ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

REDES DE SENSORES INALÁMBRICOS PARA IOT MEDICIÓN DE LOS RETARDOS DE RETRANSMISIÓN EN NODOS LORAWAN QUE UTILIZAN EACK E IACK

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

JONATHAN STEVEN JIMÉNEZ LOVATO
jonathan.jimenez03@epn.edu.ec

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

DMQ, abril 2023

CERTIFICACIONES

Yo, JONATHAN STEVEN JIMÉNEZ LOVATO 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.

Jonathan Steven Jiménez Lovato

Certifico que el presente trabajo de integración curricular fue desarrollado JONATHAN STEVEN JIMÉNEZ LOVATO, 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.

JONATHAN STEVEN JIMÉNEZ LOVATO
CARLOS ROBERTO EGAS ACOSTA. MSc.

DEDICATORIA

El presente trabajo es dedicado a cada una las personas mas importantes en mi vida, quienes con una palabra de aliento o con un consejo han cambiado cada paso que he dado.

En primero lugar a mis padres Luis y Patricia, gracias por su confianza y apoyo constante, por su infinito amor y palabras que en los días mas tristes han sido mi luz, son mi ejemplo de vida.

A mis hermanas Nicole y Sophia y a mi sobrino Leandro, quienes con su cariño y sonrisa cambiaron mi vida por completo. Gracias por ser mi motivo e impulso en cada momento, por acompañarme en cada uno de mis días a pesar de lo difícil de la situación.

A mis amigos, por todas las risas y aventuras compartidas a lo largo de mi carrera universitaria. Gracias por todo el tiempo compartido y cada palabra de aliento y por mantener una amistad tan sincera a pesar de los años.

El presente trabajo no solo es resultado de mi dedicación y esfuerzo, si no también de cada una de las personas que son importantes en mi vida. Gracias por estar siempre para mi, por ser mi motivo mi apoyo constante y sobre todo mi refugio en los dias mas tristes.

AGRADECIMIENTO

De inicio, gracias a mis padres Luis y Patricia, lo mas grande que tengo en mi vida, por su infinito amor y comprensión, por su apoyo de manera incondicional para continuar en mi carrera profesional y tener siempre una palabra de aliento para poder superar todos los obstáculos que se han presentado a lo largo de mi vida personal y profesional, lo que me ha llevado a tener éxito y poder formarme como persona.

A mi abuelo Luis, por impartir su sabiduría y cada una de sus historias. Gracias por aquellas risas y cada momento, un abrazo al cielo.

A mis grandes amigos y amigas, Lorena, Nubia, Elvis, Michael y Axel, que han sido parte de esta larga travesía, con quienes compartí miles de anécdotas, alegrías, tristezas y momentos únicos, gracias por su valiosa amistad, me siento afortunado de haber conocido a cada uno de ustedes, me llevo siempre lo mejor de cada uno de ustedes.

A la Escuela Politécnica Nacional y a todos los ingenieros que fuero parte primordial de mi formación profesional, por los conocimientos y valores impartidos que han logrado consolidar en mi un criterio de responsabilidad, honestidad y respeto, herramientas necesarios para afrontar los nuevos retos que se presenten en mi vida profesional.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En la actualidad las distintas tecnologías de red inalámbricas han evolucionado notablemente, surgiendo nuevas tecnologías, un ejemplo de eso es LoRaWAN el cual es un protocolo de red de la tecnología LORA que se utiliza principalmente para comunicar y administrar un sin numero de dispositivos. Es por ello que el presente trabajo tiene como objetivo medir los retardos existentes en una retransmisión de nodos LoRaWAN utilizando EACK e IACKS para lo cual se dividió en las siguientes etapas para el desarrollo del trabajo: En el primer capítulo se enfoca en la introducción. – Se profundiza los fundamentos teóricos de la tecnología LoRaWAN, detallando cada una de sus características principales, y mostrando los tipos de retardos que se tendrá que implementar al igual que las características mas relevantes del nodo a utilizar. El segundo capítulo se enfocará en metodología. – Muestra la fase de diseño y como consecuente la implementación, describiendo los equipos utilizados en el presente proyecto y como ultimo el diagrama de flujo del funcionamiento del código a implementar El tercer capítulo describe los resultados y discusión. – Evaluación del algoritmo implementado, al igual que el desempeño de la red entre los dos nodos E5 ubicando a una distancia adecuada para poder trasmitir tramas de una manera correcta y así evaluar las diferencias entre uno y otro método. Por último, en el capítulo cuarto se establece conclusiones y recomendaciones de las pruebas realizadas con el nodo E5 para el presente proyecto.

PALABRAS CLAVE: EACKS, IACKS, LoRa, modulo, retardo, transmisión.

ABSTRACT

Currently, the different wireless network technologies have evolved significantly, emerging new technologies, an example of this is LoRaWAN which is a network protocol of LORA technology that is mainly used to communicate and manage a number of devices. That is why this work aims to measure the existing delays in a LoRaWAN node retransmission using EACK and IACKS for which it was divided into the following stages for the development of the work: The first chapter focuses on the introduction. - The theoretical foundations of LoRaWAN technology are deepened, detailing each of its main characteristics, and showing the types of delays that will have to be implemented as well as the most relevant characteristics of the node to be used. The second chapter will focus on methodology. - It shows the design phase and consequently the implementation, describing the equipment used in this project and finally the flowchart of the code to be implemented. The third chapter describes the results and discussion. - Evaluation of the implemented algorithm, as well as the performance of the network between the two E5 nodes located at a suitable distance to be able to transmit frames in a correct way and thus evaluate the differences between one method and the other. Finally, the fourth chapter establishes the conclusions and recommendations of the tests performed with the E5 node.

KEYWORDS: Delay, EACKS, IACKS, module, LoRa, transmission.

1 INTRODUCCIÓN

Con el avance del internet agregando nuevos e innovadores protocolos de comunicación para las distintas tecnologías y ámbitos dieron lugar a un nuevo paradigma denominado Internet de las Cosas (IoT, Internet of Things). Las soluciones de IoT están siendo utilizadas en la agricultura, industria ciudades inteligentes entre otros. Donde, los dispositivos IoT facilitan su uso, con baterías y baja potencia de operación para redes de área amplia de baja potencia (LPWAN). Estas redes constan de dispositivos simples, económicos que se comunican a largas distancias con velocidades bajas de transmisión.

Un claro ejemplo de esto es el protocolo LoRaWAN, una de las tecnologías que más promete debido a que aprovecha al máximo el espectro extendido. Sin embargo, existen algunos factores que afectan la distancia de un enlace LoRaWAN, entre ellos están la temperatura, humedad, ruido y cambio climático. LoRaWAN presenta algunos retos, como la colisión de paquetes debido al aumento de distintos dispositivos en la red causando interferencia con otras tecnologías, provocando retardo en la transmisión de paquetes. En este trabajo se busca proporcionar más información sobre como utilizando métodos eACK e iACKS pueden mejorar los tiempos de retardo de retransmisión entre dos nodos LoRaWAN mediante el nodoE5 que es de fácil implementación en el enlace dado que consta de interfaces plug & play. Estos procesos son desarrollas a nivel de capa de enlace de datos dato que se requiere reducir el procesamiento de los nodos y así poder optimizar la transmisión de datos en el enlace evitando utilizar los servicios de capa de red.

Por último, se evaluará una expresión general que permitirá evaluar el retardo que existe de extremo a extremo con un tamaño previamente establecido de los paquetes, el cual permitirá reconocer cuál de los dos modos EACKS o IACKS permiten un mejor tener un tiempo de retardo aceptable.

1.1 OBJETIVO GENERAL

Estudiar la medición de retardos en una retransmisión de nodos LoRaWAN analizando los EACK e IACK en una red de sensores inalámbricos para IoT.

1.2 OBJETIVOS ESPECÍFICOS

1. Analizar el funcionamiento de los nodos LoraWAN que van hacer utilizados para la red de sensores inalámbricos.
2. Estudiar los EACKS e IACKS al ser implementados en nodos LoraWAN.
3. Analizar teóricamente los retardos utilizando EACK e IACKS.
4. Evaluar los retardos en el prototipo de red mediante los EACK e IACK

1.3 ALCANCE

La implementación del presente documento se centra en la medición de retardos en redes LoRaWAN utilizando EACKS e IACKS, para determinar cuál de ellos contribuye con mayor retardo en la transmisión de la información de nodo a nodo. El módulo a utilizarse para los nodos, es el nodoE5, el cual opera en la banda de 915 MHZ y emplea modulaciones LORA (G)FSK, BPSK y (G)MSK. La metodología que se prevé utilizar es cuantitativa donde se utilizara la recolección de información de distintos contextos científicos, permitiendo centrarse en aspectos que deben ser estudiados más a fondo.

Por ende, las fases para la presente implementación son:

- **Fase de planteamiento:** Se estudiará las características de transmisión de LoRaWAN, así como el funcionamiento de una transmisión confiable utilizando EACK e IACK. Es importante también analizar el formato de las tramas e LoRaWAN lo cual nos permitirá evaluar teóricamente los retardos en la transmisión al utilizar los métodos antes mencionados. Se estudiará las principales características del protocolo LoRaWAN, enfocándose en cómo funciona, como envía información de un nodo a otro y cuanto es el tiempo de espera para el envío previo de otro paquete ante la confirmación de un IACK o EACK Siendo este último el tema central para ser implementado en un módulo para LoRaWAN.
- **Fase de implementación:** Se definirá mediante el análisis de datos obtenidos, cual de las dos medidas de retransmisión es mejor al utilizar eACK o iACK en el protocolo

LoRaWAN. El módulo a utilizarse para los nodos, es el nodo E5, el cual opera en la banda de 915 MHZ y emplea modulaciones LORA (G)FSK, BPSK y (G)MSK. La metodología que se prevé utilizar es cuantitativa donde se utilizara la recolección de información de distintos contextos científicos, permitiendo centrarse en aspectos que deben ser estudiados más a fondo. Se analizará los tiempos de espera en cuanto al envío de paquetes entre dos nodos E5, los cuales serán configurados con paquetes permitiendo analizar a fondo la trama implementada y la previa confirmación mediante un de los dos métodos.

- **Fase de evaluación y análisis de resultados:** Se analizará los resultados obtenidos en cuando a la medición obtenida de los retardos en cada una de las pruebas realizadas en el envío y la retransmisión de paquetes en nodos LoRaWAN cuando se utiliza un eACK o iACK.

1.4 MARCO TEÓRICO

Se han elaborado numerosas soluciones para la implementación del internet de las cosas (IoT), abarcando diversos temas en distintos ambientes de las telecomunicaciones, específicamente en radio frecuencia (RF), aportando confiabilidad, robustez y sobre todo un continuo avance en la investigación de la redes de sensores inalámbricas, que han implementado nuevas tecnologías. Un claro ejemplo de esto es LoRaWAN un protocolo de red de LoRa que trabaja en capa física con números ventajas en la aplicación de redes de sensores inalámbricos tanto en costos, implementación y transmisión de información .

1.4.1 INTERNET DE LAS COSAS

Se denomina internet de las cosas (Internet of Things), a los distintos equipos que incorporan sensores, software entre otros, que permite intercambiar y conectar datos a través del Internet [1] [2].

