro y es compatible con los últimos desarrollos de LoRaWAN, como las últimas versiones de LoRaWAN 1.1 y 1.0.4 [15].

La arquitectura Things Stack se basa en microservicios permitiendo una mejor distribución, mejor escalamiento e interoperabilidad con otras redes LoRaWAN [15].

Admite todas las clases de LoRaWAN (A, B, C), grupos de nodos de multidifusión y todos los parámetros regionales definidos por LoRa Alliance. Admite el roaming pasivo. Ofrece actualizaciones de firmware por aire, técnicas avanzadas de agrupamiento y balance de carga [15].

Las APIs avanzadas ofrecen integraciones gRPC, HTTP y MQTT. Se pueden rastrear problemas, monitorear el comportamiento del dispositivo y obtener alertas en modo de depuración. También cuenta con una integración de almacenamiento para los datos recibidos [15].

La conexión a Packet Broker, permite el intercambio de tráfico entre TTN y las redes privadas LoRaWAN, lo que aumenta la cobertura, el rendimiento y la capacidad de la red, además, prolonga la duración de la batería del nodo terminal [15].

1.4.7 Node-Red

Es una herramienta de desarrollo open-source de programación flujos; fue creada por IBM para conectar nodos terminales, APIs y servicios en línea [16].

Node-RED permite conectar gráficamente bloques preprogramados, llamados nodos, para realizar tareas específicas. La conexión de los nodos, independientemente del tipo que estos sean, forman un flujo o flow [16].

Entre los nodos disponibles se pueden encontrar desde protocolos estándar como MQTT, REST, Modbus, OPC-UA, Bacnet, Websocket, etc. hasta integraciones a APIs externas como Microsoft Azure, Amazon Web Services, Twitter, Facebook, etc. [16].

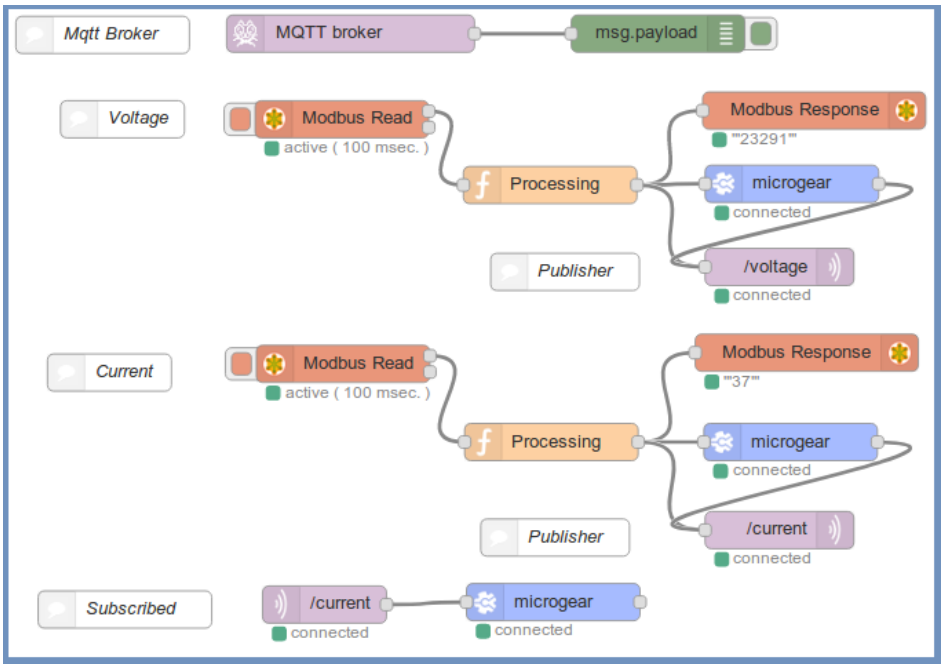

Figura 1.7: Ejemplo de flujo en Node-Red[16]

2 METODOLOGÍA

2.1 PROBLEMA A RESOLVER

Con el constante crecimiento de las redes inalámbricas de nodos sensores, se ha vuelto una necesidad el conocer el alcance real o el radio de cobertura de los mismos, sin embargo, este depende de muchos factores que se encuentran en constante cambio.

Es por esto que se vuelve indispensable el contar con un sensor integrado que permita conocer el RSSI, ya que según el dispositivo a usarse, el fabricante proporciona un nivel de sensibilidad en dBm, que permitirá tener un valor de referencia para poder calcular y validar el alcance de un dispositivo a desarrollarse.

De esta manera, en búsqueda de resolver esta problemática, en el presente trabajo de integración curricular se presenta el diseño e implementación de un nodo sensor de nivel de energía de la señal recibida, tanto en software como en hardware.

2.2 ANÁLISIS DE REQUISITOS DE LA RED

Según lo revisado anteriormente, para conformar una red LoRaWAN se debe asegurar un rango de largo alcance, trabajar con bajas velocidades, obtener bajos niveles de SNR, mantener una sensibilidad muy alta trabajando con niveles bajos de RSSI, y asegurar cierta calidad de servicio y condiciones de seguridad para los parámetros antes descritos.

De esta manera se definen requerimientos mínimos y máximos para diseñar la red con la que se realizarán las pruebas posteriores, entre estos se define la banda de frecuencias, el factor de Spreading, la potencia de transmisión del nodo, la sensibilidad del nodo, la distancia máxima de cobertura, entre otros.

2.2.1 Banda de frecuencias

Debido al modelo del gateway, RG191, se utilizarán las especificaciones definidas para USA, entre estas se ha elegido la banda US915, ya que es la banda compatible con el nodo

Wio-E5 [17].

La banda US915 presenta 16 modos distintos de Data Rate (DR) como se pueden ver a continuación en la figura 2.1.

2.2.2 Factor de Spreading

Se ha decidido usar un factor de Spreading de 10 chirps, debido a que este cumple con las especificaciones para la banda de frecuencias US915 y es totalmente compatible con las capacidades del gateway usado, además, por su buen rendimiento es el factor predeterminado usado por el nodo, el mismo se encuentra en el modo DR0 como se muestra en la figura 2.1.

Como información adicional, en la figura 2.1 se muestra que se tendrá un ancho de banda de 125 kHz, además, como se muestra en la tabla 1.2 se espera tener una velocidad máxima de 980 bps y una sensibilidad máxima de -132 dBm.

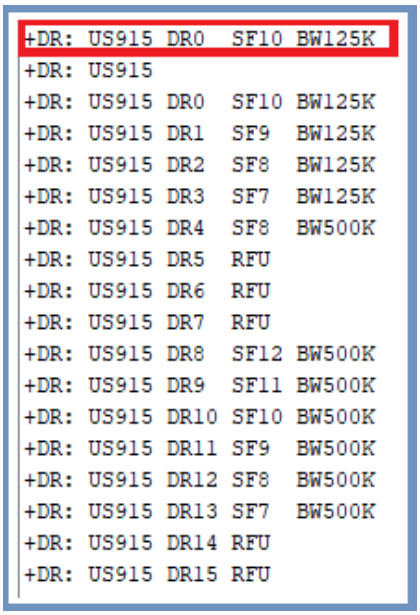

Figura 2.1: Modos de DR para la banda US915 en el nodo Wio-E5

2.2.3 Potencia de transmisión del nodo

Como se visualiza en la figura 2.2, con un voltaje suministrado mayor a 3.3V se tendrá una potencia, una salida esperada de 20.5 dBm, siendo este el valor principal de interés del diseño a realizarse.

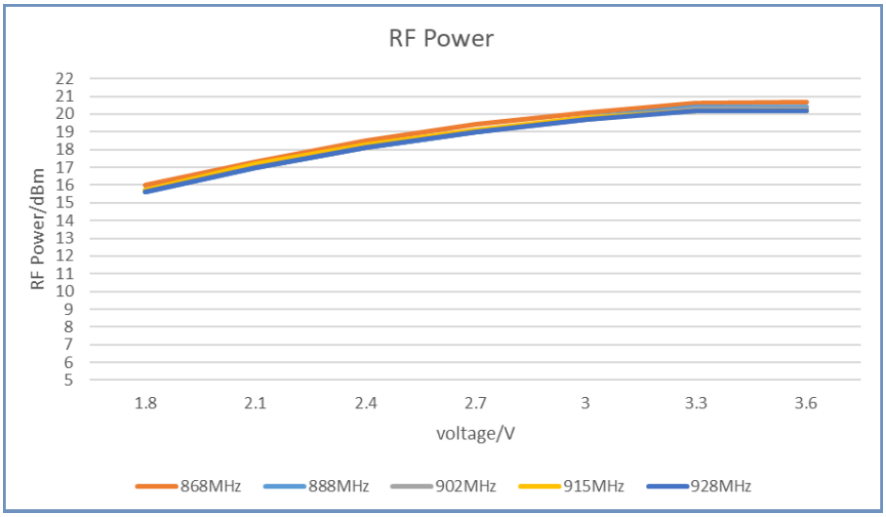

Figura 2.2: Potencia de salida (dBm) vs Voltaje de entrada (V) [17]

2.2.4 Sensibilidad

Tomando en cuanta la banda de frecuencias US915 y el factor de spreading utilizado, se asume una sensibilidad aproximada de -129 dBm lo que, como se verá en los siguientes puntos, servirá para el cálculo de las distancias de cobertura.

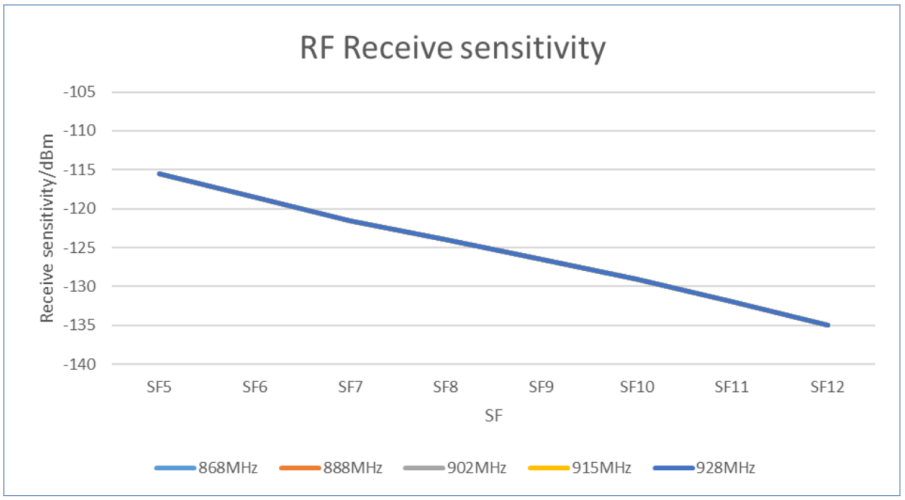

Figura 2.3: Sensibilidad (dBm) vs Factor de spreading [17].

2.2.5 Distancia máxima de cobertura

Para el cálculo de distancias de cobertura se utilizará varios métodos de cálculo y se hará uso de varias simulaciones para obtener datos que asemejen un escenario real haciendo uso de MATLAB.

2.2.5.1 Por Ecuación de Friis

Con la ecuación de Friis (2.1) se puede obtener la distancia máxima de cobertura usando la sensibilidad (-129 dBm) como potencia del receptor, la potencia del emisor se toma como 27 dBm [9], la ganancia de la antena del transmisor: 2 dBi [13], La ganancia de la antena del receptor: 3 dBi [18], la velocidad de la luz: 3x10^8 m/s y la frecuencia central: 915 MHz, además se consideró una pérdida inicial aproximada de 40 dBm, valor que se ha obtenido de manera empírica tras el uso de la placa Wio-E5.

$$P_{Rx}(dBm) = P_{Tx} + G_{Tx} + G_{Rx} + 20Log_{10}\left(\frac{c}{4\pi D_r F_0}\right) - 40 \tag{2.1}$$

Al despejar D_r de la ecuación 2.1 se obtiene que la distancia máxima para un RSSI de -129 dBm es $D_r \approx 29km$.

Tras usar MATLAB para obtener las distancias haciendo uso de la ecuación 2.1 en conjunto con los datos descritos anteriormente, se ha obtenido los siguientes resultados.

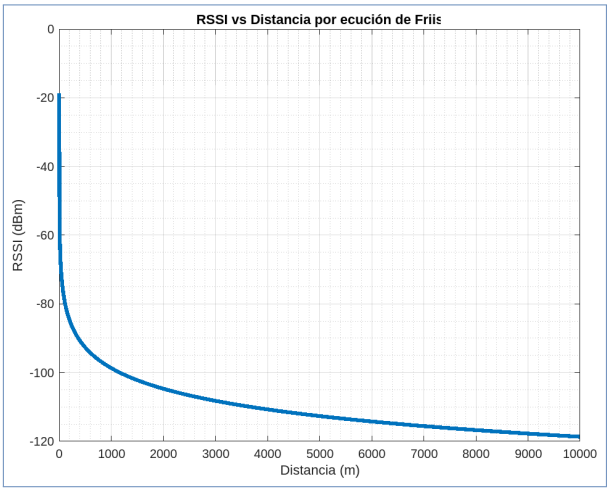

Figura 2.4: Sensibilidad (dBm) vs Distancia (m) por Ecuación de Friis

Como se puede observar en la figura 2.4 teóricamente, si se considera un canal con una sola perdida inicial, las distancias alcanzadas son sumamente altas, sin embargo, como no existe dicho canal, se han implementado dos modelos de canal distintos para establecer un rango de distancias para las pruebas.

El código utilizado para obtener la figura 2.4 se encuentra disponible en el Anexo 1.

2.2.5.2 Modelo Log-Distance

Para esta simulación se basó en el modelo log distance optimizado tras la obtención de datos reales para distintas frecuencias [19], de esta manera el valor de la distancia máxima para la sensibilidad dada es $D_r \approx 2.35Km$, como se puede observar en la figura 2.5.

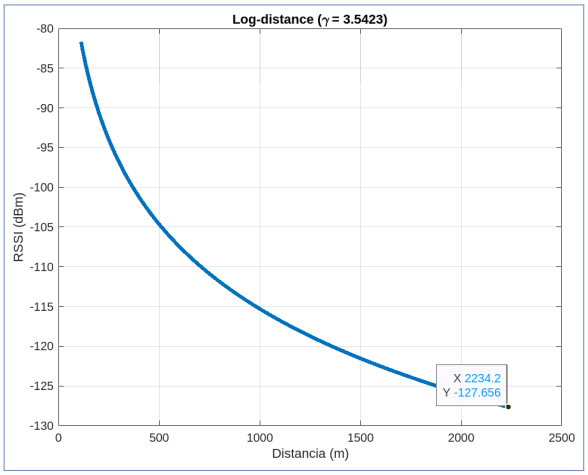

Figura 2.5: Sensibilidad (dBm) vs Distancia (m) por modelo Log-Distance, basado en [19]

2.2.5.3 Modelo Okumura-Hata

También se realizó la prueba haciendo uso del modelo Okumura-Hata [20] para entornos urbanos y una antena de transmisión ubicada a 30m de altura con una antena de recepción ubicada a 1.5m de altura, se ha usado las mismas distancias de prueba con la misma frecuencia central de 915 MHz, de esta manera se obtuvo una distancia máxima $D_r \approx 1.26Km$, como se puede observar en la figura 2.6

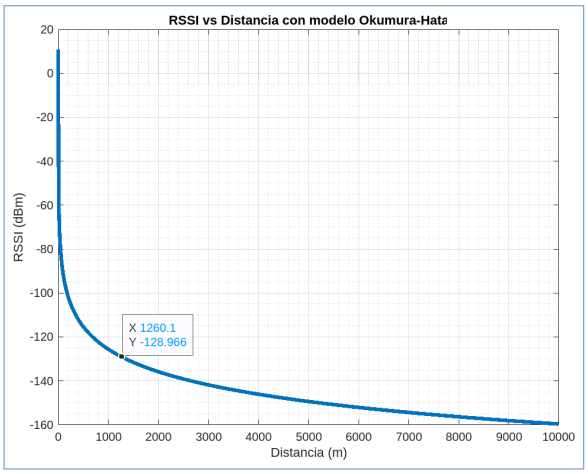

Figura 2.6: Sensibilidad (dBm) vs Distancia (m) por modelo Okumura-Hata, basado en [20]

Para el diseño de la red de sensores inalámbricos se usará 5 elementos principales, que no solo permitirán la realización de este proyecto de integración curricular, sino que servirá también para trabajos futuros debido a la posibilidad de añadir sensores y de utilizar librerías especializadas para el prototipo.

2.2.6 Red de sensores inalámbricos

En el diseño se plantea utilizar un controlador (Placa Arduino Nano) conectado a un sensor (Sensor de nivel de energía de señal integrado en la placa Wio-E5), el controlador procesará la información obtenida y la comunicará a un transceiver (Módulo Lora-E5 presente en la placa Wio-E5), este envía la información procesada en formato hexadecimal al gateway (Gateway Sentrius RG191) que está conectado a un servidor de red (Consola de TTN), este servidor a su vez envía los datos mediante el protocolo MQTT a un dashboard (Aplicación desarrollada en Node-Red).