1.4.2 REDES DE SENSORES INALÁMBRICAS

Se conoce como una red de sensores a los dispositivos autónomos que recolectan información de un determinado ambiente de manera colectiva, con la característica que se comunica inalámbricamente, es mas barata y ofrece un sistema flexible [3].

Una red de sensores inalámbricos o por sus siglas en ingles WSN (Wireless Sensor Network), es una red con numerosos dispositivos distribuidos espacialmente, que utilizan sensores para controlar diversas condiciones en distintos puntos, entre ellas la temperatura, el sonido, la vibración, la presión, el movimiento o los contaminantes [4].

1.4.3 REDES LPWAN

Las Redes de área amplia de baja potencia, o por sus siglas en ingles Low Power Wide Area Network (LPWAN), se utilizan en las telecomunicaciones para sistema inalámbricos y están destinadas al envió de datos de pequeño tamaño a un bajo consumo, el cual permita a los dispositivos IoT operar durante un rango de tiempo amplio sin necesitad de preocuparse por recargar o cambiar las baterías [5].

Su principal característica es que utiliza menos potencia que las redes WAN y soporta varios nodo para solo una estación base, y a su vez es compatible con varias tecnologías Sigfox, LoRa/LoRaWan, NB-IoT, LTE Cat M, entre otras [6] [7].

1.4.4 LoRA

LoRa es una tecnología que fue desarrollada por Semtech, un tipo de modulación de señales de RF que emplea una modulación de amplio espectro (spread spectrum), capaz de enviar pequeños paquetes en largas distancias y a su vez con un mínimo consumo de energía. Entre las principales características esta [6] [8]:

- Capacidad hasta 1 millón de nodos.
- Permite tolerar ruido y altas interferencias.
- Bajo consumo de energía (baterías de 3 a 5 años).
- Alcance de 15 a 20Km (linea de vista).

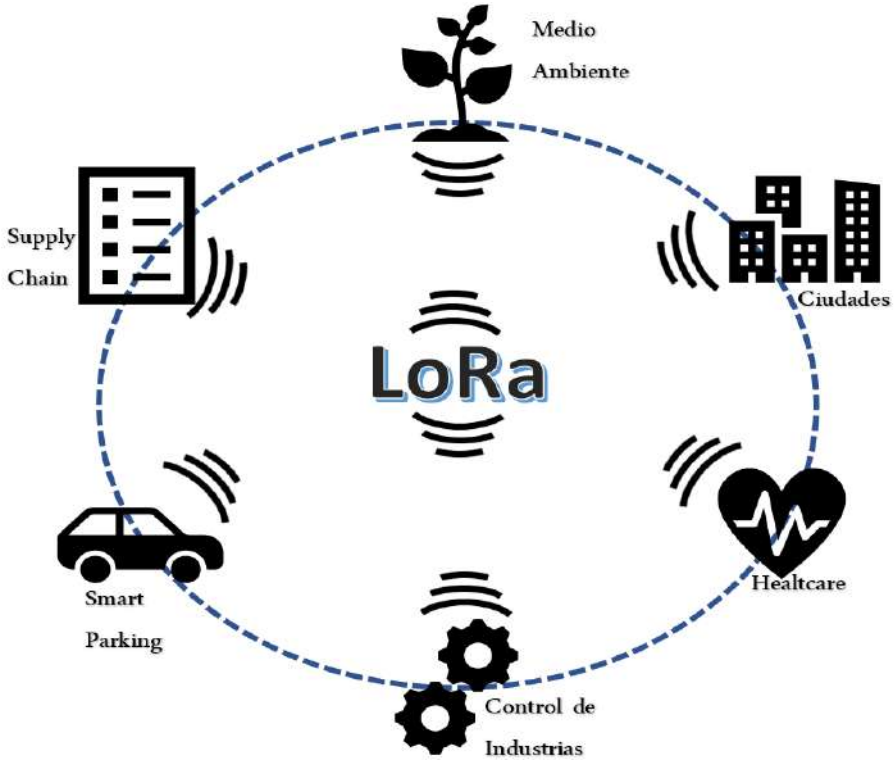


Figura 1.1: LoRa
Fuente: Autor

1.4.5 LoRaWAN

LoRaWAN es una tecnología inalámbrica de bajo consumo de energía que posibilita la transmisión de datos a larga distancia a través de una red de área amplia de baja potencia. Esta tecnología es especialmente apta para permitir la comunicación de dispositivos de bajo consumo energético, que envían pequeñas cantidades de datos de manera regular.

Para establecer una red LoRaWAN se requieren tres componentes principales: dispositivos finales, estaciones base (también llamadas gateways) y un servidor de red. Los dispositivos finales se comunican con las estaciones base, que luego envían los datos al servidor de red. Este último se encarga de la administración de la red y del envío de datos a las aplicaciones de usuario.

La tecnología LoRaWAN emplea la modulación de espectro ensanchado por secuencia directa para la transmisión de datos a través del aire. Gracias a esto, es posible transmitir datos a largas distancias, incluso en ambientes urbanos o interiores, con un consumo de energía reducido.[9].

1.4.5.1 Arquitectura LoRaWAN

La arquitectura LoRaWAN se compone de tres niveles principales: el nivel del dispositivo final, el nivel de acceso y el nivel del núcleo. Se considera como una estrella de estrellas, como se puede observar en la figura 1.2.

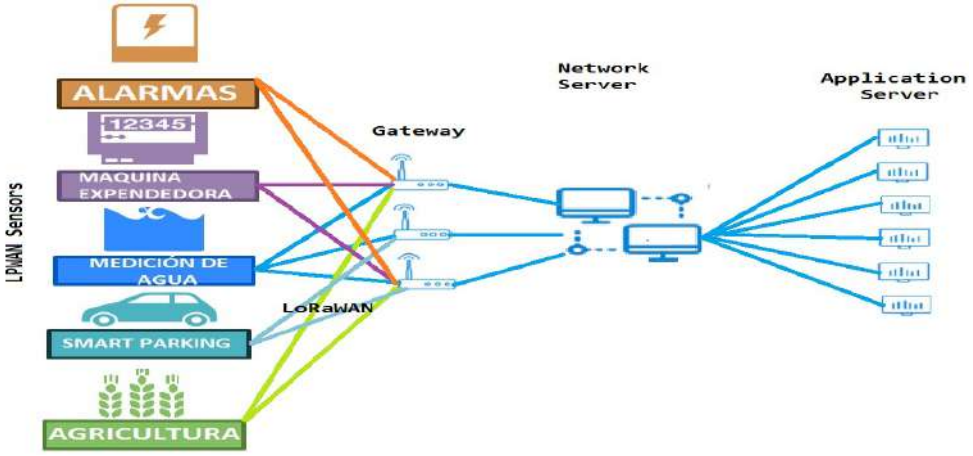


Figura 1.2: Redes LPWAN
Fuente: Autor

De acuerdo a la figura 1.2, está compuesta por varios componentes definidas como:

- **LoRA Sensors:** Son los dispositivos finales los cuales realizan las puertas de enlace para la comunicación de LoRa y LoRaWAN.
- **Gateway:** Son los encargados de enviar las tramas LoRaWAn desde los dispositivos finales a los servidores de red mediante una interfaz Ethernet 3G/4G o Wi-Fi.
- **Network Server:** Es el encaragado de decoficar los paquetes enviados por los dispositivos finales, a su vez se encarga de realiar comprobaciones de seguridad y velocidad de datos adaptable, para poder enviar de vuelta los dispositivos.
- **Application Server:** Recibe los datos del servidor de red y se encarga de decodificar los paquetes de seguridad y decide que acción va a ejecutar cada aplicación [10].

1.4.5.2 Bandas de frecuencias equipos LoRaWAN

LoRaWAN opera en bandas no licenciadas que se conocen como bandas ISM, son especialmente utilizadas para uso no comercial (Industria, Científica y Médica), con un mayor

rango al comparar con Wi-Fi, llevando a varias restricciones especificadas en cada país. Los módulos LoRaWAN disponen de 16 canales para Europa entre 433MHz y 868MHz, mientras 72 canales para los 900MHz.

En el presente trabajo se utilizara el nodoE5 en la banda de los 915MHz, donde en la tabla 1.1 se mostrara los canales de frecuencia empleados para LoRaWAN [11]:

Tabla 1.1: LoRaWAN bandas de frecuencias

Numero de canal	**Parámetros**	**Bandas de Frecuencia**	
		433 MHz	**868 MHz**
0	Frecuencia	433175 MHz	868100 MHz
	Ciclo de Trabajo	0.33 %	0.33 %
	Date Rate	0-5	0-5
	Estado	On	On
1	Frecuencia	433375 MHz	868300 MHz
	Ciclo de Trabajo	0.33 %	0.33 %
	Date Rate	0-5	0-5
	Estado	On	On
2	Frecuencia	4333575 MHz	868500 MHz
	Ciclo de Trabajo	0.33 %	0.33 %
	Date Rate	0-5	0-5
	Estado	On	On
3	Frecuencia	868325 KHz - 869750 KHz	868325 KHz - 869750 KHz
	Ciclo de Trabajo	*	*
	Date Rate	0-7	0-5
	Estado	Off	Off
$T_{off} = \frac{Tiempoalaire}{SubbandaCiclodetrabajo} - TiempoalAire$			

1.4.5.3 Clases de dispositivos:

Segun las especificaciones definen 3 tipos de nodos en LoRaWAN, como se puede observar en la Figura 1.3, donde topos los dispositivos LoRaWAN tienen las funciones dde Clase A [12].

- **Clase A (Dispositivos bidireccionales):**

  Se caracteriza por tener comunicación bidireccional, es la mas utilizada en todos los

dispositivos LoRaWAN debido a que permite el máximo ahorro de energía dado que entra en modo escucha después de haber enviado datos al gateway.

- **Clase B (Dispositivos bidireccionales con ranura programadas):**

  Dispone de ventanas de recepción con base a tiempos predeterminados con el Gateway, y tiene un mayor consumo de energía a diferencia de la Clase A dado que tiene recepción consecutiva.

- **Clase C (Dispositivos bidireccionales con recepción máxima):** Esta disponible siempre para recibir mensajes pero con la características que las ventanas en Rx están cerradas al estar enviando información, a su vez el tiempo de recepción es continuo, ofreciendo el menor ahorro de energía [12].

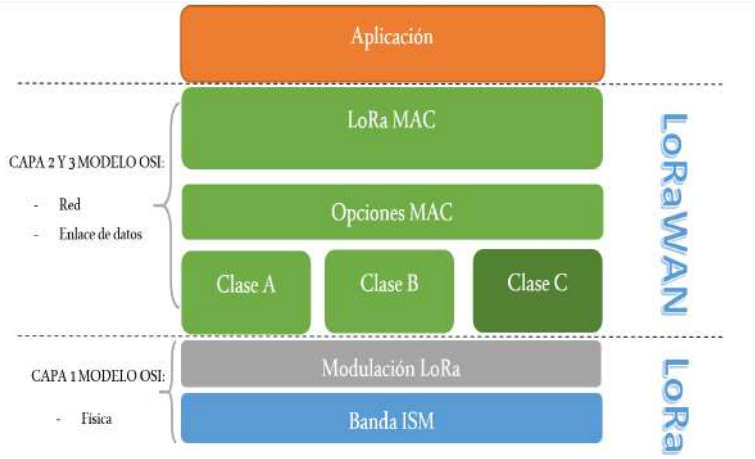


Figura 1.3: Clases LoRaWAN
Fuente: Autor

1.4.5.4 Estructura de paquetes LoRaWAN

Los paquetes LoRaWAN tienen una estructura definida, debido a que el estándar LoRa le permite funcionar en la capa física habilitando enlaces de RF en la banda ISM, LoRaWAN añade 2 capas mas en la capa Física (Enlace de Datos y Aplicación), como se puede observar en la Figura 1.3 [13].

Tabla 1.2: Estructura Trama Física de LoRa

Preámbulo	FHDR + PHDR_CRC	PHYPAYLOAD	CRC

- **Preámbulo:** Define el esquema de modulación del paquete, puede variar y va a depender del como se va a utilizar el nodo, su longitud es de 8 Bytes.

- **FHDR + PHDR_CRC:** Consta de una longitud de 20 bits y se encuentran codificado a la tasa mas confiable y a su vez un código de redundancia cíclica el cual evita tener tramas erróneas agregando redundancia a la trama.

- **PHYPAYLOAD:** Este parte de la trama contiene la información que va hacer enviada, y la conforman: MAC Header, MAC Payloady el MSI los cuales se pueden observar en la tabla 1.3.

Tabla 1.3: Trama MAC dentro de PHYPayload

PHY Payload		
MAC HEADER 1 byte	MAC PAYLOAD M bytes	MIC 4 bytes

  - *MAC Header:* Esta capa define el tipo de mensaje (datos o gestión)y la versión final de la trama especificada que ha sido codificada.
  - *MAC Payload:* Esta conformado por 3 campos: Frame Header, Frame Port y Frame Payload, utilizada para la autenticación de dispositivos.
  - *MIC:* Este campo es una clave para evitar la falsificación de mensajes y generada por los campos MHDR y MACPayload [14].

- **CRC:** Este campo permite agregar redundancia y asi asegurar que la trama es la correcta y puede ser enviada por los componentes de la red LoRa [13], [14] .

Tabla 1.4: Capa Aplicación dentro de MAC Payload

MAC Payload		
Frame Header 7 -22 bytes	Frame Port 1 bytes	Frame Payload N bytes

  - *Frame Header:* Identifica a la red y a su vez contiene un ontador de mensajes y el orden en el que deben ser transmitidos.
  - *Frame Port:* Permite identificar si el Frame Payload contiene datos de aplicación o comandos MAC.
  - *Frame Payload:* Permite cifrar la información a transmitir mediante un algoritmo AES128 usando APP_Skey.

1.4.6 COMUNICACIÓN ENTRE NODOS

Existe una gran diferencia con las redes LoRa debido a que en LoRaWAN los sensores se comunican con los servidores a través del gateway, esto a pesar que la comunicación se realice de la misma manera que LoRa, es distinta ya que los paquetes LoRaWAN se convierten estos paquetes en UDP y viceversa. El gateway recibe los mensajes desde sensores y los envía al Upstream (servidor de red) y los mensajes de configuración de como va ha estar establecida la red hacia los sensores Downstream [15].

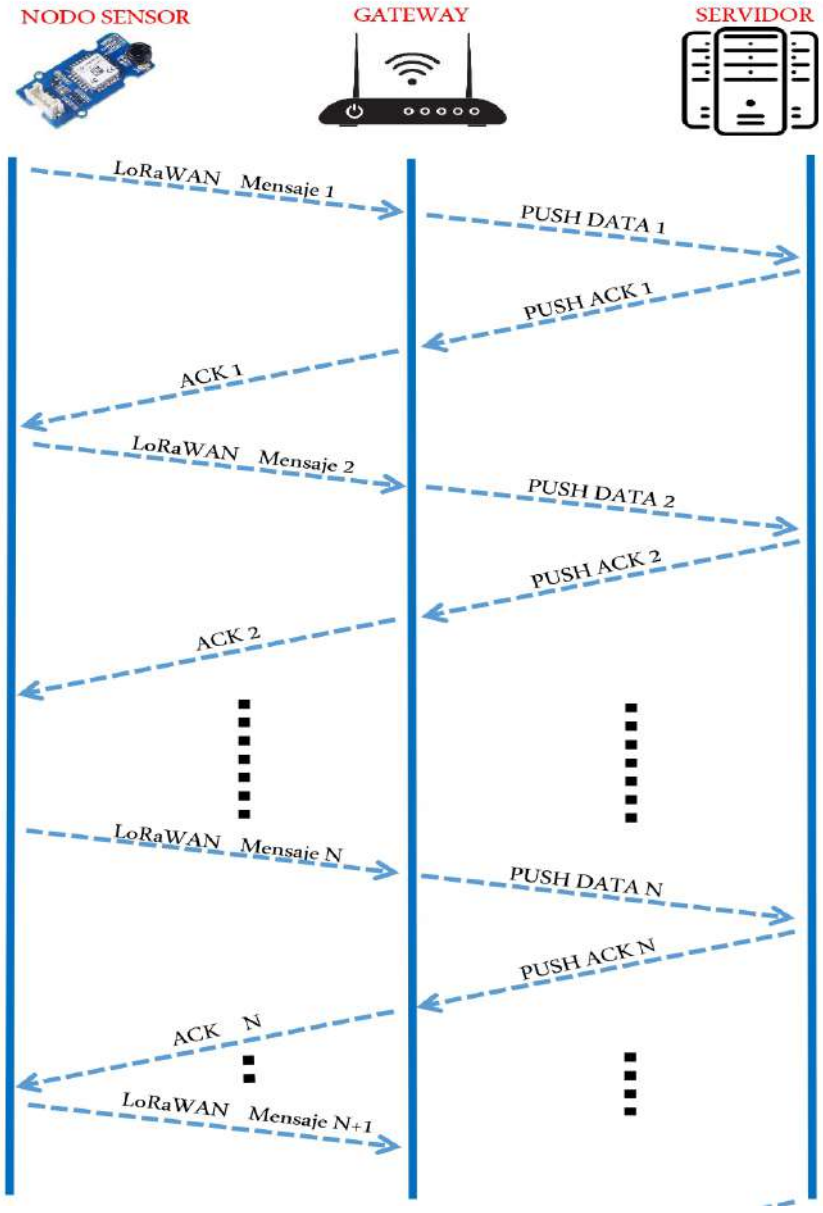


Figura 1.4: Comunicación Nodo-Gateway-Servidor (Upstream)
Fuente: Autor

Como se puede observar en la figura 1.4 se transmiten N mensajes seguidos y a su vez debe tener habilitados N canales. Al transmitir un mensaje del nodo al servidor, este llega al gateway donde se lo convierte en un paquete UDP, seguido lo reenvía al servidor de red y esto envía una confirmación de vuelta al gateway y al nodo.

Por otro lado, los mensajes PUSH DATA, incluyen la información de los nodos sensores (MAC gateway y un token aleatorio), donde este envía nuevamente un PUSH ACK con el mismo token generado una vez enviado el ACK correspondiente para procesar los paquetes.

Para downstream, se puede observar en la figura 2.6, donde se realiza la comunicación entre el Servidor y el Gateway, enviando paquetes PULL DATE periódicamente para mantener el NAT abierto y así el servidor pueda enviar un PULL RESP a través del mismo puerto por el que se envían los PULL DATE [16].

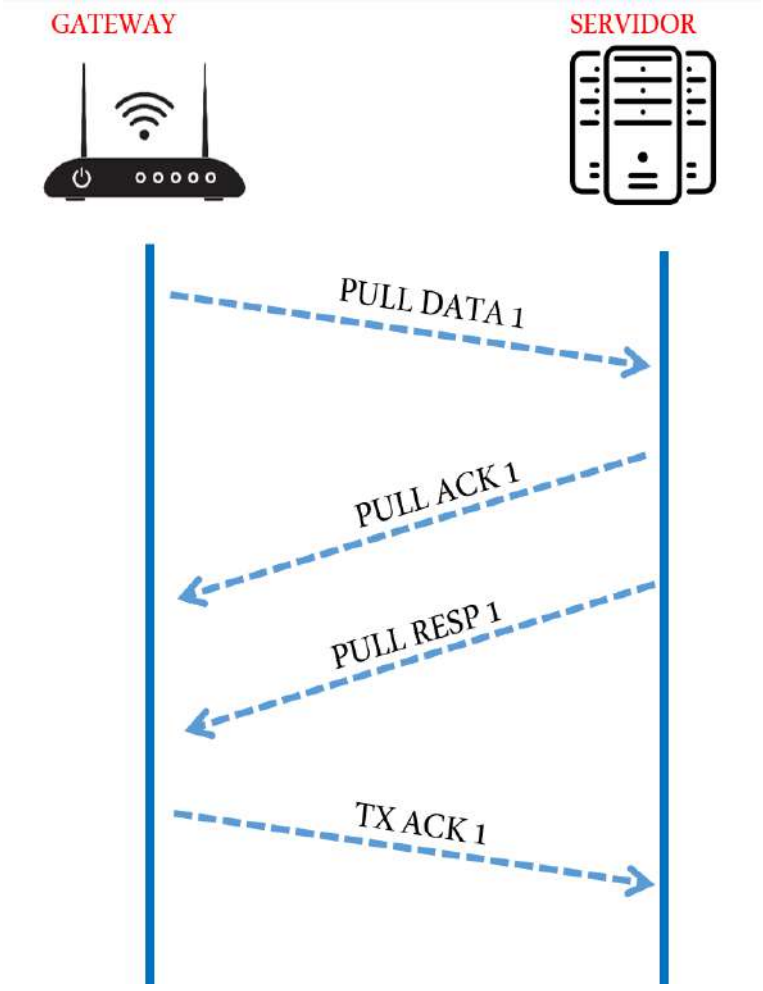


Figura 1.5: Comunicación Gateway-Servidor (Downstream)
Fuente: Autor

1.4.7 RETARDO DE PROPAGACIÓN EN LOS NODOS

En términos de telecomunicaciones el retardo (conocido como latencia), se define como el tiempo en el que un mensaje (paquete), tarda en llegar a su destino de manera completa una vez que el bit inicial de transmisión se envié desde el origen. El retardo se puede calcular sumando los siguientes componentes: retardo de propagación, retardo de trasmisión, retardo de cola y de procesamiento. Donde cada uno de esto serán detallados a continuación [17].

- **Retardo de Procesamiento:**
  Se conoce como el tiempo que un dispositivo intermedio en la comunicación necesita para tomar la decisión de por cual interfaz reenvía el paquete. Las características de velocidad que tendrá dependerá mucho de:

  - El protocolo y el procesador del equipo.
  - El tipo de arquitectura que disponga la red para minimizar retrasos.