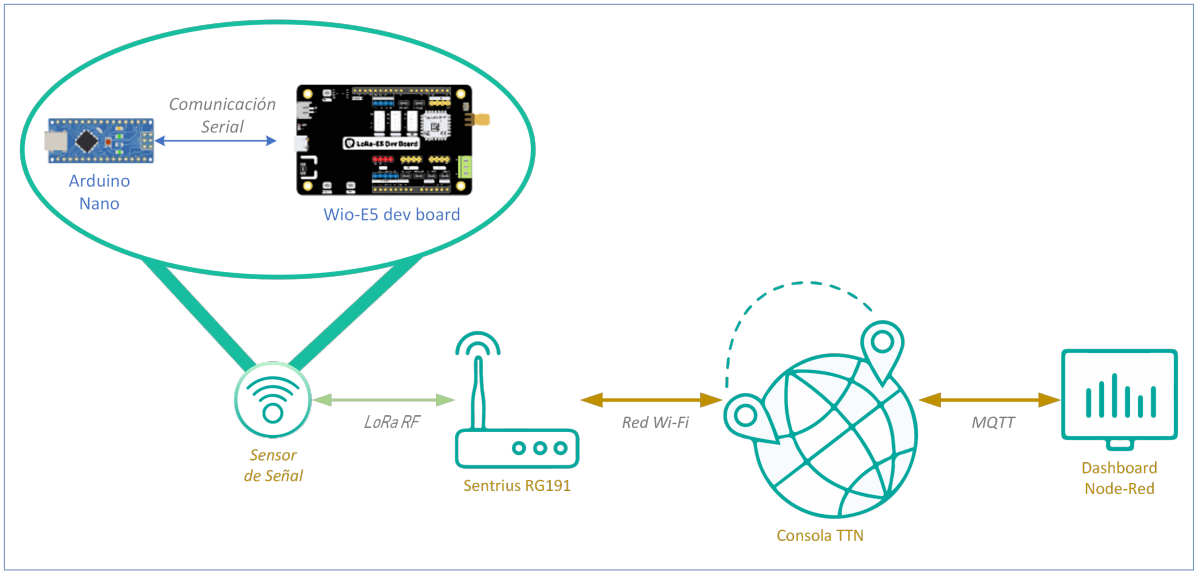

Figura 2.7: Red de sensor de nivel de señal diseñada

2.3 JUSTIFICACIÓN DE COMPONENTES Y PLATAFORMAS

2.3.1 Arduino Nano

Se decidió usar la placa Arduino Nano debido a la facilidad de programación, el enorme repositorio y disponibilidad de librerías que ofrece la comunidad de Arduino CC y el tamaño

reducido que esta ocupa.

Además, se sacó provecho de la librería *Software Serial* para implementar un segundo comunicador serial en los pines D6(Rx) y D7(Tx) del controlador, de esta manera este puede hacer uso de la comunicación serial con el computador a través del puerto mini USB tipo B, al mismo tiempo que se comunica con el primer puerto serie (UART) de la placa Wio-E5 a través del comunicador definido por la librería.

2.3.2 Wio-E5 Development Kit

Se ha escogido el kit de desarrollo Wio-E5 debido a la robustez de su construcción, además cuenta con varios puertos de comunicación cono, UART, I2C, LPUART, SPI, entre otros, de esta manera se utiliza uno de los puertos de comunicación serial (UART) para realizar la conexión con la placa Arduino Nano. El Wio-E5 también cuenta con varios pines analógicos de 10 bits, además de los sensores de nivel de energía de la señal y SNR internos presentes en el módulo Lora-E5, que son fácilmente accesibles desde los comandos AT que se activan desde el puerto de comunicación serial UART [21].

El diagrama de conexión de la placa Arduino Nano con la placa Wio-E5 se presenta en la figura 2.8.

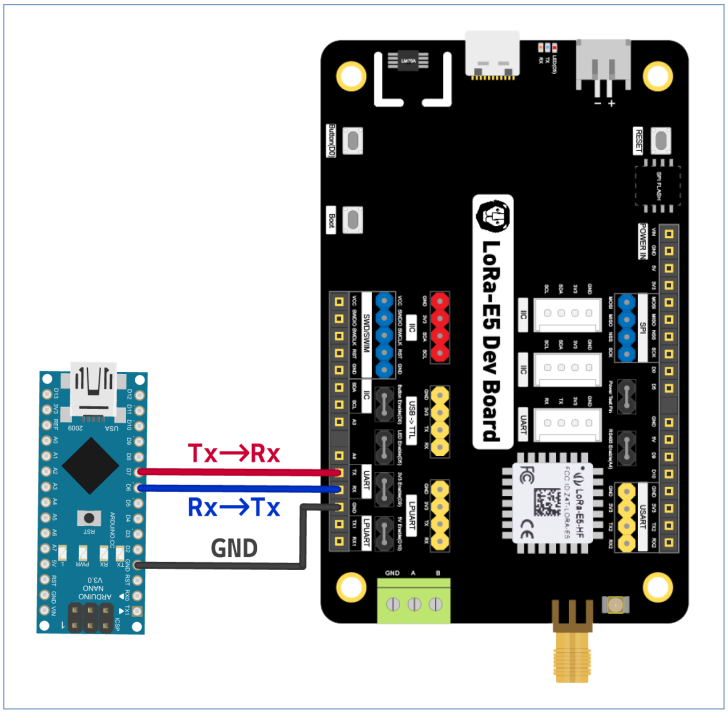

Figura 2.8: Comunicación serial entre Arduino Nano y Wio-E5

2.3.3 Sentrius RG191

Se ha escogido el gateway de Laird Sentrius RG191 debido a la libertad de diseño que este brinda, ya que cuenta con una interfaz Wi-Fi de nivel empresarial de doble banda y una interfaz Ethernet como opciones principales para la red de backhaul, también cuenta con una interfaz Bluetooth v4.0 BLE para diversos tipos de funciones adicionales [14].

Existe también la versión con certificación IP67 para funcionar en exteriores, sin embargo, para la implementación de este trabajo de integración curricular se usará la versión normal con una antena LoRa para interiores (2 dBi) [18] y una antena LoRa para exteriores (12 dBi) [22], conectado a una red Wi-Fi residencial que le permitirá el acceso a la consola del servidor de red montado en The Things of Network (TTN).

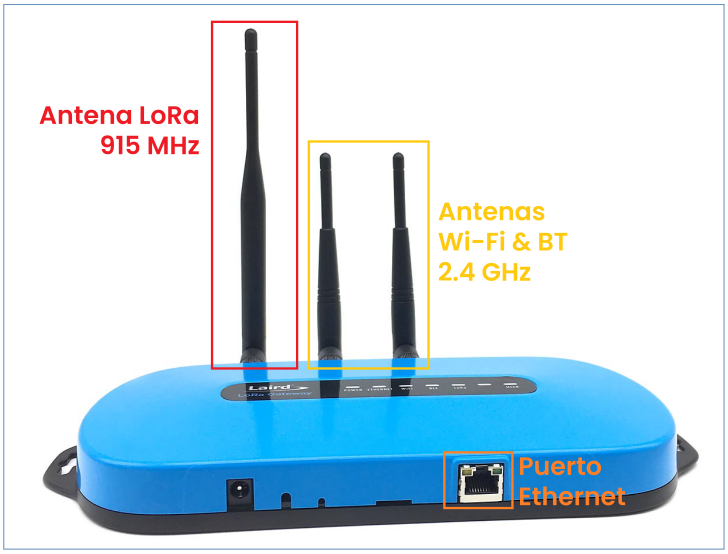

Figura 2.9: Interfaces del Sentrius RG191

Se ha usado la construcción del gateway virtual y la API implementadas en trabajos anteriores, sin embargo, se ha creado una nueva aplicación para la conexión del nodo mediante OTAA.

2.3.4 The Things of Network

Se ha decidido usar la plataforma TTN y su integración TTS (The Things Stack) como servidor de red privado, debido a la facilidad de uso de los mismos, sus servicios se prestan de manera gratuita, cuenta con una amplia comunidad que continuamente presenta avances,

guías y tutoriales para distintos modelos de gateways y nodos, por su compatibilidad con el gateway Sentrius Rg191, además de que permite usar diversas integraciones que permiten mejorar la experiencia de usuario en el servidor aplicación.

Para la implementación se ha creado una aplicación en la consola de TTN en donde se encuentra registrado el nodo terminal y donde se muestran los datos en tiempo real, además se ha agregado un payload formatter para la correcta decodificación de los datos, y se ha hecho uso de la integración MQTT para crear un servidor que use el protocolo para comunicarse con la aplicación de dashboard desarrollada en Node-Red.

2.3.5 Node-Red

Se decidió usar Node-Red como servidor aplicación por la facilidad de programación mediante flujos y la posibilidad de crear dashboard dinámicos a partir de la captura de datos desde un servidor MQTT, además de la posibilidad de usar librerías de Node JS y editar el front-end de la aplicación mediante codificación con Angular JS.

Para la implementación del servidor se ha usado una máquina virtual con la versión LTS actual de Ubuntu, la misma se encuentra conectada a la red por un adaptador puente desde la VM y figura como el servidor aplicación dentro de la red local, para trabajos futuros se puede utilizar una máquina virtual en un servicio de virtualización online como Azure o AWS para tener acceso al dashboard desde cualquier parte del mundo.

2.4 IMPLEMENTACIÓN

Para iniciar con la implementación se definieron 4 etapas usando cada uno de los componentes antes descritos para la red diseñada. Primero se realizó una investigación sobre los comandos AT que pueden usarse con la placa Wio-E5, obteniéndose varias opciones que puedan indicar los datos buscados para la obtención del nivel de energía de la señal, las mismas que serán descritas más adelante.

Seguido se implementaron varias funciones propias en Arduino para conseguir la comunicación con la placa Wio-E5, la obtención de datos, el procesamiento de los datos y el envío de los datos hacia el gateway.

Una vez se lograron transmitir los datos se creó una aplicación en la consola de TTN que

no solo permitía el acceso mediante OTAA, sino que también realiza la interpretación de los datos recibidos en formato hexadecimal y los muestra en pantalla en tiempo real, estos datos son enviados desde la integración MQTT.

Estos datos son recibidos por un flujo MQTT Receiver en Node-Red y son filtrados por funciones propias programadas en JavaScript, después estos son mostrados a través de nodos dashboard que permiten una fácil lectura e interpretación para el usuario.

2.4.1 Comandos AT Wio-E5

Para la implementación realizada en el presente trabajo se ha dividido los comandos AT en dos categorías, según el fin de los mismos, estas son comandos para configuración y comandos para obtención y envío de datos.

2.4.1.1 Comandos para configuración

En esta categoría entran todos los comandos usados para configurar la comunicación, ya sea desde la banda de frecuencias usadas hasta el tipo de acceso que se tendrá como se lista a continuación.

- **AT:** Este es un comando de test para probar si se está comunicando con la placa Wio-E5, en caso de que la comunicación serial sea efectiva se tendrá la respuesta +AT: OK [21].
- **AT+ID:** Este comando permite obtener las diferentes credenciales del nodo, entre estas están el DevAddr, el DevEui y el AppEui [21]. Al ejecutarse se obtiene la respuesta:
  +ID: DevAddr, 26:0C:CB:0C
  +ID: DevEui, 2C:F7:F1:20:32:30:4D:E1
  +ID: AppEui, 80:00:00:00:00:00:00:06
- **AT+DR:** Con este comando se pueden definir las velocidades de comunicación [21] y todos los datos que intervienen en la velocidad del enlace, como la banda de frecuencias, el factor de spreading, etc.
  - **AT+DR=US915:** A través de este comando se setea la banda de frecuencia US915 [21], al ejecutarse se obtiene la respuesta +DR: US915.

- ✧ **AT+DR=DR0:** Con este comando se establece un factor de spreading de 10 y un ancho de banda de 125 kHz [21]. Los distintos DRx para configuración se pueden observar en la figura 2.1. Al ejecutarse se obtiene la respuesta:
  +DR: DR0
  +DR: US915 DR0 SF10 BW125K

- ❒ **AT+CH=NUM,0-15:** Con este comando se habilitan los primeros 15 canales de comunicación, para la banda usada se puede tener un máximo de 96 canales si se habilitan todos a la vez [21]. Al ejecutarse se obtiene la respuesta +CH: NUM, 0-15.

- ❒ **AT+CLASS=A:** A través de este comando se escoge la clase del nodo, estas pueden ser A, B o C [21]. Al ejecutarse se obtiene la respuesta +CLASS: A.

- ❒ **AT+MODE=LWOTAA:** Este comando se utiliza para entrar en el modo LWOTAA que es el que permite el acceso vi OTAA del nodo al gateway; también existen los modos LWABP, para entrar en modo ABP, y el modo test, que permite entrar en modo de prueba para conexión punto a punto [21]. Al ejecutarse se obtiene la respuesta +MODE: LWOTAA.

- ❒ **AT+JOIN:** A través de este comando se inicia el acceso y se abre la comunicación con el gateway [21]. Al ejecutarse se obtiene la respuesta:
  +JOIN: Start
  +JOIN: NORMAL
  +JOIN: Network joined
  +JOIN: NetID 000013 DevAddr 26:0C:CB:0C
  +JOIN: Done

2.4.1.2 Comandos para obtención y envío de datos

En esta categoría se listan los comandos a usarse para obtener el nivel de RSSI funcionales y los que se intentaron usar, pero no funcionaron, también se encuentran los comandos que se usan para enviar los datos al gateway.

- ❒ **AT+TEST = RSSI:** Este comando permite obtener el RSSI del nodo en el modo TEST [21], en principio se trató de utilizar este comando para medir el RSSI del nodo durante la conexión con el gateway, sin embargo, el comando funciona únicamente dentro del

modo TEST, mismo que no permite la transmisión de datos entre el nodo y el gateway, por lo que se terminó descartando el uso del comando

- **AT+MSG:**Este comando generalmente es usado para enviar un mensaje en formato de texto plano [21], sin embargo, cuando no se le añade contenido el mismo busca un acuse de recibo que proporciona información sobre el enlace de comunicación, entre estos datos se encuentran el RSSI y la SNR medidas en el nodo, por lo que se ha decidido usar este comando para la obtención del nivel de energía de la señal. Al ejecutarse se obtiene la respuesta:
  +MSG: Start
  +MSG: TX
  +MSG: Link 20, 1
  +MSG: RXWIN1, RSSI -93, SNR 6.25
  +MSG: Done

- **AT+MSGHEX="Mensaje en hexadecimal":** Este comando es usado para realizar el envío de datos en formato hexadecimal, en el presente trabajo de integración curricular es el método escogido para el envío de datos, ya que permite enviar las cadenas de texto de manera que los caracteres especiales no sean eliminados o generen errores en su llegada al gateway, ya que se envía los datos en hexadecimal y en el gateway se procesan y se traducen estos datos a JSON. Al ejecutarse se obtiene la respuesta:
  +MSGHEX: Start
  +MSGHEX: Done

Se ha realizado la revisión de cada uno de los comandos AT a utilizarse en la implementación del prototipo, ya que se ha decidido usar estos para evitar quemar un nuevo firmware para poder programar la tarjeta Wio-E5

2.4.2 Consola The Things of Networks

Para la implementación en TTN se han realizado dos procesos con el fin de registrar tanto el gateway como el nodo en una red privada LoRaWAN, ambos procesos son necesarios debido a si algún componente no se encuentra registrado la comunicación no será efectiva.

2.4.2.1 Consola

Primero se debe ingresar al la página https://www.thethingsnetwork.org/ e iniciar sesión para poder acceder a la consola, el crear y mantener un usuario en TTN es totalmente gratuito. Una vez dentro se hace clic sobre el perfil del usuario y se selecciona Console.

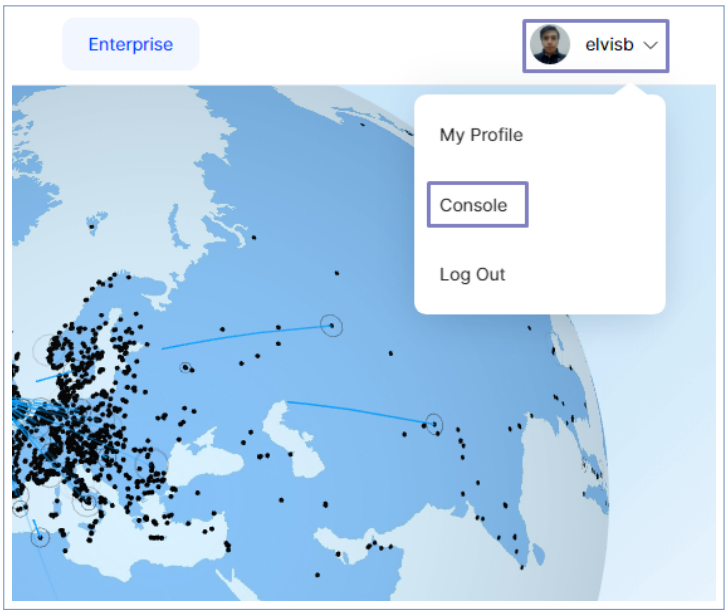

Figura 2.10: Acceso a la consola de TTN

En la nueva ventana se puede seleccionar el país en el que se realizan los desarrollos, y se puede seleccionar uno de los 3 clústeres, los mismos tienen 3 ubicaciones distintas alrededor del mundo, estas son *Europa*, ***Norteamérica*** y *Australia,* entre estas, tanto por la cercanía como por la banda de frecuencias a usarse se selecciona Norteamérica.

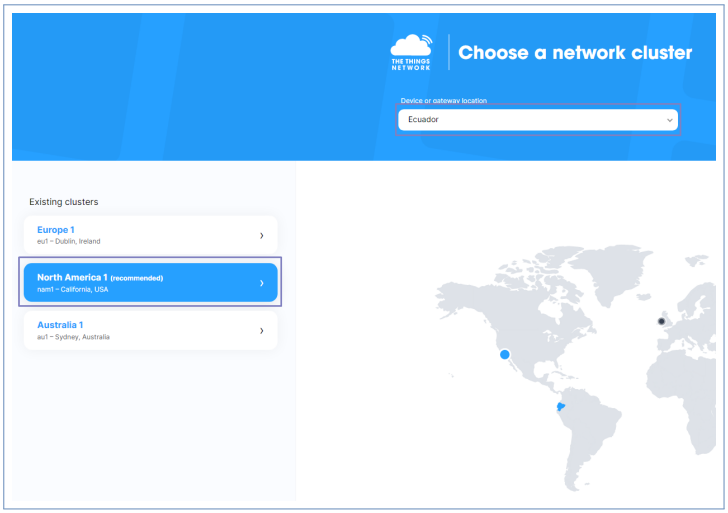

Figura 2.11: Selección de país y ubicación del cluster

Ya dentro de la consola se puede configurar, tanto los gateways como las aplicaciones, cabe recalcar que cada gateway puede registrarse una sola vez en TTN, si se ha registrado con otra cuenta al estar enlazado a la MAC no se podrá registrar nuevamente hasta que este sea borrado de la cuenta previa; en las aplicaciones se puede registrar varios nodos terminales, y estos a su vez pueden estar registrados en varias aplicaciones distintas.