- **Retardo de Transmisión:**
  Es el tiempo en el que tarda un paquete en ser transmitido desde el host al enlace de transmisión, esto se vera afectado por el tamaño de los datos y a su vez el ancho de banda del canal, este retardo se puede calcular mediante la ecuación 1.1.

$$T_t = \frac{L}{B} \tag{1.1}$$

  Donde:

  - L: Longitud de los datos (Bit).
  - B ancho de banda (bps).

- **Retardo de Propagación:**
  Es el tiempo que tarda el ultimo bit de los datos transmitidos en llegar al destino, después de que este se transmite al medio de propagación. Los factores que mas afectan a este tipo de retardo son:

  - Distancia (D): debido a que si el medio en el cual va hacer transmitido el paquete, tarda mas en llegar al destino.

- Velocidad (V): la velocidad de la velocidad debe ser menor a la del paquete.

Este retardo puede ser calculado mediante la ecuación 1.2.

$$T_p = \frac{D}{V} \tag{1.2}$$

- **Retardo de Cola:**

Es la cantidad de tiempo que tiene que esperar el paquete en cola antes de ser procesado, este debe esperar en un ”buffer” y depende de los siguientes factores:

- Si es mayor el numero de enlaces o servidores, mayor sera el retraso en cola.
- El procesador que dispone el equipo y los protocolos que puedes utilizarse para optimizar los retardos.
- Si el tamaño de la cola es grande, mayor sera el retraso en la cola [18].

2 METODOLOGÍA

En la presente sección se mostra el desarrolló del proyecto el cual es de tipo investigativo, aplicado y experimental, el cual se divide en cuatro fases: la primera la cual se define los distintos programas y materiales a utilizar, junto con sus respectivas características, en el segundo apartado indica el diseño funcional para la medición de retardos en nodos LoRaWAN, en la tercera parte muestra la configuración realizada al prototipo y códigos implementados. Finalmente, en el ultimo apartado se mostrará las pruebas realizas para alcanzar los objetivos planteados.

2.1 ELEMENTOS QUE COMPONEN EL PROTOTIPO DE RED LORAWAN

En la figura 2.1 se observa la Red LoRaWAN a implementar una vez se haya configurado los distintos elementos, el consta de nodos dos Nodos-E5 STM32WLE5JC para transmitir la informacion, un Gateway RG191 el cual permite la conexión a TTN (The Thing of Networks), el depurador/ programador ST Link v2 que permite implementar la software para los nodos LoRa y las fuentes de alimentación, en este caso las 18650.

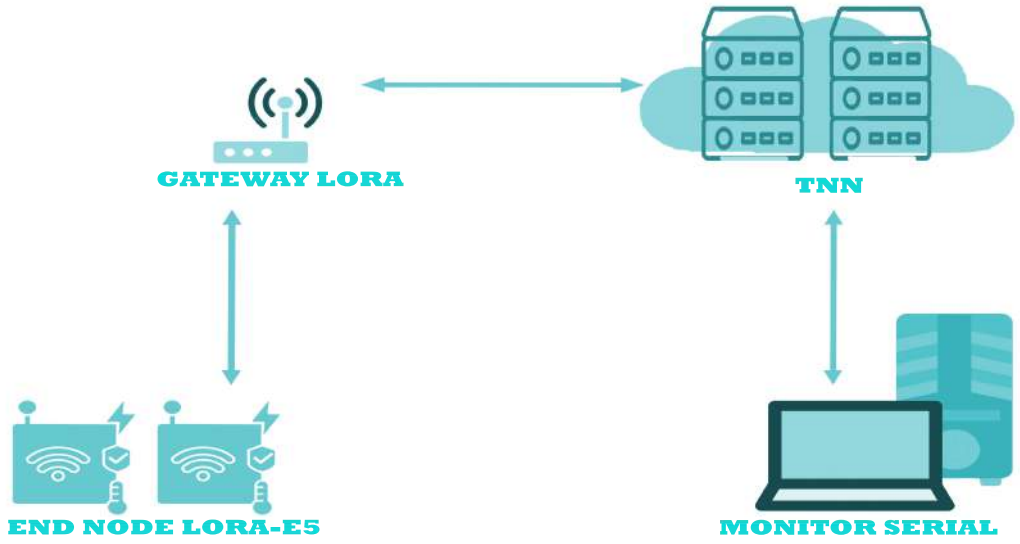


Figura 2.1: Esquema de la red LoRaWAN a implementar

2.1.1 The Thing of Networks

TTN es una red global de código abierto que provee una infraestructura de red a desarrolladores, empresas y otros usuarios para crear aplicaciones y servicios IoT. Su objetivo es ofrecer una plataforma abierta y colaborativa para conectar dispositivos a través de LoRaWAN y hacer que esta tecnología sea más accesible. La red está formada por una comunidad global de desarrolladores y entusiastas de IoT que trabajan juntos para mantener y mejorar la red [8].

Los usuarios de TTN pueden construir y lanzar sus propias aplicaciones IoT, conectar dispositivos a la red TTN y utilizar los servicios de la red, que incluyen la gestión de dispositivos, la integración de datos, la visualización de datos y la gestión de usuarios. Además, TTN cuenta con una API abierta que permite la integración con otros servicios y plataformas de IoT.

TTN opera como una red de código abierto, lo que significa que los usuarios pueden contribuir a su desarrollo y mejora. TTN también ofrece opciones de conectividad de red privada para permitir a los usuarios desplegar y operar su propia red LoRaWAN en entornos cerrados [19].

2.1.2 Dispositivo Final o Nodo

Un dispositivo LoRaWAN es aquel que se conecta a través de la tecnología inalámbrica de largo alcance LoRa, permitiendo la transmisión y recepción de datos de forma inalámbrica. Estos dispositivos pueden ser sensores, medidores o cualquier otro tipo de dispositivo capaz de recopilar información y enviarla a través de la red LoRaWAN. Los dispositivos suelen tener una potencia de transmisión baja y una duración prolongada de la batería [20].

2.1.2.1 Nodo LoRaWAN E5

El módulo LoRaWAN E5 se caracteriza por ser de bajo costo y a su vez de un bajo consumo de un diseño compacto que brinda un alto rendimiento. Contiene un chipset STM32WL5JC. Este módulo también está integrado con MCU de potencia ultrabaja ARM Cortex M4 y LoRa® SX126X y, por lo tanto, es compatible con el modo (G)FSK y LoRa®. El ancho de

banda con el que trabaja es variado de 62,5 kHz, 125 kHz, 250 kHz y 500 kHz, lo que lo hace adecuado para el diseño de varios nodos de IoT, compatibles con EU868 y US915 [21].

En la figura 2.2 se muestra el diagrama de pines del módulo a utilizar LoraE5 STM32WL5JC, de la marca LoRa® de Semtech Corporation. En la tabla 2.1 se muestra las especificaciones técnicas, que muestran las razones por las cuales fue elegido este módulo, como es el alto rendimiento dado su sensibilidad de 136.5 [dBm] y uso a larga distancia.

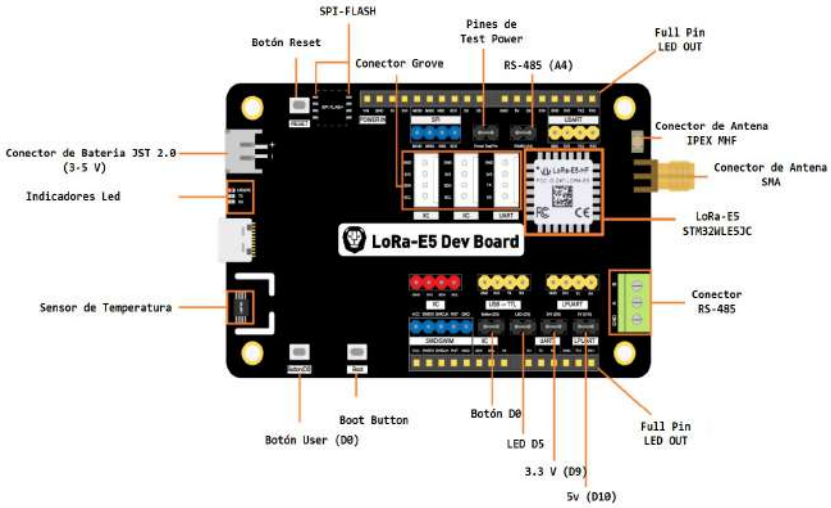


Figura 2.2: Nodo-E5 Especificaciones de Hardware.

Tabla 2.1: Tabla de especificaciones Nodo-E5.

Características	Especificaciones
Tamaño	Placa: 85.6 x 54 mm
Volataje IN	Bateria: 3-5 [V] Cable Tipo C: 5 [V]
Voltaje OUT	3.3 [v] - 5 [V]
Corriente	En modo de trabajo a 2.1[uA]
Frecuencia	EU868, US915, AU915, AS923, KR920, IN865
Modulación	BPSK, (G)MSK, (G)FSK, LoRa ®
Temperatura	-40℃~ 85℃

2.1.3 ST Link v2

Es un depurador y programador utilizado para los microcontroladores STM8 y STM32 respectivamente. El cual para comunicar con los distintos procesadores de las placas de desarrollo utilizan: la interfaz de un solo cable (SWIM) y un cable serial para depuración (SWD). Una de las características mas relevantes es que utiliza las aplicaciones STM32 para la su interfaz USB y así utilizar la velocidad completa y establecer un comunicación estable con las distintas aplicaciones de desarrollo (Tasking, Keil, etc) [22].

Figura 2.3: ST-LINK V2

En la siguiente tabla 2.2 se define la distribución de pines para el ST-LINK V2, donde se tiene distintas características como:

- Alimentación: 5 [V] mediante USB 2.0.
- Modo SWIM para alta y baja velocidad, SWD para JTAG/depuración.
- Velocidad de programación
  - Baja velocidad: 9,7 [Kbps].
  - Alta velocidad: 12,8 [Kbps].
- Actualizacion mediante modo DFU [23].

Tabla 2.2: Especificación de Pines ST-LINK V2

Pines ST-LINK V2			
1.	RST	2.	SWDIO
3.	GND	4.	GND
5.	SWIM	6.	SWCLK
7.	3.3 [V]	8.	3.3 [V]
9.	5 [V]	10.	5 [V]

2.1.4 Arduino Nano

Arduino Nano es una versión compacta y de bajo costo de la placa Arduino. Es similar a otras placas de la familia Arduino, pero tiene un tamaño mucho más reducido (aproximadamente 4,5 cm x 1,8 cm) y cuenta con un menor número de pines de entrada y salida.

Arduino Nano ofrece características similares a otros modelos de placas Arduino, incluyendo un microcontrolador ATmega328P, que se puede programar utilizando el entorno de desarrollo integrado de Arduino (IDE), y una serie de pines de entrada/salida que permiten conectar sensores, actuadores y otros dispositivos electrónicos [24].

2.1.5 Gateway Sentrius G191

El G191 es un Gateway de bajo costo y alto rendimiento que utiliza tecnología LoRaWAN para conectar sensores y dispositivos a la nube. Está diseñado para su uso en entornos industriales, comerciales y de Iot (Internet de las cosas), donde se requiere la comunicación inalámbrica de datos a larga distancia. El Gateway Sentrius G191 es compatible con múltiples protocolos de comunicación inalámbrica, incluyendo Bluetooth Low Energy (BLE), Zigbee y Wi-Fi, lo que lo hace extremadamente versátil en términos de conectividad. El dispositivo cuenta con una amplia cobertura inalámbrica de hasta 10 km en áreas urbanas y hasta 15 km en zonas rurales, lo que lo hace ideal para entornos donde la conectividad es crítica y donde la transmisión de datos inalámbrica de larga distancia es necesaria [25].

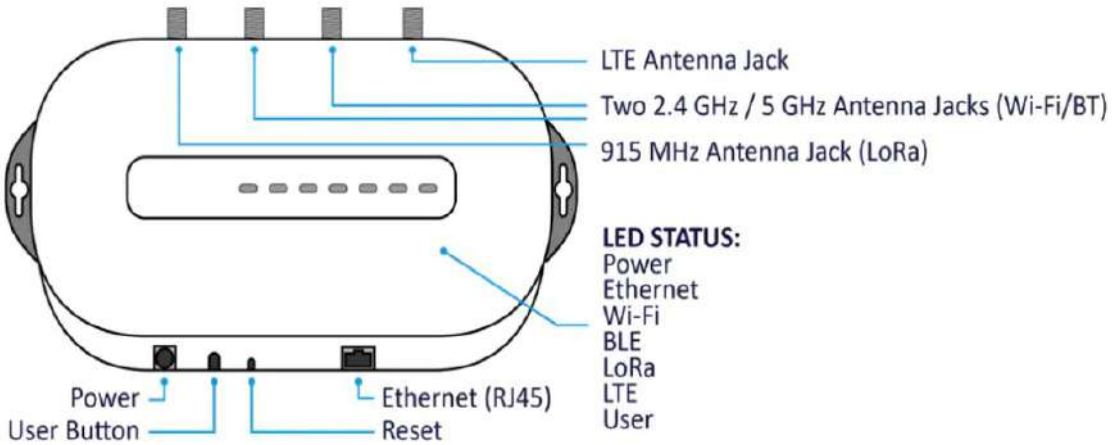


Figura 2.4: Gateway G191 [25].

2.1.6 Batería Tipo 18650

Este tipo de baterías se caracterizan por disponer de una celda de iones de litio, fabricado para la acumulación energía eléctrica y cuyo con un diámetro de 18 [mm] y una longitud en forma circular de 65 [mm], proporcionando una capacidad superior a las pilas tradicionales [26].

Figura 2.5: Batería 18650 [26].

Disponen de una alta densidad de energía por lo cual almacenan mayor carga por unidad de volumen, y gracias a ello son utilizadas en distintos en distintos aparatos electrónicos, desde el más pequeño (linternas, controles, radios portátiles, etc), hasta aparatos que requieren una gran capacidad de potencia [27]. Disponen distintas características que están detalladas en la siguiente tabla (2.3):

Tabla 2.3: Características de la batería 18650 [27]

Características	Descripción
Voltaje de carga	Máximo: 4.2 [V].
Voltaje de descarga	Minimo: 3[V].
Voltaje nominal	3.6 [V] DC.
Corriente nominal	2.850 [mA]
Tiempo de carga	3.5 horas aprox.
Dimensiones	1.4 [mm] x 65 [mm]
Peso	48 [g] aprox.

2.2 INTEGRACIÓN DE LOS COMPONENTES DE LOS NODOS A LA RED LORAWAN

Es importarte detallar los distintos softwares para el correcto funcionamiento y ejecución del firmware a cargar en el programador y a su vez en el nodo LoRa-E5, los cuales se pueden observar en la tabla 2.4:

Tabla 2.4: Materiales a utilizar para la Red LoRaWAN.

Elemento	Característica
Laptop	Intel Core I7 8va G. 16 Gb RAM.
Gateway	Gateway G191.
Nodo	LoRa-E5 STM32WLE5JC.
Arduino	Arduino Nano SAMD21.
Programador	ST-Link V 2.0.
Cables	Dupont (Hembra-Macho).
Programas	1. STM32 ST-Link Utility. 2. STM32CubeIDE. 3. STM32CubeProgrammer. 4. ArduinoIDE.

Como primer paso se debe descargar los drivers tanto para el nodo LoRa-E5 y el programador ST-Link V2, estos archivos se pueden encontrar en la pagina oficial de STM32 para el programdor, los drivers permite evitar inconvenientes a la hora de conectar estos dispositivos al portátil y no sean desconocidos .

Una vez instalado los drivers de los dispositivos antes mencionados, se empieza con la descarga e instalacion de los programas antes mencionados (*STM32CubeIDE, STM32CubeProgrammer y STM32 ST-Link Utility*) de la pagina oficial de STMicroelectronics, al igual que el software STM32 ST-LINK Utility, el cual se encuentra como STSW-LINK004 v4.6 y también el programa STM32Cubeprogrammer v2.12, en este caso se descargo la versión para windows de 64b, como se observa en la siguiente figura:

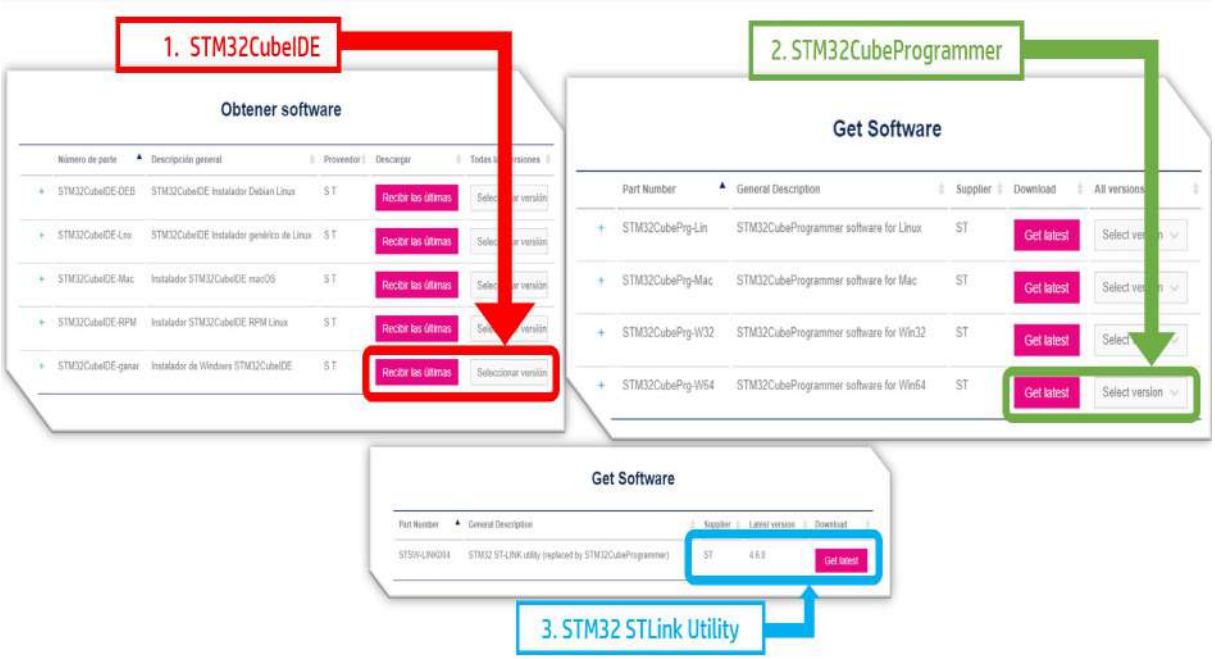


Figura 2.6: Descarga de los programas: STM32CubeIDE, STM32CubeProgrammer y STM32 ST-Link Utility.

A continuación se presenta una descripción breve de cada uno de los programas a utilizar.

1. **STM32CubeIDE:** Es un software de desarrollo multiplataforma en lenguaje C/C++, que permite la generación, compilación y depuración de código para los distintos microprocesadores y/o microcontroladores de la familia STM32. STM32CubeIDE incluye funciones de depuración básicas y para desarrolladores que permiten leer los registros del núcleo de la CPU, los registros periféricos y memorias de los microcontroladores, por otro lado también permite controlar las variables en vivo, la interfaz Serial Wire Viewer o el analizador de fallas [28].

2. **STM32CubeProgrammer:** Es una herramienta multi-OS (Windows, Linux, MacOS), para programar productos STM32, este software permite manejar un eterno eficiente para verificar la memoria de un dispositivo mediante dos interfaces: interfaz de depuración (JTAG y SWD) y la interfaz del cargador de arranque (UART, USB DFU, etc) [29]. Donde las características más relevantes son :

   - ✧ Actualización de firmware ST-Link.
   - ✧ Borra, programa y verifica memorias externas, el cual permite la creación segura de firmware.
   - ✧ Flasheo y arranque de periféricos de la serie STM32MP1.

3. **STM32 STLink Utility:** Es una software que permite programar las funciones para los microcontroladores STM32, que ofrece características como Flasheo, verificación del contenido de programación, y automatizar de la misma realizando una verificación durante y después de la programación para los STM32. STM32 ST-LINK contiene una interfaz gráfica de usuario (GUI) y una interfaz de línea de comandos (CLI) [30].

Una vez realizada la instalación de los distintos programas, es importante conectar el programador con el nodo LoRa-E5 siguiendo la distribución de pines tal como se muestra en la tabla 2.5

Tabla 2.5: Conexión de pines entre el programador ST-LINK V2 y nodo LoRa-e5

ST- LINK V2 (Pines)	LoRa-E5 (Pines)
2. SWDIO	DIO
4. GND	GND
6. SWCLK	CLK

En la figura 2.7 se puede observar la conexión de pines entre el nodo LoRa-E5 y el programador ST-Link v2.0, es importante recalcar que solamente se esta utilizando 3 pines del debido a que para la alimentación del nodo LoRa-E5 se esta utilizando la alimentación mediante el cable USB-Type C, si se desea alimentar la el nodo mediante un cable dupont hembra-macho se debe conectar desde el Pin 8 (3.3 [V]) hace el pin 3.3 [V] del nodo y a su vez desconectar el cable USB del LoRa-E5, esto debido a que se puede dañar, dado que se puede tener una sobrealimentación de la placa.

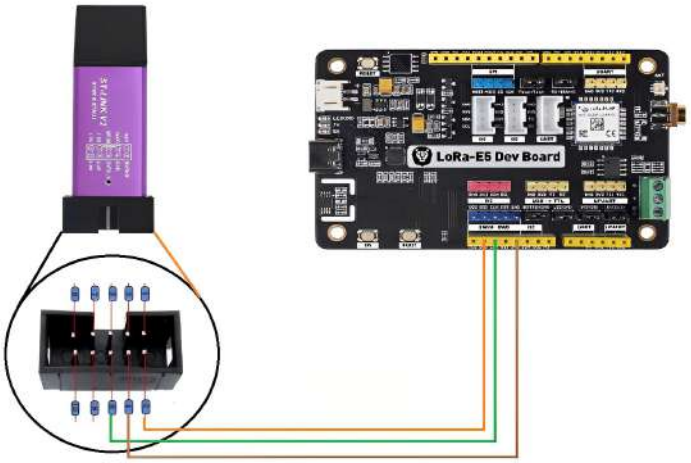


Figura 2.7: Diagrama de conexión entre el ST-Link V2 y el nodo LoRa-E5.