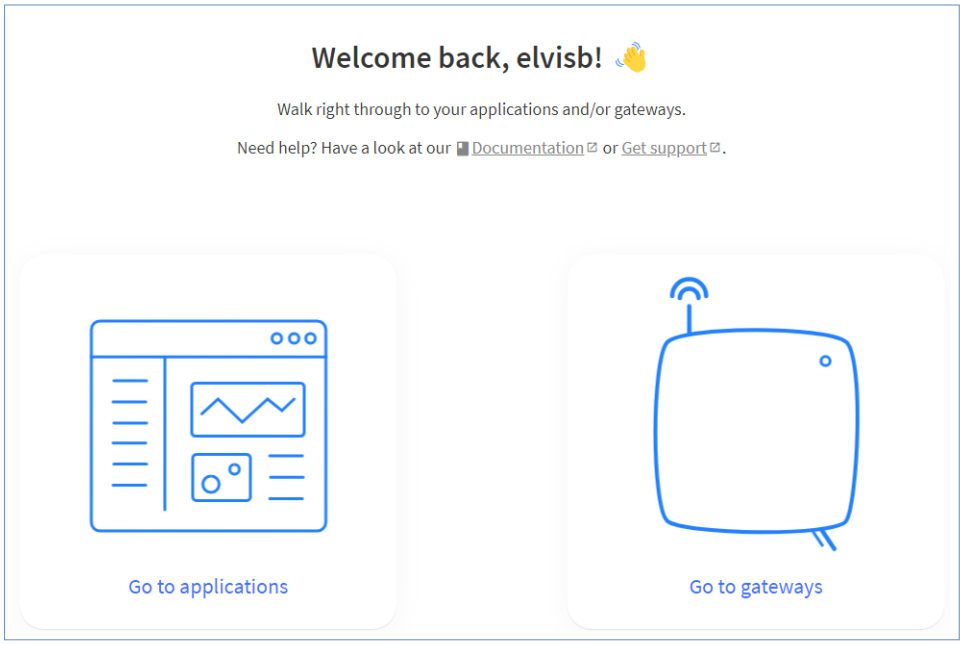

Figura 2.12: Selección entre gateway o aplicación

2.4.2.2 Registro de Gateway

Al seleccionar *Go to gateways* se redirige a una nueva ventana en donde se listan los gateways registrados y se permite registrar nuevos gateways.

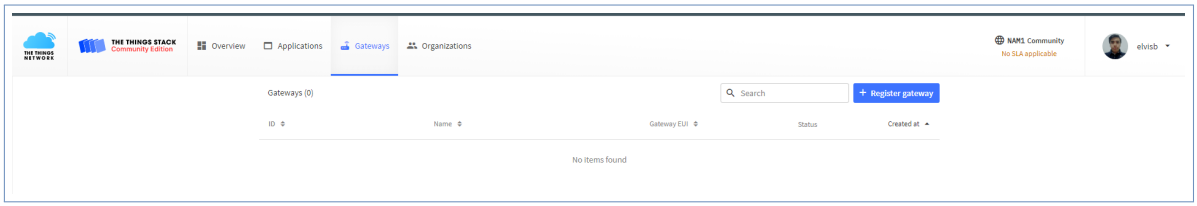

Figura 2.13: Pestaña Gateways

Para registrar un nuevo gateway se da clic en el botón *Register gateway*, y en la ventana emergente se nos pedirá incluir nuestro Gateway EUI, que es un código hexadecimal de 8 bytes, para continuar con el proceso, el mismo lo podemos encontrar en la cara inferior del equipo, incluido como el apartado DevEUI como se muestra en la figura 2.14.

Figura 2.14: Gateway EUI

En cuanto se ha ingresado y confirmado el gateway EUI se pide establecer un identificador único (Gateway ID) y establecer un plan de frecuencias (US915), además de que permite llenar más opciones no obligatorias tal y como se muestra en la figura 2.15.

Figura 2.15: Registro de gateway

En cuanto se han ingresado todos los datos correctamente, se da clic en Register gateway para finalizar el proceso.

2.4.2.3 Creación de Aplicación

Para crear la aplicación se debe seleccionar Go to aplicacions en la ventana de la figura 2.12, seguido se ingresará a una ventana similar a la de la pestaña gateways con la diferencia que en lugar de listar los gateways registrados se lista las aplicaciones creadas.

En esta ventana se da clic en el botón *Create application* que nos llevará a una nueva ventana en donde se nos pide incluir un identificador único para la aplicación (Application ID), además de poder llenar opcionalmente los campos de nombre y descripción. Si el nombre de la aplicación se deja en blanco, esta utilizará el mismo nombre que el ID de la aplicación de forma predeterminada.

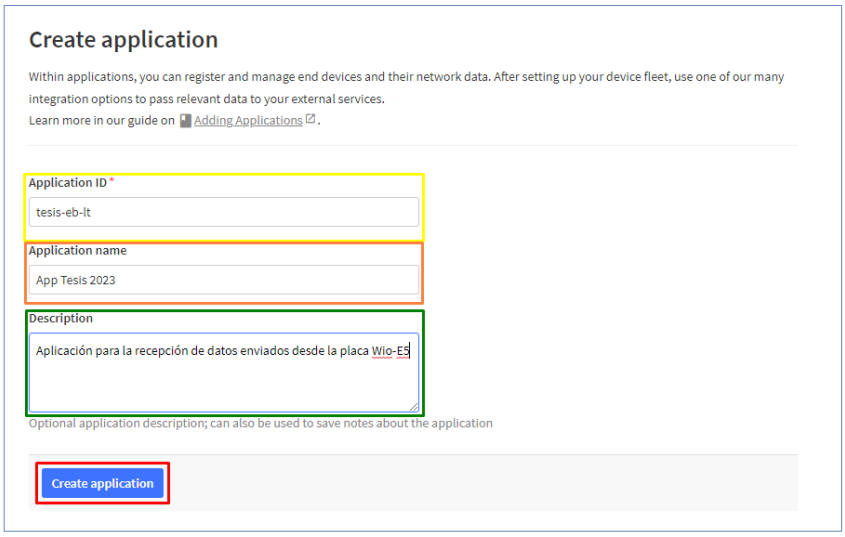

Figura 2.16: Creación de aplicación

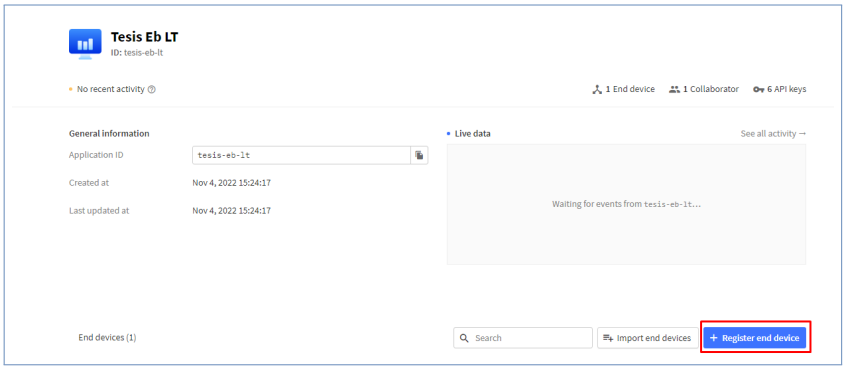

Figura 2.17: Aplicación creada

Con la aplicación creada se puede iniciar el proceso de agregar nodos terminales para comenzar con el envío de datos. Para esto se da clic en el botón *Register end device* y en

la ventana emergente se selecciona el plan de frecuencias (US915), seleccionar LoRaWAN Specification 1.0.2 como versión de LoRaWAN y RP001 Regional Parameters 1.0.2 revisión B como versión de parámetros regionales, tal y como se muestra en la figura 2.18

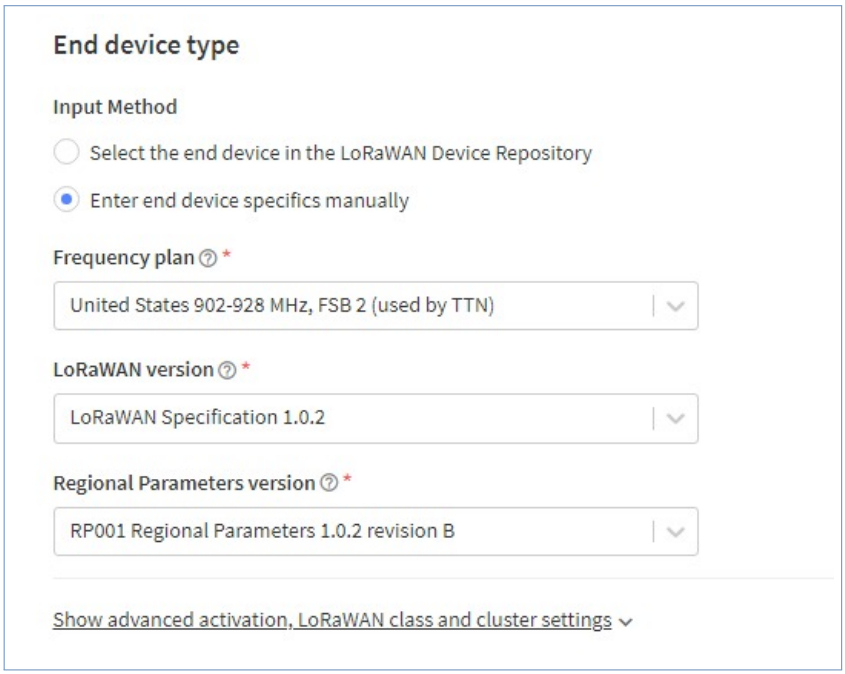

Figura 2.18: Configuración de nodo Wio-E5

A continuación nos pedirá ingresar los campos DevEUI, AppEUI y AppKey. Estos campos pueden ser obtenidos a través del comando ***AT+ID*** en la consola serial de la placa Wio-E5, los mismos serán otorgados en el formato mostrado en la figura 2.19.

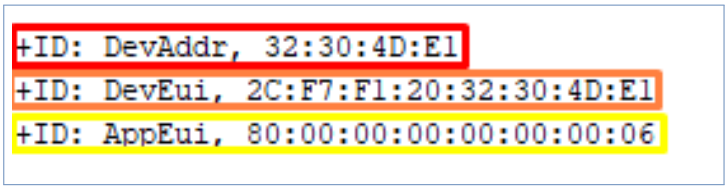

Figura 2.19: Resultado del comando AT+ID

Ya con los campos requeridos se procede a copiar y pegar la información anterior en los campos DevEUI , AppEUI y AppKey. El campo de ID del dispositivo final se completará automáticamente cuando completemos DevEUI. Finalmente, hacer clic en Registrar dispositivo final como se muestra en la figura 2.20.

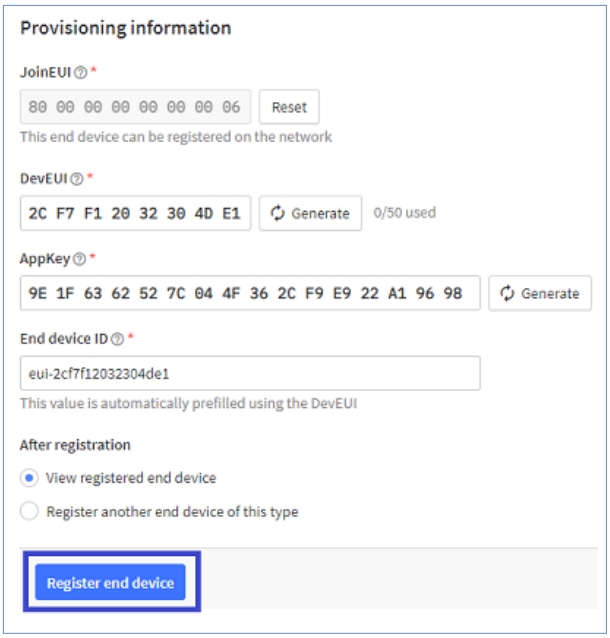

Figura 2.20: Registro del nodo Wio-E5

Con el nodo registrado se podrán hacer dos configuraciones extra para el correcto funcionamiento de esta implementación, la primera es generar un decodificador y JSON Parser para los mensajes de llegada al GW, y la segunda será habilitar un servidor MQTT a través de las integraciones para comunicarse con la aplicación desarrollada en Node-Red. Ambas configuraciones pueden realizarse accediendo en la barra lateral izquierda a las opciones: *Payload formatters ->Uplink* e *Integrations ->MQTT*, como se muestra en la figura 2.21.

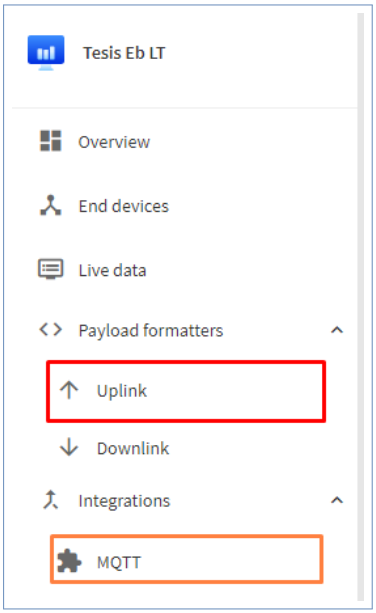

Figura 2.21: Acceso a configuraciones extra

2.4.2.4 Uplink formatter

El decodificador utilizado en el Uplink formatter es el Custom JavaScript formatter por defecto, el mismo permite mostrar en pantalla los datos obtenidos por un parser que lee un String y entrega un JSON.

```
function Decoder(bytes, port) {
  var deco = {};
if (port==8){
 return {
      Values: JSON.parse(String.fromCharCode.apply(null, bytes)) //Json parser
  };
  }
  return deco;
}
```

Código 2.1: Uplink formater

2.4.2.5 MQTT Integration

Para usar esta integración basta con ingresar un usuario y generar una nueva API key, ya que el servidor MQTT de la integración se encuentra habilitado en los puertos 1883, para acceso público, y 8883, para acceso mediante TLS, como se muestra en la figura 2.22. Todas las API Keys generadas son visibles para el usuario, estas pueden ser usados en elementos de terceros para la obtención de datos, puntualmente la API key generada será usada en Node-red para acceder al servidor MQTT TLS y adquirir los datos en formato.

Figura 2.22: Configuración del uplink formatter type

2.4.3 Aurdino

El algoritmo planteado se basa en la constante comunicación serial entre el Arduino Nano y la Wio-E5, donde a través de acuses de recibo de mensajes piloto enviados hacia el gateway, el Arduino puede procesar las cadenas de texto recibidas y obtener el RSSI, una vez con el dato se compone un mensaje en formato JSON y se lo convierte a hexadecimal para su envío al GW, dicho proceso se describe en la figura 2.23.

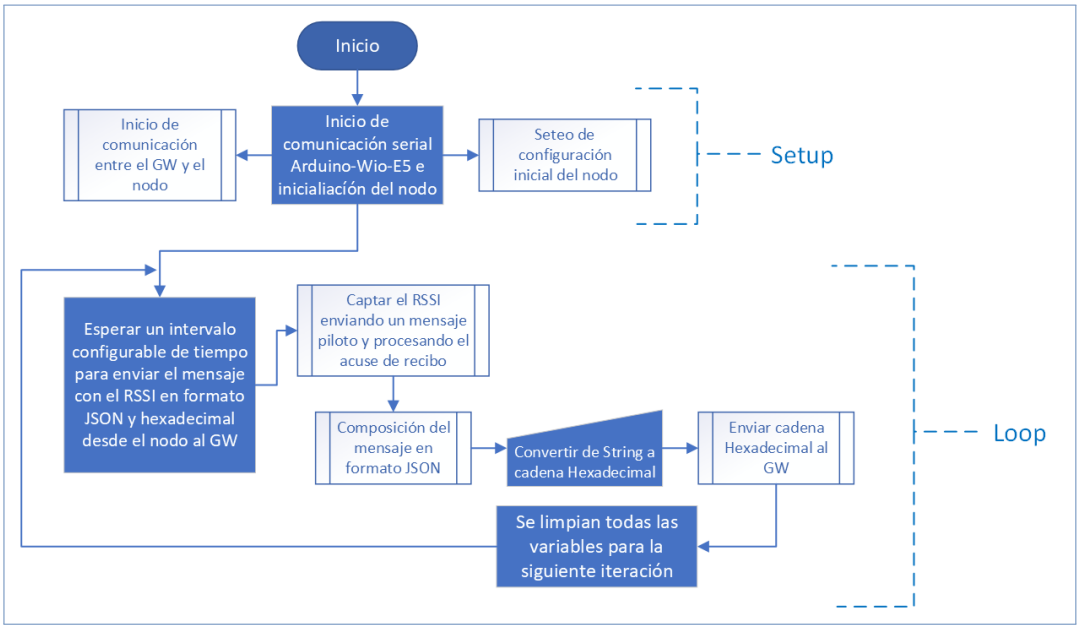

Figura 2.23: Diagrama de flujo general

Para conseguir la correcta ejecución del programa se ha dividido el mismo en varias funciones que se usan entre sí y se engloban en un solo subproceso llamado desde el loop principal.