Una vez realizado la conexión entre el Modulo ST-Link V2.0 y el nodo se conecta al puerto USB de la portátil como se muestra en la figura 2.8. Seguido se debe realizar la actualización del programador ST-Link v2.0 mediante el programa *STM32 ST-Link Utility*.

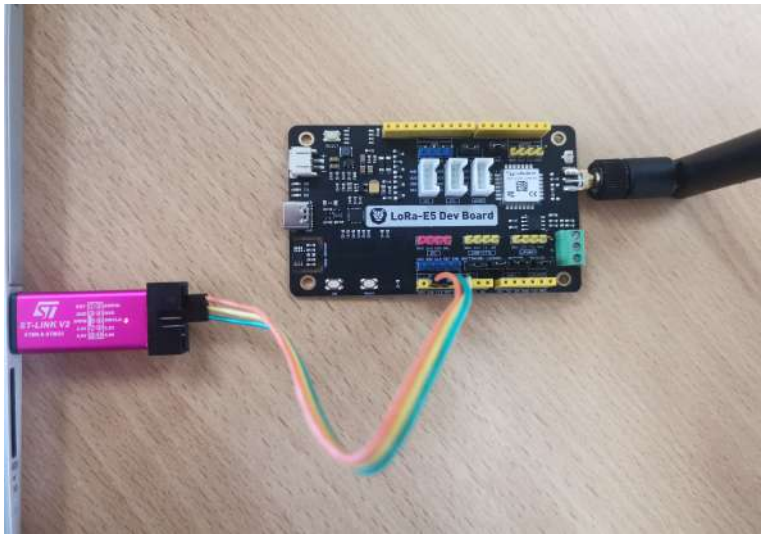


Figura 2.8: Conexión entre ST-Link V2- LoRa-E5 y la portátil.

Al ejecutar el programa STM32 ST-Link Utility (2.9, debemos seleccionar la pestaña que se muestra en la cinta de opciones ST-Link y seleccionar la opción *"Firmware Update"*, seguido se abre una nueva ventana mostrando las opciones como se observa en la figura 2.10, si no se reconoce el programador y se muestra el siguiente error *"ST-Link is not in the DFU mode"*, se debe desconectar y volver a conectar el programador, seguido se da clic en *Device Connect* y y clic en *Yes*, donde aparecerá un mensaje mostrando que el ST-Link se actualizo correctamente. Por ultimo se debe realizar el reemplazo del firmware del nodo LoRa-E5 para evitar inconvenientes a la hora de cargar el firmware con el programador.

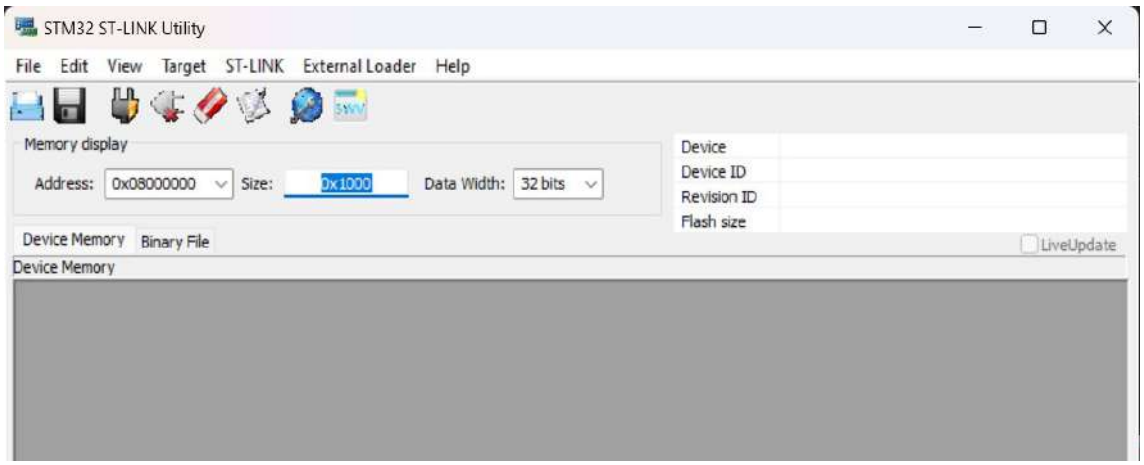


Figura 2.9: STM32 ST-Link Utility.

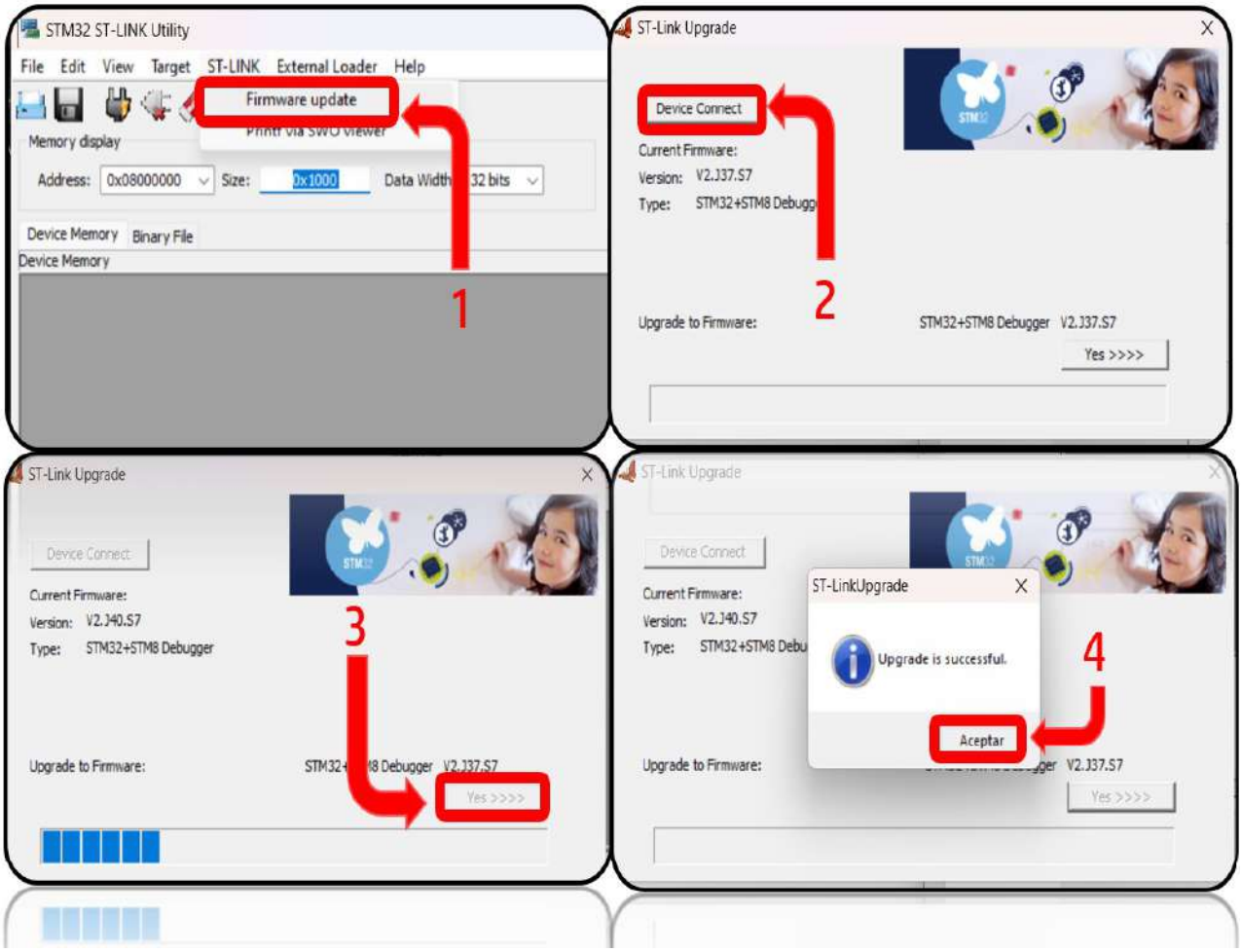


Figura 2.10: Actualización Firmware ST-Link v2

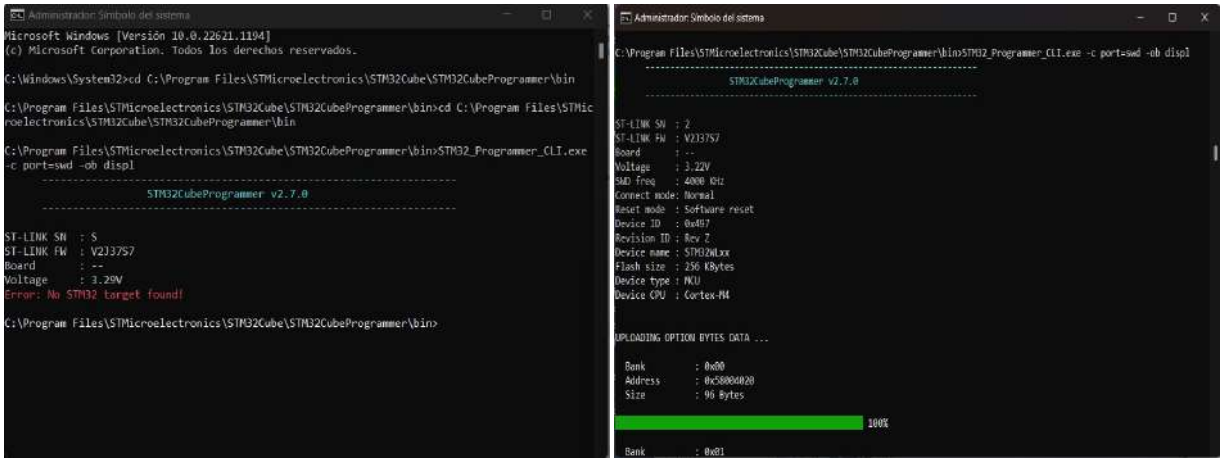
2.2.1 Reemplazo Firmware AT con un Firmware Basado en Arduino

En esta subsección se mostrará como se realizó el reemplazo del firmware original para el Nodo-E5 (comandos AT), utilizando el programa ArduinoIDE, es importante realizar el reemplazo de memoria con ayuda de el programador ST-Link v2 el cual permitirá cargar el archivo. Después de la actualización para el modulo ST-Link V2, se debe deshabilitar la protección de lectura. Para ello se conecta al puerto USB del ordenador los módulos ST-Link v2 y el Nodo LoRa-E5, se abre con permisos de administrador una ventana del CMD de Windows, el cual permitirá mediante comandos retirar la protección de lectura (RDP). En el CMD de windows se debe mover a la ruta por defecto de STM32cubeprogrammer, tal como se muestra en la figura 2.11, mediante el comando: `cd C:\Program Files\STMicroelectronics-`

\STM32Cube\STM32CubeProgrammer\bin.