Para esto se listará en orden cronológico de como fueron desarrolladas y donde fueron utilizadas para el funcionamiento de todo el algoritmo.

2.4.3.1 Función enviarMensaje

Con esta función se intercambia mensajes en sentido Arduino Nano a Wio-E5 a la vez que se imprime en el monitor serial de Arduino el mensaje que se enviará al nodo LoRa seguido se captura la respuesta enviada en sentido Wio-E5 a Arduino Nano y se muestra en el monitor serie dicha respuesta. Cuenta con un argumento de entrada de tipo String y no

cuenta con argumentos de salida al ser una función de tipo Void.

```
void enviarMensaje(String msj){ //Funcion para enviar un mensaje arduino-wio
  Serial.write("Tx:"); //Se muestra en pantalla
  Serial.println(msj); //el mensaje a enviar
  e5.println(msj); //Envio de mensaje desde arduino a wio
  delay(2000); //Tiempo de espera para comunicacion efectiva
  while(e5.available()){ //Lectura de buffer
    char inChar = (char)e5.read(); //Lectura char a char
    inputString += inChar; //Se compone el string char a char
  }
  Serial.print(inputString); //Se muestra en pantalla el string
  inputString = ""; //Se limpia el string
}
```

Código 2.2: Función de envío de mensaje serial Arduino-WioE5

2.4.3.2 Función initE5

Esta función aprovecha la funcionalidad de ***enviarMensaje*** para setear las configuraciones básicas requeridas para que el nodo LoRa pueda conectarse al gateway correctamente, entre estas se encuentran la banda de frecuencias, el SF, el número de canales habilitados, la clase del nodo, el método de activación, etc. Además, inicia comprobando que la comunicación serial entre el Arduino Nano y la Wio-E5 sea efectivo, seguido indica las credenciales de acceso del nodo. Esta función no cuenta con argumentos de entrada ni de salida.

```
void initE5(){ //Funcion para setear las configuraciones iniciales de la Wio
    -E5
  delay(500); //Tiempo de guarda
  Serial.println("Iniciando");
  enviarMensaje("AT"); //Se comprueba que este lista la comunicacion serial
  enviarMensaje("AT+ID"); //Permite verificar las credenciales de acceso del
     nodo
  enviarMensaje("AT+DR=US915"); //Se establece la banda de frecuencias US915
  enviarMensaje("AT+DR=DR0"); //Se establce SF de 12 y AB de 125kbps
  enviarMensaje("AT+CH=NUM,0-15"); //Se habilitan los primeros 15 canales de
     comunicacion
  enviarMensaje("AT+CLASS=A"); //Se setea al nodo como calse A
  enviarMensaje("AT+MODE=LWOTAA"); //Se configura el metodo de activacion
    como OTAA
```

```
11  }
```

Código 2.3: Función de inicialización del nodo

2.4.3.3 Función joinE5

Esta función inicia la conexión entre el gateway y el nodo LoRa a través del comando *AT+JOIN*, además cuenta con un tiempo de guarda que permite realizar la conexión y recibir el acuse de acceso, lo muestra por pantalla y borra el contenido del string leído.

```
1  void joinE5(){ //Funcion para iniciar la comunicacion con el GW
2    Serial.write("Tx:AT+JOIN"); //Se muestra en el monitor serial ed arduino
       la accion a realizar
3    e5.println("AT+JOIN"); //Se inicia la comunicacion con el GW
4    delay(20000); //Tiempo de guarda
5    while(e5.available()){ //compone un string char a char
6      char inChar = (char)e5.read();
7      inputString += inChar;
8    }
9    Serial.print(inputString); //MUestra el string en el monitor serial
10   inputString = ""; //Limpia el string
11   }
```

Código 2.4: Función de conexión con el gateway

2.4.3.4 Setup

En esta función se setean las velocidades para iniciar la comunicación serial tanto PC-Arduino como Arduino-Wio-E5 en 9600 baudios, seguido se llama a las funciones ***initE5*** y ***joinE5*** para aprovechar el subproceso que estas realizan con miras a mantener la comunicación entre el nodo sensor y el gateway.

```
1  void setup() { //Funcion de inicializacion de arduino
2    Serial.begin(9600); //Se inicia la comunicacion serial a 9600 baudios pc-
       arduino
3    e5.begin(9600); //Se inicia la comunicacion serial a 9600 baudios arduino-
       wio
4    initE5(); //Se realiza las configuraciones de inicio para la Wio-E5
```

```
5   joinE5(); //Se inicia la comunicacion con el arduino
6 }
```

Código 2.5: Función setup de arduino

2.4.3.5 Función capRssi

Esta función captura el RSSI del nodo a través del siguiente subproceso, en principio se envía un mensaje piloto en búsqueda de obtener un acuse de recibo en el que se reciba varios datos, entre estos el RSSI, Seguido se realiza la captura del string de respuesta, y se procesa los datos al punto de separar el valor de RSSI del resto de la cadena, el mismo es guardado en la variable *newV1* de tipo String el proceso es documentado en varios mensajes mostrados en el monitor serie de Arduino. Esta función no cuenta con argumentos de entrada ni de salida, sin embargo, edita variables globales que se usan en otras funciones.

```
1  void capRssi(){ //Definicion de funcion sin retorno
2    inputString=""; //Se limpia el srting
3    Serial.println("\nTx:AT+MSG"); //Se muestra un mensaje en el monitor
       serial de arduino
4    e5.println("AT+MSG"); //Se env a el mensaje para acuse de recibo
5    delay(17000); //Tiempo de espera promedio en recibir acuse de recibo
6    while(e5.available()){ //Funcion para leer el buffer del comunicador
       arduino-wio
7      char inChar = (char)e5.read(); //Lectura char a char
8      if(inChar=='X'){
9        inputString=""; //Se limpia el string tras una X ya que se conoce el
       mensaje esperado
10       }
11     inputString += inChar; //Se compone el string char a char
12   }
13   Serial.println(inputString); //se muestra por pantalla el string compuesto
14   int pos1=inputString.indexOf("RSSI"); //Se busca la plabara clave RSSI
       para realizar la captura
15   if (pos1>=0) { //Funci n para capturar el RSSI enviado en el acuse de
       recibo.
16    Serial.print("se ha detectado la palabra RSSI en la posicion ");
17    Serial.println(pos1);
```

```
18    newV1=inputString.substring(pos1+4, inputString.length()); //Se separa el
        RSSI del resto del string
19    newV1=newV1.substring(0, newV1.indexOf(",")); //Se captura el RSSI
       enviado en el acuse de recibo.
20    Serial.println(newV1); //Se muestra por pantalla el valor
21   }
22 }
```

Código 2.6: Función de obtención del RSSI

2.4.3.6 Función stringToHex

Como lo dice su nombre, esta función se encarga de convertir las cadenas de texto, que ingresen en su único argumento de entrada, de texto plano a cadenas hexadecimales, es decir, convierte cada carácter del string a un valor hexadecimal. Además, muestra tanto el texto original como la cadena hexadecimal generada a través del monitor serie de Arduino. Finalmente, se encarga de realizar el envío del mensaje final convertido haciendo uso del comando *AT+MSGHEX*. Esta función no cuenta con argumentos de entrada ni de salida, sin embargo, edita variables globales que se usan en otras funciones.

```
1 void stringToHex(String s1){ //Funcion para transformar un string en
     hexadecimal
2 int hex_dec; //Variable local auxiliar
3 Serial.println();
4 Serial.print("string: ");
5 Serial.println(s1); //Se muestra el string inicial
6 Serial.println("hexval: ");
7 for (const auto &item : s1) { //Se convierte cada char del string a su
     equivalente hexadecimal y junta todo en una sola cadena
8   hex_dec = int(item);
9   char hexaDeciNum[100];
10  int i = 0;
11  while (hex_dec != 0) { //convierte cada char del string a su equivalente
      hexadecimal
12    int temp = 0;
13    temp = hex_dec % 16;
14    if (temp < 10) {
15      hexaDeciNum[i] = temp + 48;
16      i++;
```

```
17      }
18      else {
19        hexaDeciNum[i] = temp + 55;
20        i++;
21      }
22      hex_dec = hex_dec / 16;
23    }
24    for (int j = i - 1; j >= 0; j--){ //Compone la cadena final hexadecimal
25      output += hexaDeciNum[j];
26    }
27  }
28  output="AT+MSGHEX="+output; //Prepara el mensaje a enviarse al GW
29  Serial.println(output); // muestra por el monitor serial de arduino el
        comando a utilizar
30  e5.println(output); //envia el mensaje al GW
31  }
```

Código 2.7: Función de conversión de strings a hexadecimal

2.4.3.7 Función msjGw

Esta función está compuesta por 4 etapas, primero llama a la función ***capRssi*** para obtener el RSSI del nodo, y comienza a dar formato a la información que se mostrará en pantalla, seguido compone un srting que contenga tanto la etiqueta como el valor de RSSI en texto formato JSON, como tercer punto llama a la función ***stringToHex*** ingresando la cadena de caracteres, elaborada en el punto anterior, como argumento de entrada, por último limpia todas las variables globales dejando el programa listo para la siguiente iteración. Esta función no cuenta con argumentos de entrada ni de salida, sin embargo, edita variables globales que se usan en otras funciones.

```
1  void msjGW(){ //Funcion para enviar mensaje al GW
2    capRssi(); //se captura el RSSI
3    Serial.println("");
4    Serial.println("--------------------");
5    Serial.println("Resultado");
6    String res="{\"rssi\":"+newV1+"}"; //Se da formato al mensaje que se
       quiere enviar
7    Serial.println(res); //Se muestra el mensae que se enviara
8    Serial.println("--------------------");
```

```
9   Serial.println("");
10  Serial.println("");
11  Serial.println("--------------------");
12  Serial.println("Resultado");
13  stringToHex(res); //Se transforma el mensaje a hexadecimal
14  Serial.println("--------------------");
15  Serial.println("");
16  delay(5000); //tiempo de guarda
17  //----------------- Se limpian todas las variables ---------------------
18  output="";
19  res="";
20  while(e5.available()){
21    e5.read();          //Funcion para limpiar el buffer
22  }
23  inputString = "";
24  //----------------------------------------------------------------------
25  }
```

Código 2.8: Función de envío de mensaje final al gateway

2.4.3.8 Loop Principal

En el loop principal de Arduino se cumplen un único proceso, en el que a través de contadores activados por la función *millis()* de Arduino, se configura un periodo de tiempo establecido en el que el programa llama a la función ***msjGw*** una vez se haya cumplido dicho periodo.

```
1 void loop() { //Funcion principal del programa
2    unsigned long nowT = millis(); //Se actualiza el tiempo actual
3   if (nowT - prevT >= intervalo) { //Se comprueba si se ha cumplido el
      intervalo
4     prevT = nowT; //Actualiza el tiempo previo
5     msjGW(); //Envia un mensaje al GW
6   }
7 }
```

Código 2.9: Loop principal del programa

Para el correcto funcionamiento del código se establecen algunas variables globales, además de que se hace uso de la librería ***SoftwareSerial.h*** y se inicializa un objeto Software-

Serial en los pines D6 y D7 del Arduino Nano, mismo que servirá como comunicador serial entre la placa Arduino y la placa Wio.

Al adicionar los últimos componentes mencionados anteriormente, el código general se muestra en el Anexo 3.

2.4.3.9 Resultados de la ejecución en tiempo real (Monitor Serial)

Tras iniciar con la ejecución del código se obtiene mucha información, esto debido a que se han ocupado varios prints en el monitor serial para indicar el proceso que se encuentra ejecutando, logrando dividir el monitoreo por puerto serie en dos etapas.

1. **Inicialización y conexión a GW:** En esta primera etapa se muestran los comandos utilizados para configurar la placa Wio-E5 y las respuestas de la misma enviadas por el puerto serial configurado en los pines D7 y D6 del Arduino. De esta manera se presentan los resultados obtenidos desde el monitor serial del IDE en la figura 2.24.

2. **Estructuración y envío de mensajes:** Una vez que la conexión se ha establecido, comienza la obtención de datos desde el sensor, y se procesan, estos se ponen en formato de texto plano tipo JSON, los mismos siguen una estructura de diccionario donde el programa asigna una etiqueta y un valor al mensaje que se enviará, seguido este se transforma en cadenas hexadecimales para finalmente enviar el mensaje al gateway.

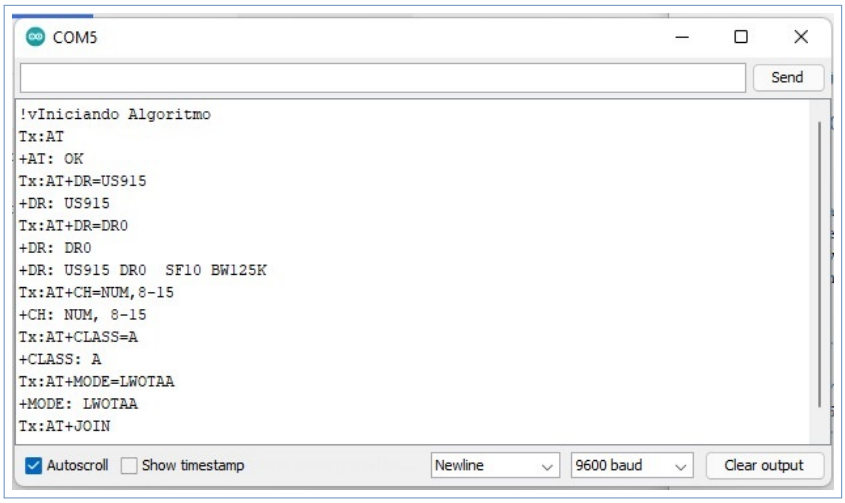

Figura 2.24: Configuración e inicio de conexión monitorizado desde el puerto serial

2.4.3.10 Ejemplo de la ejecución en tiempo real (Live Data Consola de TTN)

Al igual que en el monitor serial de Arduino la consola de TTN brinda una herramienta que permite ver toda la data intercambiada entre el nodo y el gateway desde el momento que se requiere la conexión, como se muestra en la figura 2.25, se puede diferenciar los datos enviados para requerir la conexión, para recibir los datos y los datos decodificados gracias al Uplink formatter creado anteriormente.

Figura 2.25: Presentación de data obtenida por la consola de TTN

2.4.4 Dashboard en Node-Red

Para presentar la información obtenida se ha decidido crear un dashboard haciendo uso de una máquina virtual con Ubuntu, en la misma se ha instalado Node.js y el package de Node-Red, para iniciar la ejecución de Node-Red se debe abrir un terminal en la máquina virtual y ejecutar el comando ***Node-Red.***

Una vez se ha iniciado la ejecución de Node Red se debe abrir un navegador y para acceder al IDE editor de flujos se debe ingresar a la dirección local en el puerto 1880, una forma de hacerlo es con el URL *127.0.0.1:1880*, de esta manera se iniciará un nuevo flujo.

Antes de iniciar con la edición del flujo se debe instalar la paleta ***Node-red-dashboard***, para esto se debe ingresar a opciones en la barra superior, el botón se encuentra en la parte superior derecha a un lado del botón de *Deploy*, en el menú desplegable se selecciona *Manage Palette*.

En la ventana emergente se da clic sobre la pestaña Install y en el buscador se escribe *Node-red-dashboard*, en la primera opción que se lista, se da clic en el botón install, esto debido a que la primera opción será la más actualizada.

Con todas las paletas listas Se inicia el desarrollo del flujo, el mismo se puede separar en tres etapas, conexión, filtrado de datos y presentación de datos, así como se muestra en la figura 2.26.

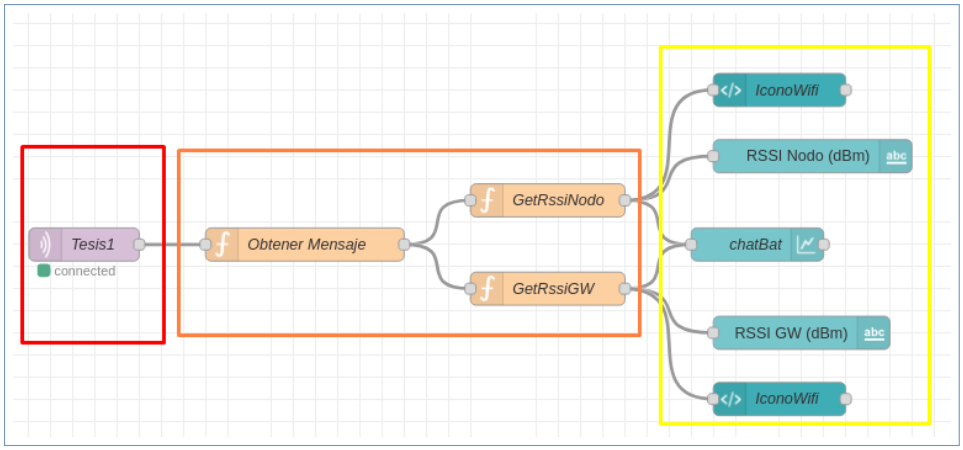

Figura 2.26: Flujo de creación del Dashboard

2.4.4.1 Etapa de conexión

Esta etapa consta de un único nodo del tipo *Mqtt in* en este se configura la conexión al servidor, para esto se especifica el servidor ***nam1.cloud.thethings.network***, en el puerto ***1883***, en la pestaña de seguridad se ingresa el usuario y la API Key definidas para la integración Mqtt configurada previamente.

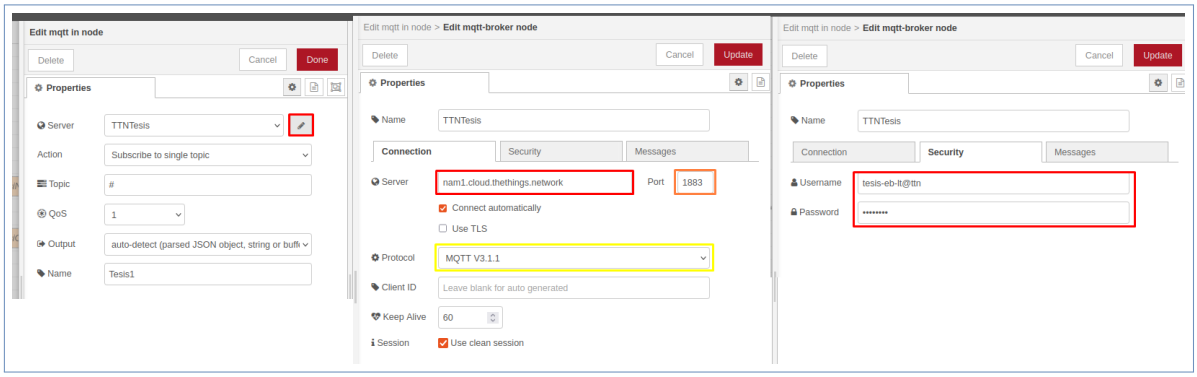

Figura 2.27: Configuración de la entrada de datos por MQTT

2.4.4.2 Etapa de filtrado de datos

Para la etapa de filtrado de datos se utiliza tres funciones escritas en JavaScript, cada una permite la obtención del mensaje, y de los datos específicos respectivamente, estas son:

1. **Obtener Mensaje:** Esta función permite separar los mensajes que contienen datos codificados de los que solo son mensajes de control.

```
if (msg.payload.uplink_message.decoded_payload) {
    return msg;
}
```

Código 2.10: Función Obtener Mensaje

2. **GetRssiNodo:** Esta función permite obtener el dato llamado RSSI del mensaje codificado enviado desde el gateway.

```
msg.payload = msg.payload.uplink_message.decoded_payload.Values.rssi;
msg.topic = "RSSIN";

return msg;
```

Código 2.11: Función getRssiNodo

3. **GetRssiGW:** Esta función permite acceder a los metadatos enviados desde el gateway y capturar el RSSI en el gateway.

```
msg.payload = msg.payload.uplink_message.rx_metadata[0].rssi;
msg.topic = "RSSIG";

return msg;
```

Código 2.12: Función getRssiGW

Cada una de las funciones tiene como entrada un mensaje en formato JSON enviado desde el gateway y retorna una salida específica luego de haberse aplicado el filtro.

2.4.4.3 Etapa de presentación de datos

A su vez esta etapa se divide en dos secciones clasificadas por su forma de mostrar los datos, en la primera se utiliza texto y diagramas predefinidos en la paleta de Dashboard, en la segunda sección se utiliza nodos template con el que se puede hacer uso de Angular.JS y HTML para mostrar los datos a través de un icono representativo de RSSI que cambia en función del porcentaje de nivel de energía recibido.

Conociendo los nodos a usar se inicia con la creación del dashboard, para esto se dirige a la barra lateral derecha, en el botón de más opciones se selecciona dashboard.

Con las opciones de dashboard abiertas en la pestaña layout se crea un nuevo Tab, el mismo requiere de un nombre para ser creado.

Con el tab creado se procede a crear un nuevo grupo, colocando el cursor sobre el nombre del tab y seleccionando la opción *+group*.

En la nueva ventana desplegada se debe ingresar el nombre del grupo y el tab al que este pertenecerá, opcional se puede editar el ancho predeterminado del grupo en la opción Width, este mismo proceso se sigue por cada grupo de widgets que se quiera mostrar en el dashboard.

Para esta implementación se han definido 3 grupos en los que se mostrará la información, los mismos son Función del tiempo, Calidad de la señal Nodo y Calidad de la señal GW.

- **Grupo Función del tiempo:** En este grupo se utiliza un nodo de tipo chart, en el mismo se deben configurar el grupo al que pertenece el nodo, el tamaño si se desea fijo o automático, se puede asignar una etiqueta al chart generado, se define el tipo de chart, se ajusta los ejes x (tiempo) e y (valores mínimos y máximos de potencia reconocida), por último se le asigna un nombre al nodo. Adicional a esto se pueden generar leyendas, se puede editar los colores de las series representadas y se puede agregar código adicional en CSS para modificar los estilos del chart.

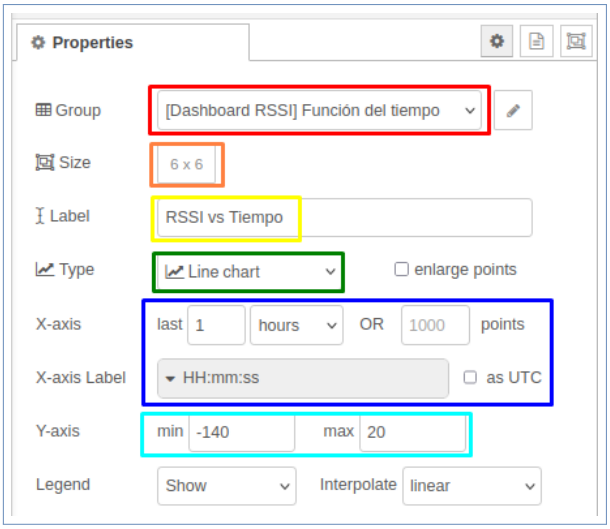

Figura 2.28: Configuración de nodo chartRssi

- **Grupo Calidad de Señal Nodo:** En este grupo se agregan dos nodos, el primero es

un nodo de tipo texto, en el que se configuran el grupo al que se pertenece, se da un nombre, se configura un tamaño y el formato del valor a recibirse.

Para el segundo se utiliza un nodo tipo template en el que se hace uso de condicionales ***ng-if*** para asignar un estilo distinto al icono representativo de Wi-Fi [23][24], usado en este caso para representar el RSSI dividido en 4 partes porcentuales que representan la cantidad de potencia recibida tanto en el nodo como en el gateway. Los niveles y nombre de los iconos usados se pueden observar en el código 2.13.

```
<div class="body" fxLayoutAlign="center center">
    <mat-card>
        <div ng-if="msg.payload > -40">
            <ng-md-icon icon="signal_wifi_4_bar" ng-attr-style="fill: #abcdef" size="200"></ng-md-icon>
        </div>
        <div ng-if="msg.payload > -80 && msg.payload <= -40">
            <ng-md-icon icon="signal_wifi_3_bar" ng-attr-style="fill: #abcdef" size="200"></ng-md-icon>
        </div>
        <div ng-if="msg.payload > -110 && msg.payload <= -80">
            <ng-md-icon icon="signal_wifi_2_bar" ng-attr-style="fill: #abcdef" size="200"></ng-md-icon>
        </div>
        <div ng-if="msg.payload > -130 && msg.payload <= -110">
            <ng-md-icon icon="signal_wifi_1_bar" ng-attr-style="fill: #abcdef" size="200"></ng-md-icon>
        </div>
        <div ng-if="msg.payload <= -130">
            <ng-md-icon icon="signal_wifi_0_bar" ng-attr-style="fill: #abcdef" size="200"></ng-md-icon>
            <script type="text/javascript">
                alert("Nivel bajo de RSSI por favor no se aleje más del GW")
            </script>
        </div>
    </mat-card>
</div>
```

Código 2.13: Generación del icono representativo de RSSI y alertas mediante condicionales

❒ **Grupo Calidad de la señal GW:** utiliza la misma configuración del grupo anterior con la diferencia que tomas los datos desde la función GetRssiGW en lugar de la función GetRssiNodo.

2.4.4.4 Despliegue del flujo creado

Como resultado esperado se genera un dashboard con los tres grupos definidos anteriormente, el primero de ellos muestra un diagrama histórico del comportamiento del RSSI en la última hora, y los otros dos grupos representan el valor actual del RSSI tanto del nodo como del gateway, de manera gráfica y en texto.

Para acceder al dashboard desarrollado se debe ingresar en un navegador web y en la barra de URL digitar ***127.0.0.1:1880/ui,*** de esta manera se obtiene la ventana mostrada en la figura 2.29.

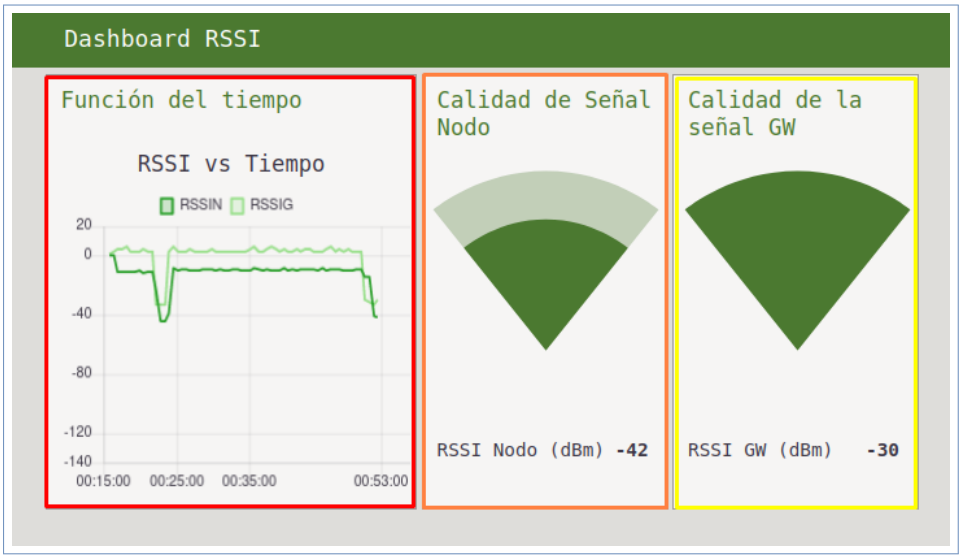

Figura 2.29: Dashboard desarrollado

2.4.5 Diseño electrónico del prototipo

Una vez se ha conseguido un prototipo funcional desde software, se requiere realizar la implementación física del nodo sensor, para esto, se ha decidido usar una baquelita perforada, con miras a futuras implementaciones sugeridas en 3.2 Trabajos Futuros, además se usarán zócalos ubicados específicamente para adaptarse a la forma de las dos tarjetas usadas, un interruptor, una entrada de batería de dos pines, una porta pilas AA, dos borneras de dos pines y segmentos de cable unifilar.

En la figura 2.30 se presenta el diagrama de conexión o esquemático del prototipo a implementarse, en el mismo se establece la conexión entre las borneras y los pines A1 y D9 del Arduino, dando la posibilidad de agregar un sensor analógico y un sensor digital externos,

además se indica la conexión entre el puerto serial del Arduino con el de la Wio-E5, por último se muestra que tanto el Arduino como la Wio-E5 son alimentadas desde la misma batería.

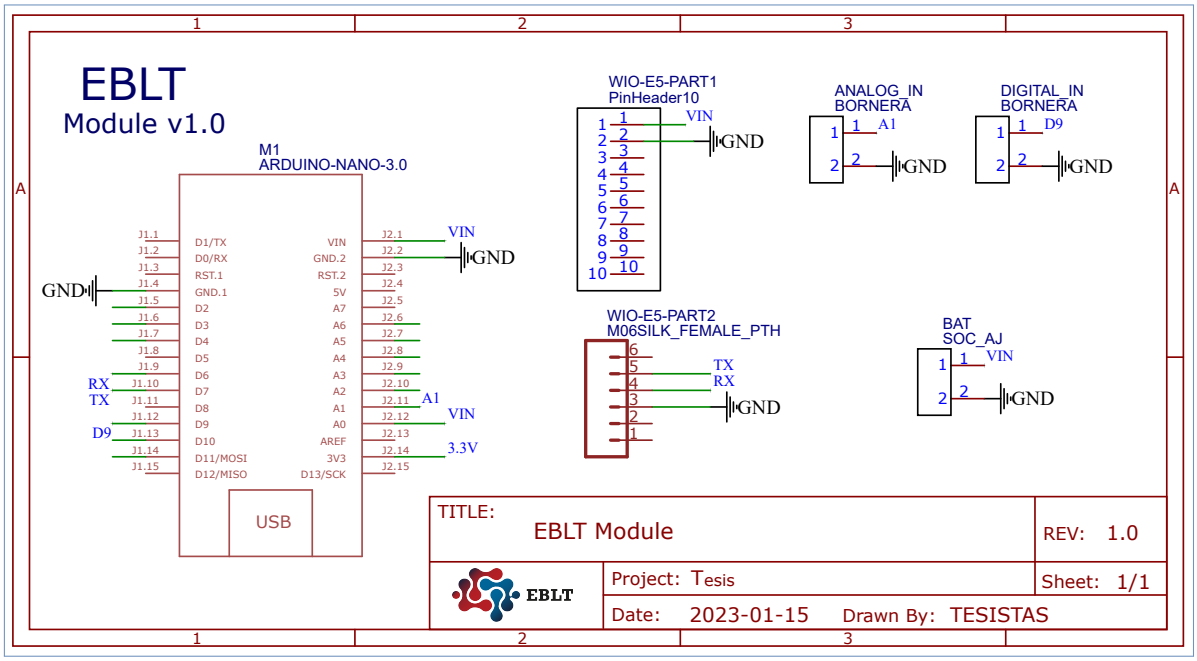

Figura 2.30: Esquemático del prototipo

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En esta sección se presentará la implementación física del prototipo, y los resultados obtenidos en diferentes escenarios de funcionamiento en los que se variará la altura del gateway y la distancia a la que se encuentra el nodo, estos serán contrastados con simulaciones en Radio Mobile y después serán presentados tanto a nivel de aplicación y de monitor serial.

3.1 RESULTADOS

3.1.1 Implementación del prototipo

Como se muestra en la figura 2.30 se ha diseñado un circuito electrónico que permita conectar una placa Arduino Nano con la Wio-E5, manteniendo la posibilidad de agregar nuevos módulos sensores, ya que se ha implementado el diseño en una baquelita perforada.

Además, se procura usar la menor cantidad de espacio posible en el prototipo, para que el mismo entre en una carcasa diseñada e impresa en 3D, a continuación se presenta el resultado final de la implementación.

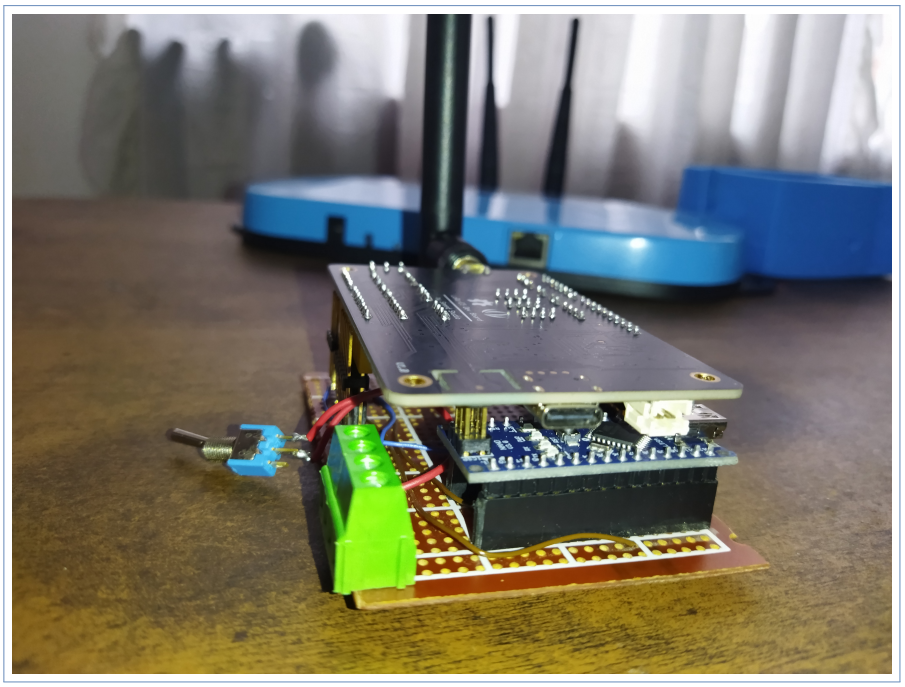

Figura 3.1: Prototipo implementado

A manera de protección antes los insectos y el polvo, se ha decidido crear una carcasa que se ajusta al prototipo implementado, dejando expuestos solo los puertos para un sensor digital y un sensor analógico, para esto se ha diseñado una estructura 3D con la ayuda de la herramienta Tinkercad, la misma es una carcasa de 99 mm de largo, 73mm de ancho y 53 mm de altura una vez armada.

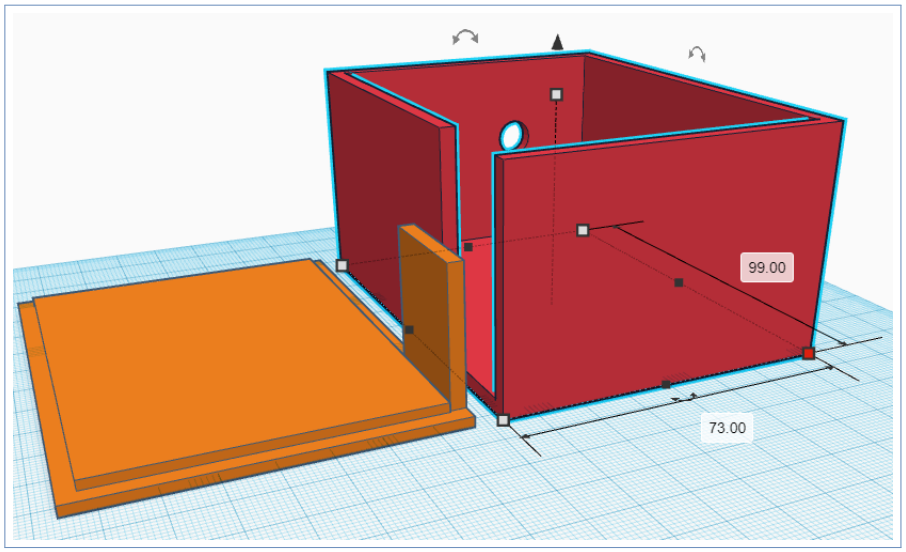

Figura 3.2: Estructura diseñada

Una vez impresa la estructura se coloca al prototipo dentro de la carcasa y se cierra dejando el interruptor expuesto para poder encender fácilmente el nodo sensor, como se muestra en la figura 3.3.