Figura 2.11: Reemplazo Firmware Comando CMD.

Una vez en la ruta definida, se debe colocar el comando: `STM32_Programmer_CLI.exe -c port=swd -ob displ`, que muestra la información del programador que se observa en la figura 2.12b, y se observa características importantes como nombre del dispositivo, el modo de reinicio entre otras. Por otro lado, si se tiene un error como el de la figura 2.12a, se recomienda abrir el software STM32 ST-Link Utility, y la pestaña de opciones seleccionar *Target* y la opción *settings* y en la nueva ventana cambiar la opción de Mode ***Normal*** por ***Connect Under Reset***, luego desconectar el programador y al conectar al ordenador se debe presionar el botón de **reset** del nodo LoRa-E5 y seguido hacer clic en el botón ***Connect to the target*** del programa.

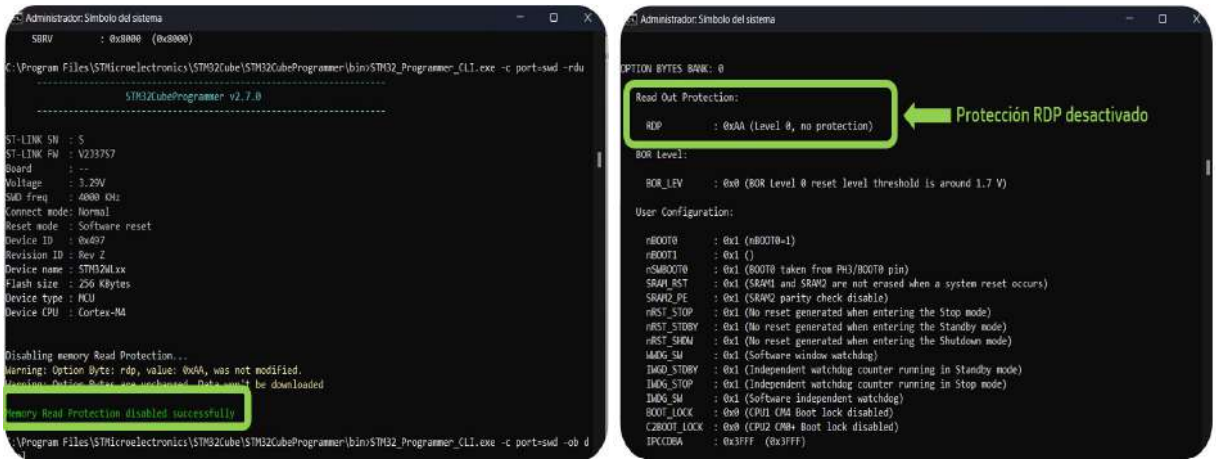


(a) Error de Lectura de ST-Link v2. **(b)** Lectura de ST-Link v2.

Figura 2.12: Ejecución de comandos mediante CMD para lectura de ST-Link V2 mediante STM32cubeprogrammer.

Después, se coloca el comando `STM32_Programmer_CLI.exe -c port=swd -rdu`, el cual permite eliminar la protección de lectura del módulo STLINK v2, esto se puede observar en la figura 2.13a, donde se muestra un mensaje en letras verdes de *"Memory Read Protec-*

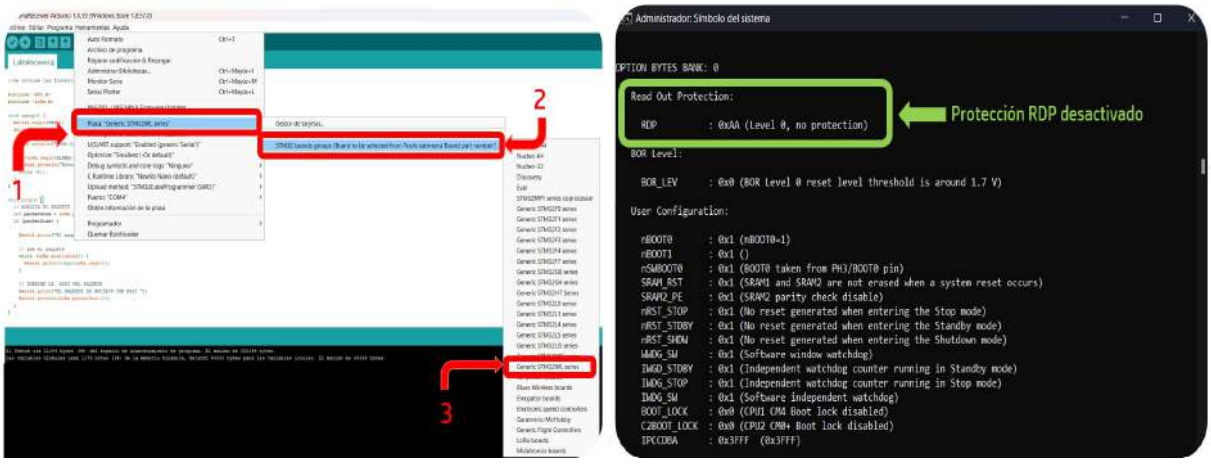
tion disabled successfully", y para comprobar se ejecuta el comando `-ob displ` donde en información de Read Out Proteccion se tiene el resultado en la figura 2.13b.



(a) Ejecución del Comando -rdu. **(b)** Comprobación de retiro de permiso RDP.

Figura 2.13: Retiro de permisos RDP mediante STM32cubeprogrammer.

Finalmente, para realizar el reemplazo del firmware del nodo LoRa-E5, se debe instalar las librerías de Lora para el software Arduino IDE y a su vez las librerías para los módulos STM32. Donde para cargar el programa de prueba antes se debe seleccionar las siguientes opciones en Arduino IDE, mostrada en la figura 2.14a, Herramientas/Placa/Gestor de placas, aquí se busca STM32 boards y se seleccionada ***Generic STM32WL series***, en el método se selecciona *STM32CubeProgrammer* y en la opción de *Board part number* se hace clic en la opcion que muestra la figura 2.14b.



(a) Opciones para seleccionar la Placa STM32. **(b)** Selección del módulo LoRA-E5.

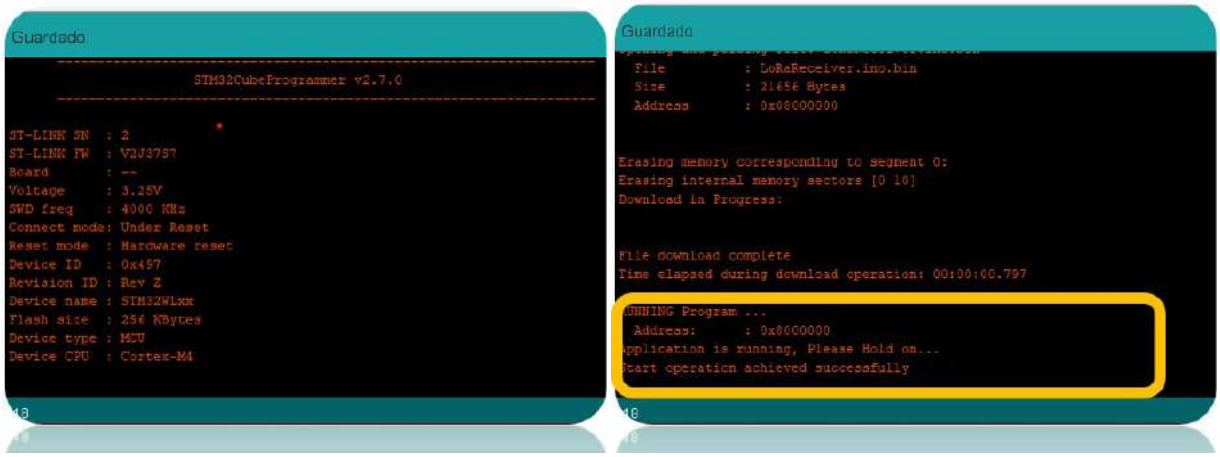
Figura 2.14: Selección del board WLE5JCIx.

Una vez realizado los cambios en Arduino IDE, se carga el programa que se muestra a

continuación, donde si se configuro correctamente el programador ST-Link V2, se lee de forma automática mediante el programa STM32cubeprogrammer mostrando las características del programador, como se muestra en la figura 2.15a y al subir el programa se deberá obtener un resultado de ***Start operation achieved successfully*** que se puede observar en la figura 2.15b.

```
1
- //se incluye las librerias para utilizar e nodo Lora
-
- #include <SPI.h>
5 #include <LoRa.h>
-
- void setup() {
-   Serial.begin(9600);
-   while (!Serial);
10
-   Serial.println("Nodo LoRa RX ");
-
-   if (!LoRa.begin(915E6)) {
-     Serial.println("Error al iniciar comunicacion LoRa");
15     while (1);
-   }
- }
-
- void loop() {
20   // ANALIZA EL PAQUETE
-   int packetSize = LoRa.parsePacket();
-   if (packetSize) {
-     Serial.print("El paquete ha sido recibido '");
-     // lee el paquete
25     while (LoRa.available()) {
-       Serial.print((char)LoRa.read());
-     }
-   // IMPRIME LA  RSSI DEL PAQUETE
-     Serial.print("EL PAQUETE SE RECIBIO CON RSSI ");
30     Serial.println(LoRa.packetRssi());
-   }
- }
```

Código 2.1: Código prueba para el cambio de Firmware (recibo de mensajes).



(a) Lectura de ST-Link mediante ArduinoIDE. **(b)** Reemplazo correcto de firmware nodo LoRa-E5.

Figura 2.15: Reemplazo de Firmware mediante Arduino IDE y STM32cubeprogrammer.

2.2.2 Configuración de Nodo Lora-E5 para la conexión con TTN

Para la configuración de los nodos Lora-E5 se implemento dos métodos distintos: el primero el cual se realizó mediante el reemplazo del firmware original del Wio-E5 colocando un firmware que permita mediante los software STM32 la programación mediante STM232Cube32 de este modo se puede crear un código en C para configurar parámetros del nodo, el segundo sera detallado a continuación el cual se centra en la conexión a la red TTN mediante comandos AT por lo cual placa de desarrollo Wio-E5 se conectara al ordenador por USB y se controla mediante los distintos comandos AT,

Es importante mencionar que para la conexión del nodo mediante comandos AT a TTN se tenga en cuenta una lista de comandos básicos los cuales nos dan características de la placa LoRa-E5. A continuación se muestran en la tabla 2.6 algunos comandos y en la figura 2.16 se observa la información de la placa obtenida:

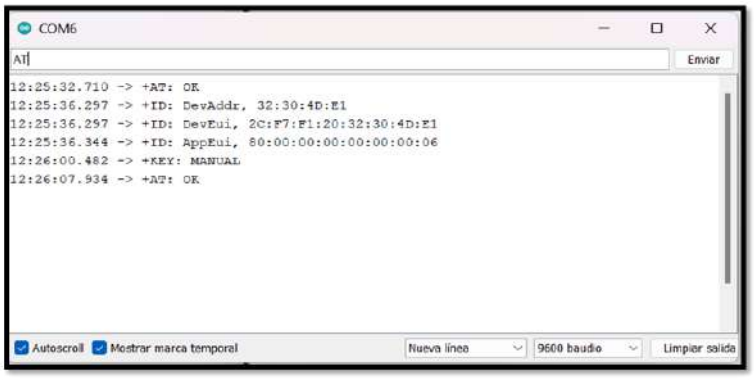


Figura 2.16: Obtención de ID placa LoRa-E5.