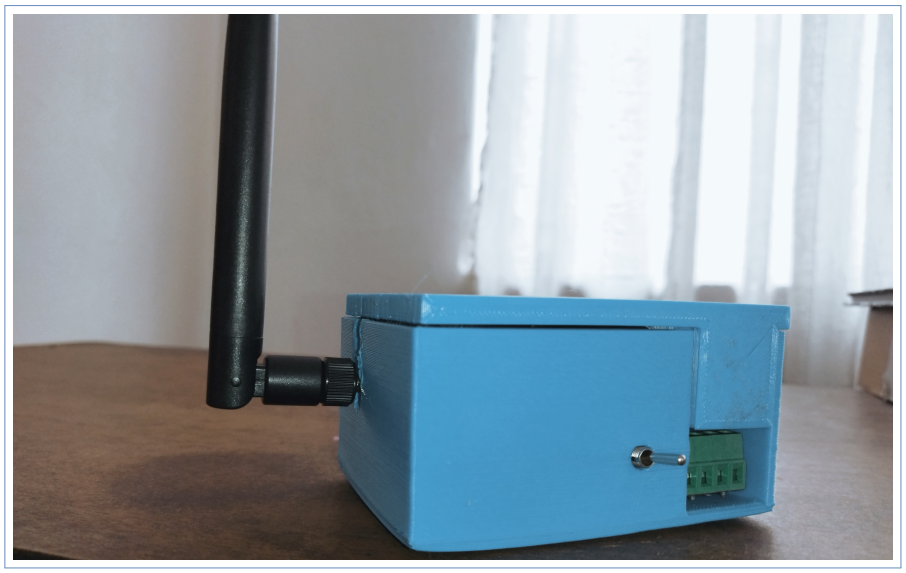

Figura 3.3: Prototipo terminado

3.1.2 Funcionamiento del sensor y monitoreo por el puerto serial

Como se explicó en las funciones presentadas anteriormente, la placa Arduino Nano permite monitorear el funcionamiento del nodo a través del puerto serial, de esta manera se presenta en pantalla cada uno de los comandos utilizados, las mediciones que realiza el nodo sensor, el mensaje que se quiere enviar, y el mensaje en código hexadecimal para su posterior conversión a formato JSON, como se muestra en la figura 3.4.

```
Iniciando
Tx:AT
+AT: OK
Tx:AT+DR=US915
+DR: US915
Tx:AT+DR=DR0
+DR: DR0
+DR: US915 DR0  SF10 BW125K
Tx:AT+CH=NUM,0-15
+CH: NUM, 0-15
Tx:AT+CLASS=A
+CLASS: A
Tx:AT+MODE=LWOTAA
+MODE: LWOTAA
Tx:AT+JOIN+JOIN: Start
+JOIN: NORMAL
+JOIN: Network joined
Tx:AT+VDD

se ha detectado la palabra VDD en la posicion 16
 3.26
Tx:AT+MSG
XWIN1, RSSI -8, SNR 13.0

se ha detectado la palabra RSSI en la posicion 7
 -8
--------------------
Resultado
{"bat": 3.26,"rssi": -8,"vcc":4.32}
--------------------
--------------------
Resultado

string: {"bat": 3.26,"rssi": -8,"vcc":4.32}
hexval:
AT+MSGHEX=7B22626174223A20332E32362C2272737369223A202D382C22766363223A342E33327D
```

Figura 3.4: Monitoreo por el puerto serial del nodo en su inicialización y la primera iteración

Como se puede observar en la figura 3.4 se sigue al pie de la letra el algoritmo presentado en el diagrama de flujo de la figura 2.23, de esta forma se obtiene los datos a partir de mensajes que contienen códigos AT y son enviados a través de un puerto serial virtual creado en el Arduino Nano conectado al puerto UART1 de la Wio-E5, una vez obtenidos los datos

se compone un mensaje en formato JSON, seguido este mensaje se transforma el mensaje a código hexadecimal y se compone un nuevo mensaje uniendo el código obtenido a un comando AT para el envío de mensajes en hexadecimal, todo este proceso es documentado por el monitor serial de Arduino Nano.

3.1.3 Funcionamiento y precisión del nodo sensor

En esta sección se definirá un escenario de pruebas, seguido se presentará los resultados obtenidos y se realizará un análisis de contraste entre los datos reales y los datos simulados en Radio Mobile, por último se presentará un análisis del error porcentual para determinar la precisión del nodo sensor de nivel de energía.

3.1.3.1 Escenario de prueba

Para dar inicio a las pruebas se ha configurado la potencia de transmisión del gateway y del nodo sensor en 20 dBm además se ha ubicado el gateway a lado de una ventana en el cuarto piso de la facultad de sistemas de la Escuela Politécnica Nacional, a una altura aproximada de 30 m.

Además, se conoce que las antenas del gateway y del nodo tienen una ganancia de 3 dBi y que al probar la red en la ciudad de Quito se puede considerar que se realiza en un entorno totalmente urbano.

La banda de frecuencias definida para realizar las pruebas es la US915 con frecuencia central en los 915MHz, además se ha configurado el nodo para funcionar como clase A y con un factor de Spreading de 10.

3.1.3.2 Obtención de valores esperados

Para obtener datos cercanos a la realidad sobre los valores esperados para el nivel de intensidad de señal se ha utilizado la herramienta Radio Mobile para simular un enlace LoRaWAN entre un nodo terminal y un gateway LoRa modificando la configuración presentada en [25], se ha seleccionado las ubicaciones del gateway en la facultad de Sistemas de la EPN y de distintas posiciones del nodo haciendo uso de Google Maps, las mismas se

presentan en el Anexo 6 en la sección de vRadioM.

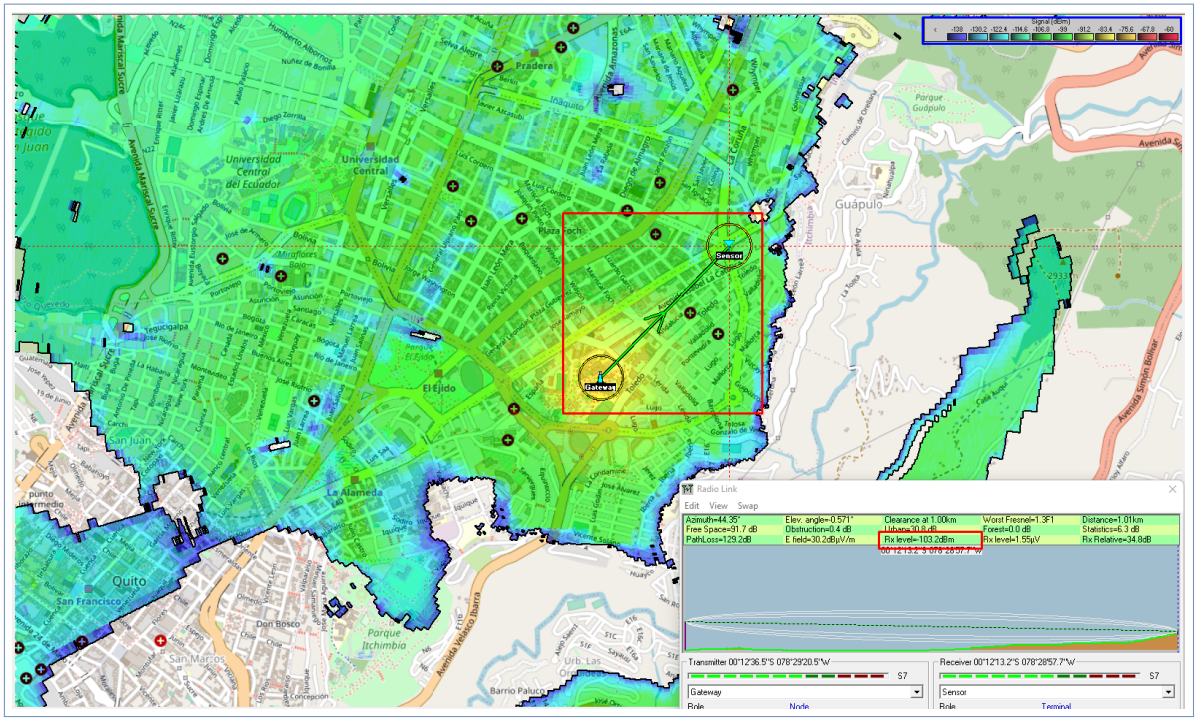

Figura 3.5: Simulación del enlace LoRaWAN en Radio Mobile

En la figura 3.5 se muestra el enlace simulado, con las posiciones tanto del gateway como del nodo, los niveles de energía de la señal representados por colores y las características del enlace, entre estas se destaca ***RX Level***, que es la medida del RSSI.

3.1.3.3 Resultados obtenidos

Los resultados fueron obtenidos tomando las siguientes mediciones en tiempo real, RSSI del Nodo, RSSI del gateway, Latitud de la posición del nodo, Longitud de la posición del nodo y distancia entre el Nodo y el gateway esta última se obtuvo mediante el uso de una función para el cálculo de la distancia entre dos puntos.

Los datos obtenidos están disponibles en el Anexo 6, además, se hizo uso de la herramienta MATLAB para graficar los mismos en función de la distancia, como se muestra en la figura 3.6. Se hace referencia a que estos datos se muestran en función de la distancia debido a que en la aplicación desarrollada se presentan datos similares, sin embargo, estos son graficados en función del tiempo, por lo que los cuadros resultantes no serían comparables.

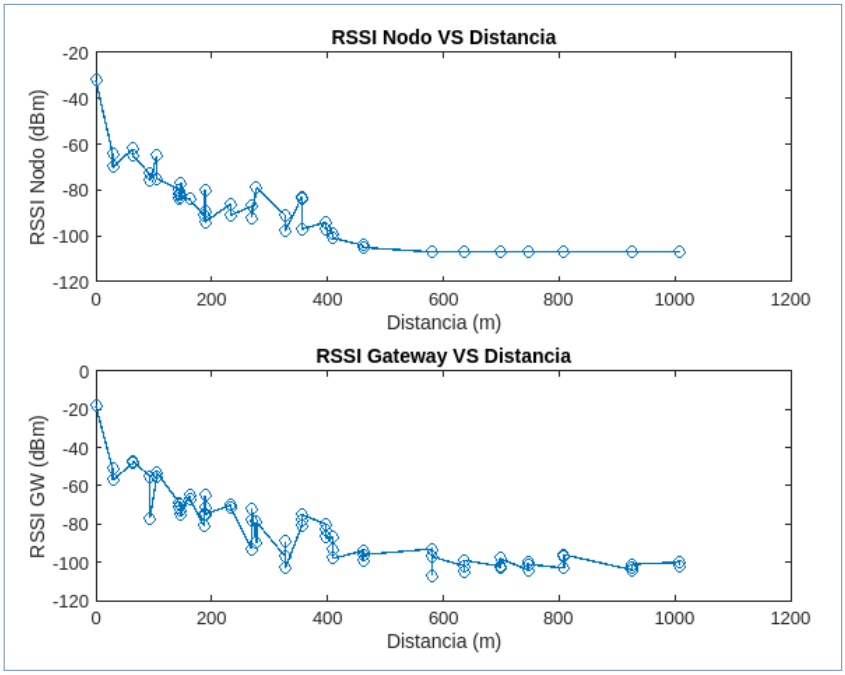

Figura 3.6: Valores obtenidos tras las pruebas

Tras la obtención de resultados, se propone usar los datos obtenidos tras varias simulaciones en Radio Mobile para contrastar los valores reales obtenidos con los valores esperados.

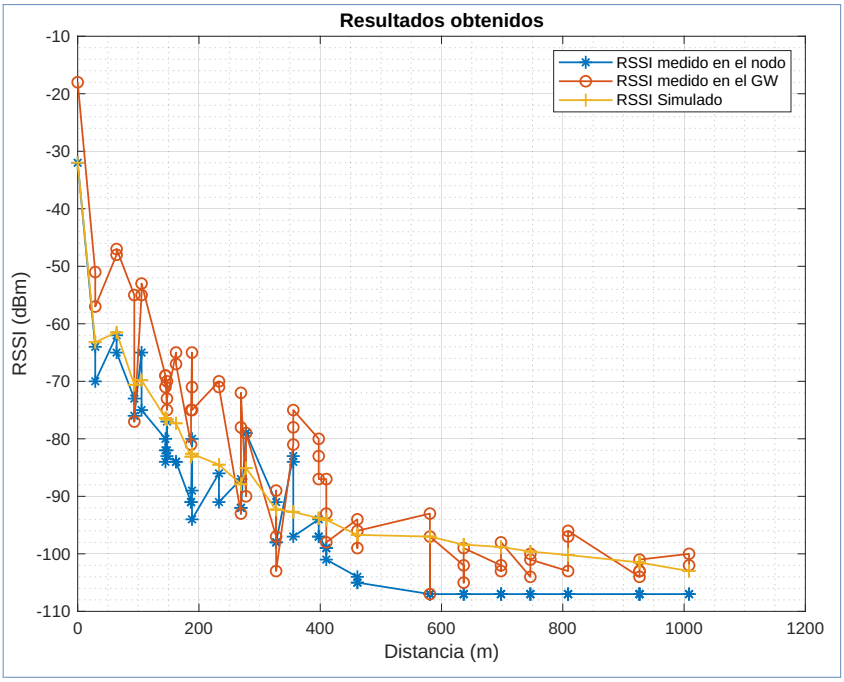

Figura 3.7: Comparación entre valores obtenidos y valores esperados

Como se puede observar en la figura 3.7 los valores tanto del RSSI del Nodo como del Gateway son bastante cercanos a los valores obtenidos en la simulación, por lo que se

espera tener un error porcentual bastante bajo.

3.1.3.4 Error porcentual y error porcentual medio

Para el cálculo del error porcentual y el error porcentual medio se ha hecho uso del código de MATLAB presente en el Anexo 7, en el mismo se calcula valor a valor para después haciendo uso de la función ***mean()*** obtener el promedio de los datos como se muestra en las figuras 3.8 y 3.8.

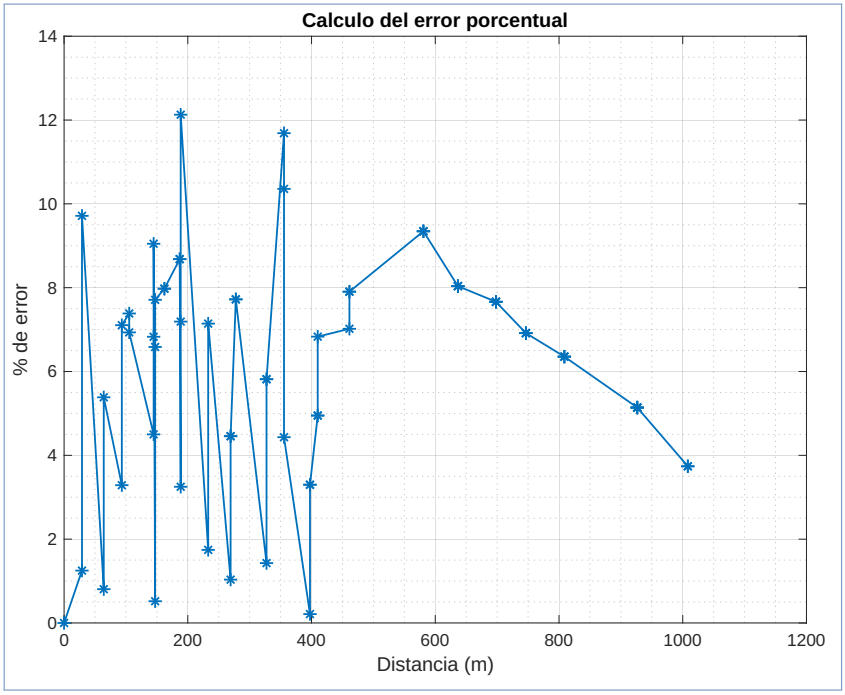

Figura 3.8: Error obtenido valor a valor

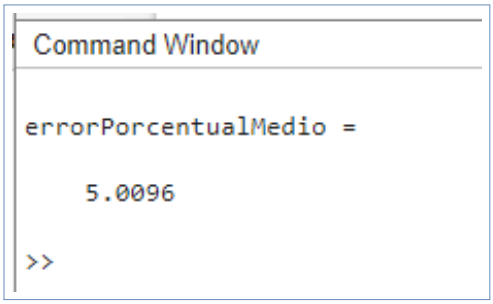

Figura 3.9: Error Porcentual Medio

Tras realizar un análisis de los errores porcentuales obtenidos se puede observar que el sensor tiene un muy buen performance, ya que individualmente los errores obtenidos en su mayoría son menores al $10\,\%$, además, al calcular el valor del error porcentual medio

se ha obtenido un valor aproximado al $5\,\%$ por lo que se puede asegurar que el valor leído por el sensor del nodo es correcto, ya que en la simulación no se ha podido considerar por completo las edificaciones que existen en la ciudad. Una vez se ha logrado determinar que las lecturas realizadas por el sensor del nodo son correctas, se procede a comprobar que las funcionalidades de monitoreo agregadas a la aplicación desarrollada funcionan correctamente.

3.1.4 Funcionamiento de la aplicación de monitoreo

Se han agregado tres funcionalidades que buscan facilitar el monitoreo e interpretación de los datos presentados en el dashboard. Estas funcionalidades son: presentación de un histórico de los valores de RSSI tanto del nodo como del gateway, icono dinámico del RSSI tanto del nodo como del gateway y sistema de alerta para valores de RSSI menores a 130 o por desconexión.

Las dos primeras funcionalidades se muestran todo el tiempo en el dashboard, como se muestra en la figura 3.10, y el sistema de alerta se muestra únicamente cuando se detecta valores demasiados bajos de RSSI o por desconexión.

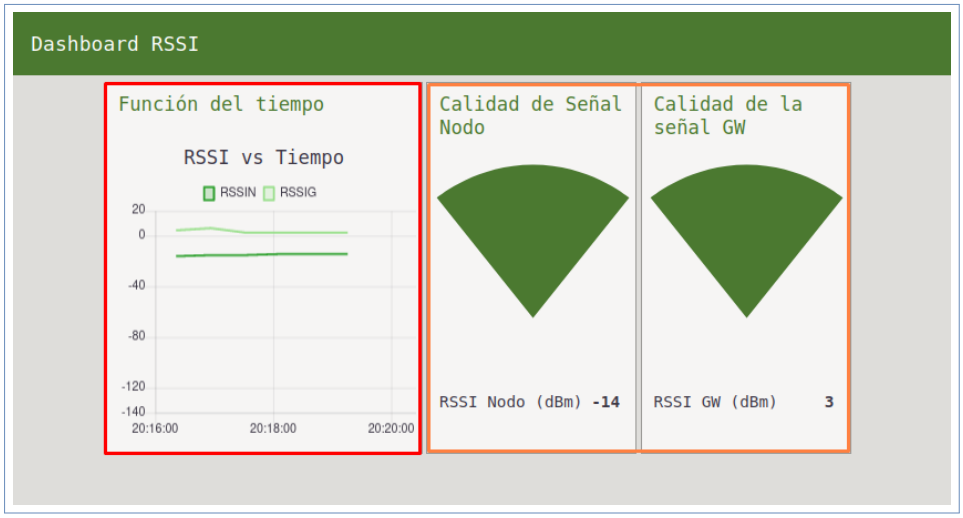

Figura 3.10: Funcionalidades permanentes desarrolladas

3.1.4.1 Histórico de valores de RSSI

En este cuadro se pueden observar todos los valores obtenidos durante la última hora de funcionamiento del nodo, además se grafica el RSSI en dBm vs el tiempo transcurrido. Al

ser un cuadro dinámico se actualiza automáticamente cada vez que llega un dato y permite distinguir entre los dos RSSI medidos a través de leyendas con colores distintos.

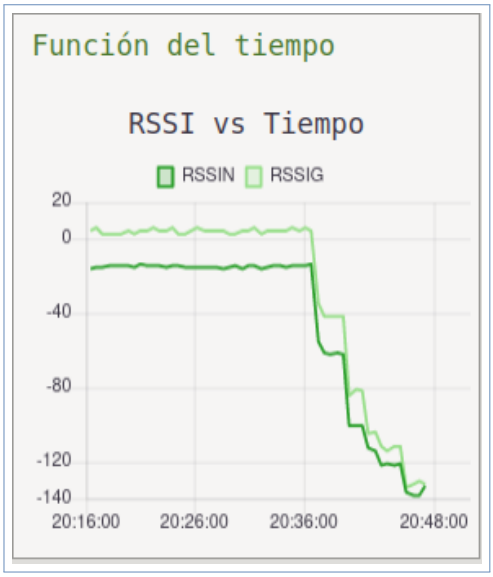

Figura 3.11: Histórico del RSSI tras la realización de pruebas

3.1.5 Iconos dinámicos del RSSI

Este cuadro permite un monitoreo porcentual de los niveles de RSSI obtenidos, se permite mostrar 5 niveles como se muestra en las siguientes figuras.

Primero se presentan los niveles más altos de señal, estos niveles se consideran en Indor, y para pequeñas distancias.

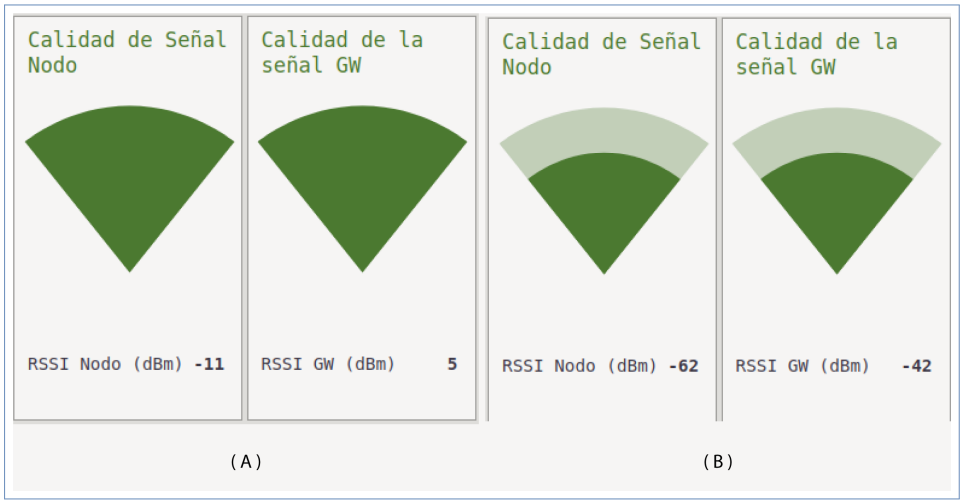

Figura 3.12: Niveles superiores al $80\,\%$ de intensidad de señal

Para los niveles intermedios se ha considerado las medianas y largas distancias, hasta -130 dBm, en donde se puede asegurar la transmisión de datos para todas las configuraciones de SF y para niveles de batería que permitan el funcionamiento tanto del Arduino Nano como de la WIO-E5, hasta 3.75V.

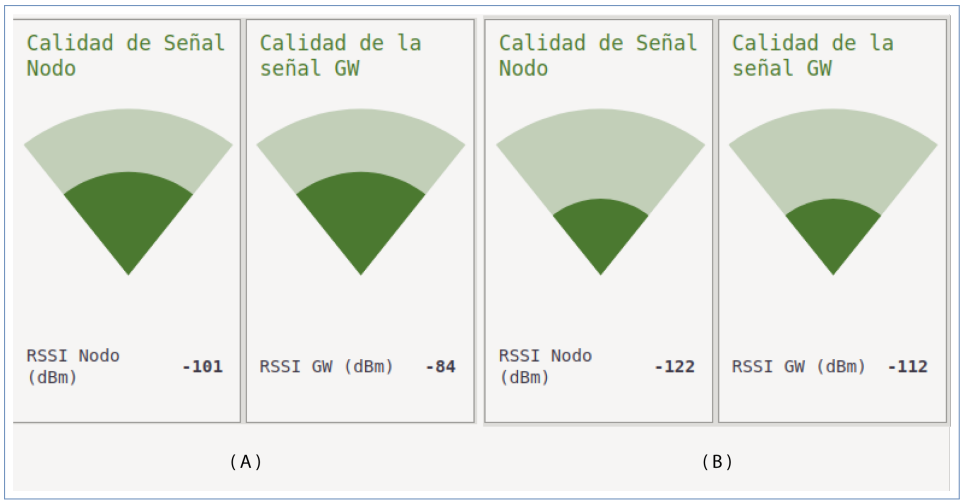

Figura 3.13: Niveles intermedios, entre el $10\,\%$ y el $80\,\%$ de intensidad de señal

Adicional se ha agregado un nivel, inferior a los -130 dBm, que muestra datos mientras siga existiendo recepción, sin embargo, este nivel se presenta en conjunto con la alerta mencionada anteriormente, para ayudar al monitoreo, en cuanto se pierda la conexión este nivel deja de actualizarse dando un punto de referencia temporal de la conexión del nodo.

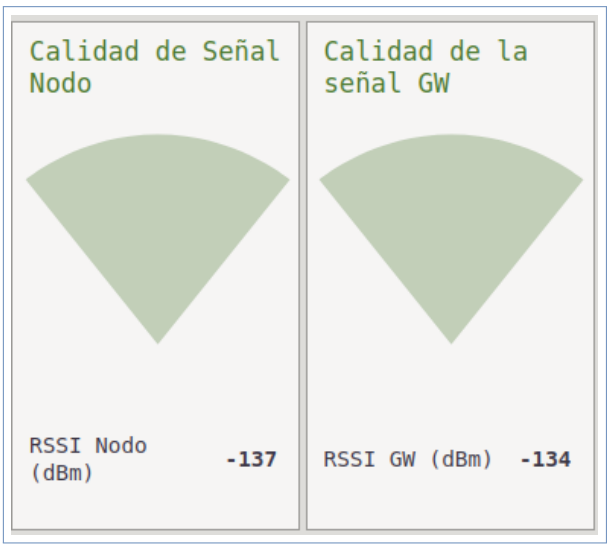

Figura 3.14: Nivel inferior al $10\,\%$ de intensidad de señal y marca temporal de desconexión

3.1.6 Sistema de alerta

Esta funcionalidad se dispara en cuanto las medidas de RSSI del nodo son inferiores a los 130 dBm como un sistema de alerta temprana para evitar la desconexión del nodo, sin embargo, se debe usar en conjunto con el icono dinámico de nivel inferior al $10\,\%$ de intensidad de la señal para detectar puntos de desconexión a través de la marca temporal.

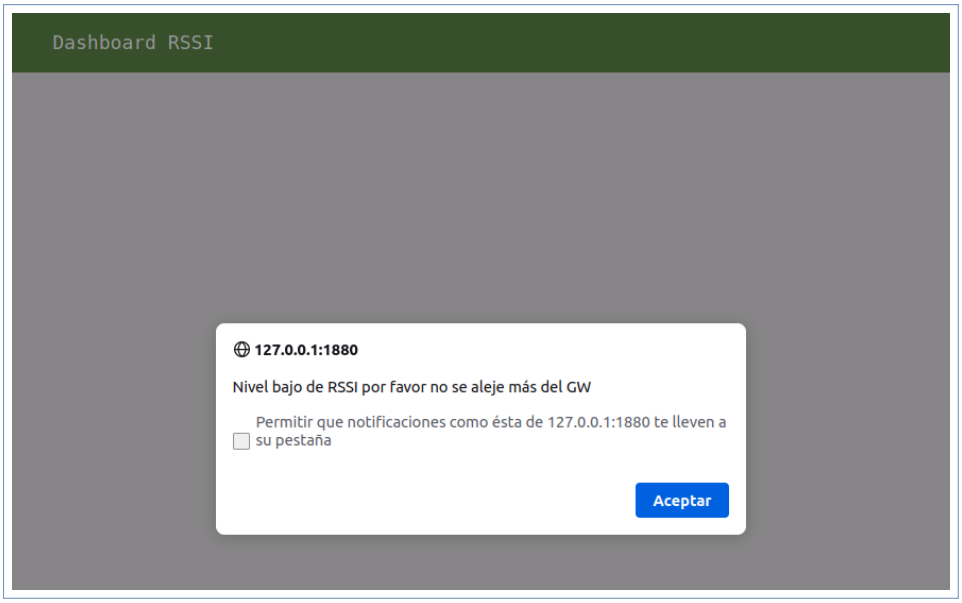

Figura 3.15: Ejemplo aleta disparada por niveles bajos de RSSI

3.2 TRABAJOS FUTUROS

En este apartado se proponen futuros trabajos que puedan utilizar tanto la tarjeta Wio-E5 como el nodo implementado en el presente trabajo de integración curricular.

- ❒ Sistema de sensores múltiples con un solo transiver LoRaWAN.
- ❒ Balanceo de carga para sistemas multisensores.
- ❒ Algoritmos de optimización de uso de la batería para nodos sensores.

3.3 CONCLUSIONES

- ❒ En el presente trabajo, se diseñó e implemento un sensor de nivel de señal para el sistema de monitoreo de red LoRaWAN en la ciudad de Quito. Adicional a esto se

desarrolló un dashboard en un servidor de red local para facilitar la visualización e interpretación de datos para el usuario, incorporando funcionalidades que permiten apreciar el cambio de datos receptados en tiempo real. De esta manera se logró determinar porcentajes de error aceptables y se consiguieron transmisiones superiores a 1 Km usando el prototipo desarrollado.

- Tras el proceso de investigación previo al diseño del nodo sensor, se logró determinar las distintas configuraciones y características de la red, tanto del nodo como de la simulación usada para la obtención de errores, de esta forma se consiguió mantener un enlace funcional con transmisiones continuas a largas distancias sin necesidad de cambiar ninguna de las configuraciones iniciales.

- La red diseñada cumple con todas las especificaciones y requerimientos comunes de una red LoRaWAN, en donde se ha diseñado un nodo sensor que cuenta con un módulo transiver LoRa, el mismo se conecta a un gateway LoRaWAN que registra los datos en un servidor de red, en este caso se usó la consola de TTN como servidor de red, además se aprovechó la integración MQTT de TTN para enviar los datos al dashboard desarrollado en Node-Red, siendo este la aplicación final que se muestra al usuario. Por esto se concluye que tanto el diseño de la red como la implementación de la misma es correcta, ya que se probaron todos y cada uno de sus requerimientos obteniéndose resultados satisfactorios.

- Tras la revisión de los diferentes modos de velocidad y SF del módulo LoRa E5 se determinó que el modo funcional es el DR0 que usa un SF de 10 y un ancho de banda de 125 kHz, esto debido a las configuraciones previas del gateway usado en trabajos de integración curricular anteriores y a las cuales no se tenía acceso. Es por esto que únicamente en este modo de velocidad la conexión era posible, sin embargo, no se necesitó realizar pruebas en los distintos modos, ya que como se pudo observar en la hoja de datos, los resultados entre uno y otro son bastante similares en las métricas que corresponden al interés del presente trabajo.

- Se consiguió aprovechar la facilidad de implementación de código Angular.JS en Node-Red para facilitar el monitoreo, esto a través de iconos dinámicos controlados por una entrada discreta de datos y el uso de condicionales y sentencias que permiten el manejo del DOM en tiempo real.

3.4 RECOMENDACIONES

- Es importante definir el funcionamiento que se quiere para el prototipo desde la fase de diseño, esto debido a que se puede obtener los resultados, ya sea programando la tarjeta directamente o usando los comandos AT, pero si se usa uno de los métodos no se puede utilizar el otro es por esto que para evitar demoras o daños permanentes en las tarjetas se debe definir que usar antes de comenzar la fase de implementación.

- Es importante comprender tanto la hoja de datos como la lista de comandos AT proporcionados por el fabricante, ya que el uso correcto de estos documentos puede evitar retrasos o inconvenientes como los presentados al querer usar el comando RSSI fuera del modo TEST de la Wio-E5, el problema se presentó debido a que el comando funciona únicamente en este modo y el mismo no permite la comunicación entre el nodo y el gateway, dejando así invalidada la red LoRaWAN diseñada. Sin embargo, en la misma lista de comandos se especifica que los mensajes pilotos para acuses de recibo también llevan como metadatos el RSSI que es el valor buscado, por lo que de esta forma se puede obtener esta medida en cualquiera de los modos de funcionamiento.

- Se recomienda usar tarjetas con más de 2 años de creación, esto con la finalidad de tener documentación suficiente al tiempo que se está realizando el trabajo de integración curricular, al usar tarjetas creadas recientemente se puede cometer errores de implementación o en su defecto realizar implementaciones ya realizadas por terceros, que por falta de información sobre el nodo no se toman en cuenta.

- Se recomienda apoyarse en plataformas como TTN, Arduino Cloud, Node-Red, Cayenne, LoRa Cloud, etc. Ya que estas facilitan implementaciones complejas como el desarrollo de dashboards, o la obtención de datos en formatos comprensibles, que de otra manera demorarían en gran medida la realización de trabajos como el presente, sin contar con que es posible que el apartado gráfico y el funcionamiento no sea el óptimo, niveles que si se logran al usar plataformas como las mencionadas anteriormente.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] M. A. M. Quimbita y C. E. P. Salvador, «Evaluación de pasarela LoRa/LoRaWAN en entornos urbanos,» es, pág. 40,

[2] J. Haxhibeqiri, E. De Poorter, I. Moerman y J. Hoebeke, «A Survey of LoRaWAN for IoT: From Technology to Application,» en, *Sensors*, vol. 18, n.º 11, pág. 3995, nov. de 2018, Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s18113995. dirección: https://www.mdpi.com/1424-8220/18/11/3995 (visitado 11-11-2022).

[3] A. Andreu Dólera, «Modelado de redes LoRaWAN aplicadas a la conservación preventiva de patrimonio cultural,» 2019.

[4] *What is Frequency-Shift Keying (FSK)?* es. dirección: https://www.techtarget.com/searchnetworking/definition/frequency-shift-keying (visitado 01-12-2022).

[5] *Forward Error Correction in Computer Networks*, es, Section: Computer Networks, jun. de 2020. dirección: https://www.geeksforgeeks.org/forward-error-correction-in-computer-networks/ (visitado 01-12-2022).

[6] M. Saari, A. M. bin Baharudin, P. Sillberg, S. Hyrynsalmi y W. Yan, «LoRa — A survey of recent research trends,» págs. 0872-0877, mayo de 2018. DOI: 10.23919/MIPRO.2018.8400161.

[7] D. Henandez, *El protocolo LORAWAN*, es-ES, sep. de 2019. dirección: https://www.zoostock.com/redes-y-sistemas/el-protocolo-lorawan (visitado 20-12-2022).

[8] T. T. Network, *LoRaWAN Architecture*, es. dirección: https://www.thethingsnetwork.org/docs/lorawan/architecture/ (visitado 20-12-2022).

[9] *LoRa-E5 Development Board - STM32WLE5JC*. dirección: https://doc.riot-os.org/group__boards__lora-e5-dev.html (visitado 28-12-2022).

[10] *Kit de desarrollo Wio-E5 - Seeed Wiki*. dirección: https://wiki.seeedstudio.com/LoRa_E5_Dev_Board/ (visitado 28-12-2022).

[11] *Arduino Nano | Arduino.cl - Compra tu Arduino en Línea*, es, ene. de 2019. dirección: https://arduino.cl/arduino-nano/ (visitado 28-12-2022).

[12] J. Damian, *Arduino Nano Pinout y características - Electrogeek*, es, Section: Arduino, mar. de 2020. dirección: https://www.electrogeekshop.com/arduino-nano-pinout-y-caracteristicas/ (visitado 28-12-2022).

[13] Mouser, *Sentrius RG1xx LoRa-Enabled Gateways - Laird Connectivity | Mouser*. dirección: https://www.mouser.ec/new/laird-connectivity/laird-sentrius-rg1-lora-gateway/ (visitado 28-12-2022).

[14] T. T. Network, *Laird RG1xx*, es. dirección: https://www.thethingsnetwork.org/docs/gateways/laird/ (visitado 28-12-2022).

[15] T. T. Network, *The Things Network - TTN*, es. dirección: https://www.thethingsnetwork.org/docs/quick-start/ (visitado 28-12-2022).

[16] pickdata, *Node-RED, the visual programming tool for Internet of Things*, en, Text, Last Modified: 2020-06-15T15:17+02:00, mayo de 2020. dirección: https://www.pickdata.net/es/noticias/node-red-programacion-visual-iot (visitado 28-12-2022).

[17] S. T. Co., *LoRa-E5 - LoRa Wireless Module - Powered by STM32WE5*, 2020. dirección: https://files.seeedstudio.com/products/317990687/res/LoRa-E5%5C%20module%5C%20datasheet_V1.0.pdf (visitado 04-01-2023).

[18] Tindie, *LoRa Antenna EU868 or US915 by M2M Shop on Tindie*, en. dirección: https://www.tindie.com/products/m2m/lora-antenna-eu868-or-us915/ (visitado 05-01-2023).

[19] S. Hosseinzadeh, *PYTHON/MATLAB optimization for log-distance model + Example*, en, sep. de 2019. dirección: https://la.mathworks.com/matlabcentral/fileexchange/70097-python-matlab-optimization-for-log-distance-model-example (visitado 05-01-2023).

[20] J. Caipa Roldan, *Okumura Hata model*, en, oct. de 2009. dirección: https://la.mathworks.com/matlabcentral/fileexchange/25941-okumura-hata-model (visitado 05-01-2023).

[21] S. T. Co., *LoRa-E5 AT Command Specification*, English, 2020. dirección: https://files.seeedstudio.com/products/317990687/res/LoRa-E5%5C%20AT%5C%20Command%5C%20Specification_V1.0%5C%20.pdf (visitado 04-01-2023).

[22] *915MHz 12dBi Omni Directional Outdoor Antenna (LoRa US915),* vi. dirección: https://ft-rf.com.tw/915mhz-12dbi-omni-directional-outdoor-antenna-us-915 (visitado 06-01-2023).

[23] *AngularJS: API: ngIf*. dirección: https://docs.angularjs.org/api/ng/directive/ngIf (visitado 20-01-2023).

[24] *Angular Material Icons*. dirección: https://klarsys.github.io/angular-material-icons/ (visitado 20-01-2023).

[25] E. B. Cadena, «Análisis de comunicaciones punto a punto con simulaciones open-source de LoRa,» es, *Revista de Investigación en Tecnologías de la Información*, vol. 10, n.º 21 (Especial), págs. 14-23, ago. de 2022, Number: 21 (Especial), ISSN: 2387-0893. DOI: 10.36825/RITI.10.21.003. dirección: https://riti.es/index.php/riti/article/view/15 (visitado 11-02-2023).

5 ANEXOS

ANEXO 1

```
clear
close all
clc
c=3e8;
D=0.1:10:10000;
Prx=27+2+3+20*log10(c./(4*pi.*D*915e6));

figure
plot(D,Prx,'LineWidth',3)
title('RSSI vs Distancia por ecución de Friis')
ylabel('RSSI (dBm)')
xlabel('Distancia (m)')
grid on
grid minor

fc=915e6;
hb=30;
hm=1.5;
flag=2;
OkuHata=zeros(length(D),1);

for i=1:length(D)
    OkuHata(i)=OkumuraHata(fc,hb,hm,(D(i)/1000),flag);
end
nPrx=Prx-(OkuHata/2.5)';
figure
plot(D,nPrx,'LineWidth',3)
title('RSSI vs Distancia con modelo Okumura-Hata')
ylabel('RSSI (dBm)')
xlabel('Distancia (m)')
grid on
grid minor
```

Código 1: Friis vs Distancia y Okumura Hata vs Distancia

ANEXO 2

```
freq= 915e6; %Transmission frequency
txPower= 27; % transmission power in dB
antennaeGain= 3;   % Total antennae gains (transmitter + receiver) in dB
refDist= 1;  % reference distance from the transceiver in meters
lightVel= 3e8;  % speed of light
refLoss= 20*log10((4*pi.*refDist.*freq./lightVel));% free space path loss at the
    reference distance (refDist)
ERP = txPower + antennaeGain - refLoss; % kind of the an equivalent radiation
    power
data = csvread('logDistanceData.csv');

%% Sorting data, totally optional
sortData = 1; % sorts data based on the distance if 1 enabled, otherwise disabled
    .
if sortData
    [data(:,1),tempInd] = sort(data(:,1),1);
    data(:,2) = data(tempInd,2);
end

x0 = [1,1];          % initial guess of the path loss exponent and system loss
lx = [1,-50];        %  lower and upper interval of solution space
hx = [15,50];        % Realisticly path loss should not exceed this

h =@(x)logDistOptimEng(x,ERP,refDist,data);
options = optimoptions('lsqnonlin','Display','iter');

%%
[xs,res,fval] = lsqnonlin(h,x0,lx,hx,options);  % Use this optimization
disp(optimizationEquations)

averageMSE = sum(fval.^2)./numel(fval);
disp(['loss exponent = ' , num2str(xs(1)),',  other losses = ', num2str(xs(2))])
disp(['RMSE = ',num2str(sqrt(averageMSE)), ',   Absolute Mean Error = ',num2str(
    mean(abs(fval))), ...
    ',   Mean Error = ',num2str(mean(fval))])
disp(['Fading Standard Deviation = ', num2str(std(fval)), ',   Fading mean = ',
    num2str(mean(fval))])
disp(['Path loss model ==> ERP - 10 * ',num2str(xs(1)),' * log(distance) - (',
    num2str(xs(2)),')'])
```

```
%%
figure
logDistEstimation = ERP - 10.* xs(1) .* log10(linspace(min(data(:,1)),max(data
    (:,1)),size(data,1))/refDist) - xs(2);
plot(linspace(min(data(:,1)),max(data(:,1)),size(data,1)),logDistEstimation,'
    linewidth',3)
grid on
xlabel('Distancia (m)');
ylabel('RSSI (dBm)');
title(['Log-distance (\gamma = ',num2str(xs(1)),')'])
```

Código 2: Modelo Log Distance

ANEXO 3

```
#include<SoftwareSerial.h>
SoftwareSerial e5(7, 6); // (RX, TX)

//Funciones
void initE5();
void enviarMensaje(String msj);
void joinE5();
void stringToHex(String s1);
void capBat();
void capRssi();

//Variables globales
unsigned long prevT = 0;
const long intervalo = 35000;
String inputString = "";
String output;
String newV1;

void setup() { //Funcion de inicializacion de arduino
  Serial.begin(9600); //Se inicia la comunicacion serial a 9600 baudios pc-
    arduino
  e5.begin(9600); //Se inicia la comunicacion serial a 9600 baudios arduino-
    wio
  initE5(); //Se realiza las configuraciones de inicio para la Wio-E5
  joinE5(); //Se inicia la comunicacion con el arduino
```

```
24 }
25
26 void loop() { //Funcion principal del programa
27    unsigned long nowT = millis(); //Se actualiza el tiempo actual
28   if (nowT - prevT >= intervalo) { //Se comprueba si se ha cumplido el
       intervalo
29     prevT = nowT; //Actualiza el tiempo previo
30     msjGW(); //Envia un mensaje al GW
31   }
32 }
33
34 void initE5(){ //Funcion para setear las configuraciones iniciales de la Wio
     -E5
35   delay(500); //Tiempo de guarda
36   Serial.println("Iniciando");
37   enviarMensaje("AT"); //Se comprueba que este lista la comunicacion serial
38   enviarMensaje("AT+ID"); //Permite verificar las credenciales de acceso del
       nodo
39   enviarMensaje("AT+DR=US915"); //Se establece la banda de frecuencias US915
40   enviarMensaje("AT+DR=DR0"); //Se establce SF de 12 y AB de 125kbps
41   enviarMensaje("AT+CH=NUM,0-15"); //Se habilitan los primeros 15 canales de
       comunicacion
42   enviarMensaje("AT+CLASS=A"); //Se setea al nodo como calse A
43   enviarMensaje("AT+MODE=LWOTAA"); //Se configura el metodo de activacion
       como OTAA
44   }
45
46 void joinE5(){ //Funcion para iniciar la comunicacion con el GW
47   Serial.write("Tx:AT+JOIN"); //Se muestra en el monitor serial ed arduino
       la accion a realizar
48   e5.println("AT+JOIN"); //Se inicia la comunicacion con el GW
49   delay(20000); //Tiempo de guarda
50   while(e5.available()){ //compone un string char a char
51     char inChar = (char)e5.read();
52     inputString += inChar;
53   }
54   Serial.print(inputString); //MUestra el string en el monitor serial
55   inputString = ""; //Limpia el string
56   }
57
58 void enviarMensaje(String msj){ //Funcion para enviar un mensaje arduino-wio
59   Serial.write("Tx:"); //Se muestra en pantalla
```

```
60    Serial.println(msj); //el mensaje a enviar
61    e5.println(msj); //Envio de mensaje desde arduino a wio
62    delay(2000); //Tiempo de espera para comunicacion efectiva
63    while(e5.available()){ //Lectura de buffer
64      char inChar = (char)e5.read(); //Lectura char a char
65      inputString += inChar; //Se compone el string char a char
66    }
67    Serial.print(inputString); //Se muestra en pantalla el string
68    inputString = ""; //Se limpia el string
69  }
70
71  void msjGW(){ //Funcion para enviar mensaje al GW
72    capRssi(); //se captura el RSSI
73    Serial.println("");
74    Serial.println("--------------------");
75    Serial.println("Resultado");
76    String res="{\"rssi\":"+newV1+"}"; //Se da formato al mensaje que se
        quiere enviar
77    Serial.println(res); //Se muestra el mensae que se enviara
78    Serial.println("--------------------");
79    Serial.println("");
80    Serial.println("");
81    Serial.println("--------------------");
82    Serial.println("Resultado");
83    stringToHex(res); //Se transforma el mensaje a hexadecimal
84    Serial.println("--------------------");
85    Serial.println("");
86    delay(5000); //tiempo de guarda
87    //------------------ Se limpian todas las variables ----------------------
88    output="";
89    res="";
90    while(e5.available()){
91      e5.read();          //Funcion para limpiar el buffer
92    }
93    inputString = "";
94    //--------------------------------------------------------------------------
95  }
96
97  void stringToHex(String s1){ //Funcion para transformar un string en
        hexadecimal
98  int hex_dec; //Variable local auxiliar
99  Serial.println();
```

```
100 Serial.print("string: ");
101 Serial.println(s1); //Se muestra el string inicial
102 Serial.println("hexval: ");
103 for (const auto &item : s1) { //Se convierte cada char del string a su
      equivalente hexadecimal y junta todo en una sola cadena
104   hex_dec = int(item);
105   char hexaDeciNum[100];
106   int i = 0;
107   while (hex_dec != 0) { //convierte cada char del string a su equivalente
      hexadecimal
108     int temp = 0;
109     temp = hex_dec % 16;
110     if (temp < 10) {
111       hexaDeciNum[i] = temp + 48;
112       i++;
113     }
114     else {
115       hexaDeciNum[i] = temp + 55;
116       i++;
117     }
118     hex_dec = hex_dec / 16;
119   }
120   for (int j = i - 1; j >= 0; j--){ //Compone la cadena final hexadecimal
121     output += hexaDeciNum[j];
122   }
123 }
124 output="AT+MSGHEX="+output; //Prepara el mensaje a enviarse al GW
125 Serial.println(output); // muestra por el monitor serial de arduino el
      comando a utilizar
126 e5.println(output); //envia el mensaje al GW
127 }
128
129 void capRssi(){ //Definicion de funcion sin retorno
130   inputString=""; //Se limpia el srting
131   Serial.println("\nTx:AT+MSG"); //Se muestra un mensaje en el monitor
      serial de arduino
132   e5.println("AT+MSG"); //Se env a el mensaje para acuse de recibo
133   delay(17000); //Tiempo de espera promedio en recibir acuse de recibo
134   while(e5.available()){ //Funcion para leer el buffer del comunicador
      arduino-wio
135     char inChar = (char)e5.read(); //Lectura char a char
136     if(inChar=='X'){
```

```
137         inputString=""; //Se limpia el string tras una X ya que se conoce el
        mensaje esperado
138         }
139       inputString += inChar; //Se compone el string char a char
140     }
141     Serial.println(inputString); //se muestra por pantalla el string compuesto
142     int pos1=inputString.indexOf("RSSI"); //Se busca la plabara clave RSSI
        para realizar la captura
143     if (pos1>=0) { //Funci n para capturar el RSSI enviado en el acuse de
        recibo.
144       Serial.print("se ha detectado la palabra RSSI en la posicion ");
145       Serial.println(pos1);
146       newV1=inputString.substring(pos1+4, inputString.length()); //Se separa el
          RSSI del resto del string
147       newV1=newV1.substring(0, newV1.indexOf(",")); //Se captura el RSSI
        enviado en el acuse de recibo.
148       Serial.println(newV1); //Se muestra por pantalla el valor
149     }
150 }
```

Código 3: Código de Arduino completo

ANEXO 4

La simulación realizada en Radio Mobile esta disponible en:

https://github.com/Elbony/TesisLoRa/blob/main/Simulaci%C3%B3nLoRaEB.net

ANEXO 5

```
1  clear
2  clc
3  close all;
4  T = readtable('pruebasTesisSis.xlsx');
5
6  % Define las coordenadas de los puntos en grados (latitud, longitud)
7  punto1 = [T.lat(1), T.long(1)];
8  punto2 = [T.lat, T.long];
9  Distancias=zeros(length(punto2),1);
10 for i=1:length(punto2)
```

```
% Conversión de grados a radianes
lat1 = punto1(1) * pi / 180;
lat2 = punto2(i,1) * pi / 180;
long1 = punto1(2) * pi / 180;
long2 = punto2(i,2) * pi / 180;

% Radio de la Tierra (en metros)
R = 6371e3;

% Cálculo de la distancia haversine
a = sin((lat2-lat1)/2)^2 + cos(lat1) * cos(lat2) * sin((long2-long1)/2)^2;
c = 2 * atan2(sqrt(a), sqrt(1-a));
dist = R * c;
Distancias(i)=dist;
fprintf('La distancia entre los puntos (%f, %f) y (%f, %f) es %f metros.\n',
    punto1(1), punto1(2), punto2(i,1), punto2(i,2), dist);
end
```

Código 4: Calculo de distancia entre dos puntos

ANEXO 6

Los resultados obtenidos tanto en las pruebas como en la simulación de Radio Mobile se encuentran en:
https://github.com/Elbony/TesisLoRa/blob/main/pruebasTesisSis.xlsx

ANEXO 7

```
clear
clc
close all;
T = readtable('DatosTesis.xlsx');
distancias=T.Distancias;
rssiNodo=T.RSSIGNodo;
rssiGW=T.RSSIGW;
simulacion=T.vRadioM;

figure()
```

```
11 subplot(2,1,1)
12 plot(distancias,rssiNodo,'-o','LineWidth',1)
13 title('RSSI Nodo VS Distancia')
14 ylabel('RSSI Nodo (dBm)')
15 xlabel('Distancia (m)')
16 subplot(2,1,2)
17 plot(distancias,rssiGW,'-o','LineWidth',1)
18 title('RSSI Gateway VS Distancia')
19 ylabel('RSSI GW (dBm)')
20 xlabel('Distancia (m)')
21
22 figure()
23 plot(distancias,rssiNodo,'-*','LineWidth',1)
24 grid on
25 grid minor
26 title('Resultados obtenidos')
27 ylabel('RSSI (dBm)')
28 xlabel('Distancia (m)')
29 hold on
30 plot(distancias,rssiGW,'-o','LineWidth',1)
31 plot(distancias,simulacion,'-+','LineWidth',1)
32 legend('RSSI medido en el nodo', 'RSSI medido en el GW', 'RSSI Simulado')
33 hold off
34
35 figure()
36 errorVal = abs((simulacion - rssiNodo) ./ rssiNodo) *100;
37 plot(distancias,errorVal,'-*','LineWidth',1)
38 title('Calculo del error porcentual')
39 ylabel('% de error')
40 xlabel('Distancia (m)')
41 grid on
42 grid minor
43 errorPorcentualMedio=mean(errorVal)
```

Código 5: Calculo de errores