Tabla 2.6: Comandos AT Básicos [31].

	Descripción	Comando	Retorno
AT	Comprobación del modulo.	AT	AT:OK
ID	Muestra el ID del modulo LoRa. También permite cambiar el ID.	AT+ID	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=US915	DR: US91
DR	Establece la tasa de datos de LoRa.	AT+DR=DR0	DR: DR0 DR: US915 DR0 SF10BW125K
MODE	Elige el modo de operacion del nodo: LWABP, LWOTAA y TEST.	AT+MODE=LWOTAA:	MODE: LWOTAA
MSGHEX	Transmite una trama en formato Hexadecimal	AT+MSGHEX= "Mensaje en hexadecimal"	MSGHEX: Start MSGHEX: Done
JOIN	Funciona solo con el modo OTAA activo, permite conectarse a una red privada.	AT+JOIN	JOIN: Start JOIN: NORMAL JOIN: Network joined JOIN: NetID 000013 DevAddr 26:0C:CB:0D JOIN: Done

Para ello es importante configurar previamente la aplicación web TTN (The Thing of Networks).

2.2.2.1 Configuración de TTN (The Thing Networks)

Previamente se debe realizar los siguientes pasos para la conexion de un nodo LoRa=E5 [32]:

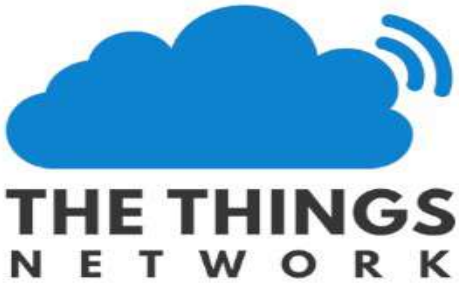


Figura 2.17: TTN (THE THING OF NETWOTKS).

1. Obtener las credenciales de TTN: Lo primero que se debe hacer es crear una cuenta en TTN en la página https://www.thethingsnetwork.org/ y registrar una nueva aplicación como se muestra en la figura 2.18. Se debe colocar el APP ID (nombre con el cual

se va a reconocer la APP) y la una debe descripción de la nueva aplicación creada.

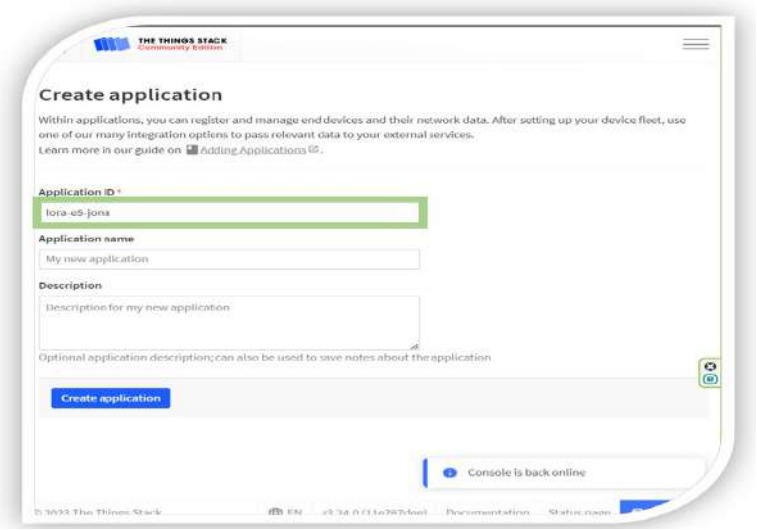


Figura 2.18: Creación de la Nueva APP en TTN

2 En la figura 2.19 se observa como se debe registrar un ***END DEVICE*** el cual será la placa LoRa-E5. Para ello se debe Configurar la región: El nodo LoRa-E5 soporta diferentes regiones, por lo que se debe configurar la región correspondiente a la ubicación geográfica donde se encuentra el nodo, en este caso **Unit States 902-920 MHz, FSB 2(USED BY TTN)**. Después se debe obtener el **DevEUI, AppEUI y APPKey del nodo LoRa-E5**, el principal es **JoinEUI** (identificador único de cada dispositivo).

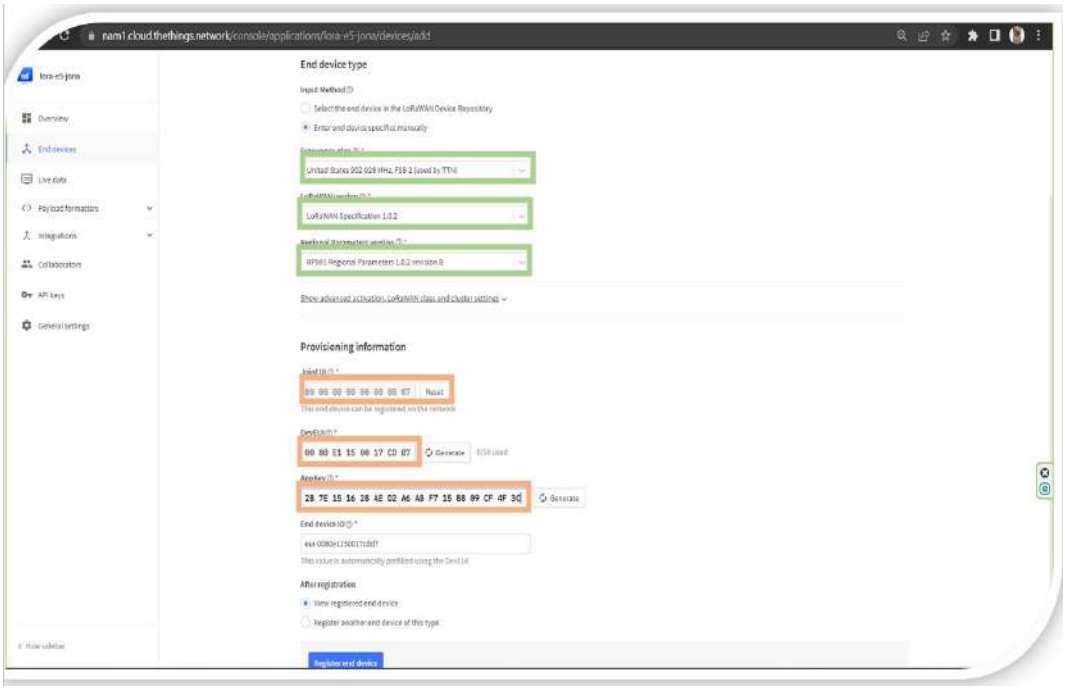


Figura 2.19: Creación de un END DEVICE en TTN

3. Es importante configurar el modo de funcionamiento para la llegada de mensajes al gateway: El nodo LoRa-E5 tiene diferentes modos de funcionamiento, pero el predeterminado y el que se va a utilizar en este caso e es un decodificador Json Parser.

4. En la figura 2.20 se observa como configurar la llegada de mensaje a TTN: Se debe configurar el nodo LoRa-E5 para que se conecte a TTN utilizando la siguiente configuración en el apartado **Payload** opción **Payload Formatters** y Clic en Uplink, donde en el apartado **Formatter type** se selecciona ***Custom Javascript formatter***.

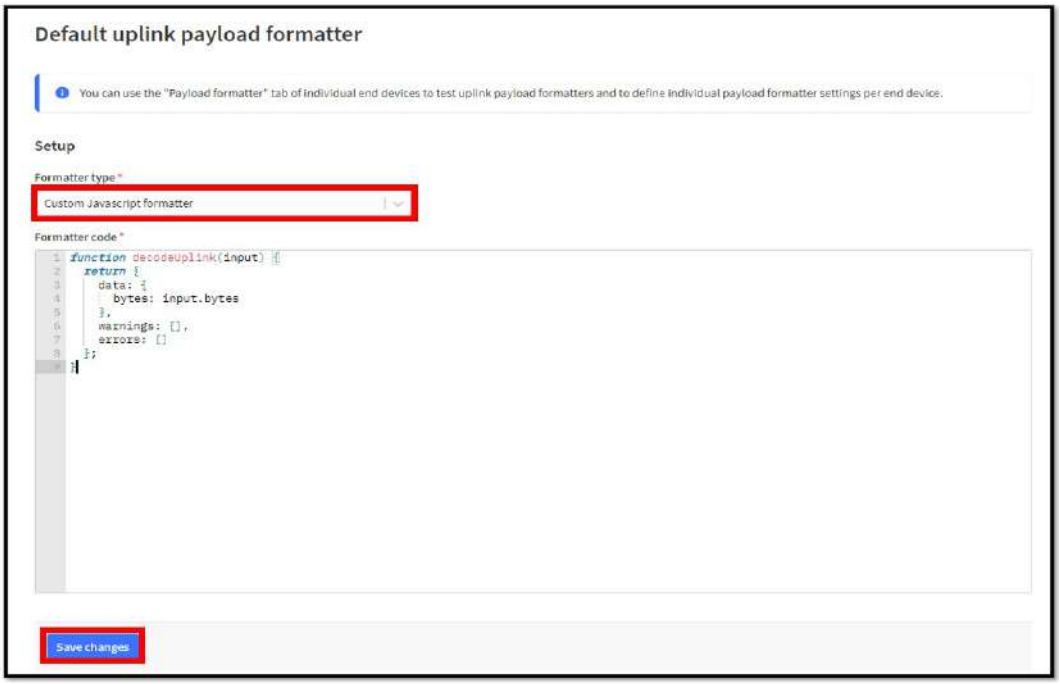


Figura 2.20: Configuración para llegada de mensajes en TTN

```
1 //codigo llegada de mensajes TTN
- function Decoder(bytes, port) {
-   var deco = {};
- if (port==8){
5  return {
-       Values: JSON.parse(String.fromCharCode.apply(null, bytes)) //Json
      parser
-   };
-   }
-   return deco;
10 }
```

Código 2.2: Código para el arribo de mensajes en TTN. UPLINK

2.2.2.2 Configuración Wio-E5 para conexión a TTN mediante STM32

Esta configuración permite conectar nuestra placa Wio-E5 con TTN a través de una conexión LoRaWAN, lo que les permite enviar datos a través de la infraestructura de puertas de enlace de la red. Donde en el siguiente enlace https://github.com/Seeed-Studio/LoRaWan-E5-Node/tree/qian se puede descargar un archivo ZIP, que permite configurar los datos de accesos de nuestra placa como se muestra a continuación:

1. Una vez descargado el archivo .ZIP del repositorio se extrae el archivo y con el software STM32CubeIDE se abre el proyecto en la Ruta `\LoRaWan-E5-Node-qian\LoRaWan-E5-Node-qian\Projects\Applications\LoRaWAN\LoRaWAN_End_Node\STM32CubeIDE` y se abre el archivo **.project**, una vez abierto se realiza la siguiente configuración como se muestra en la figura 2.21.

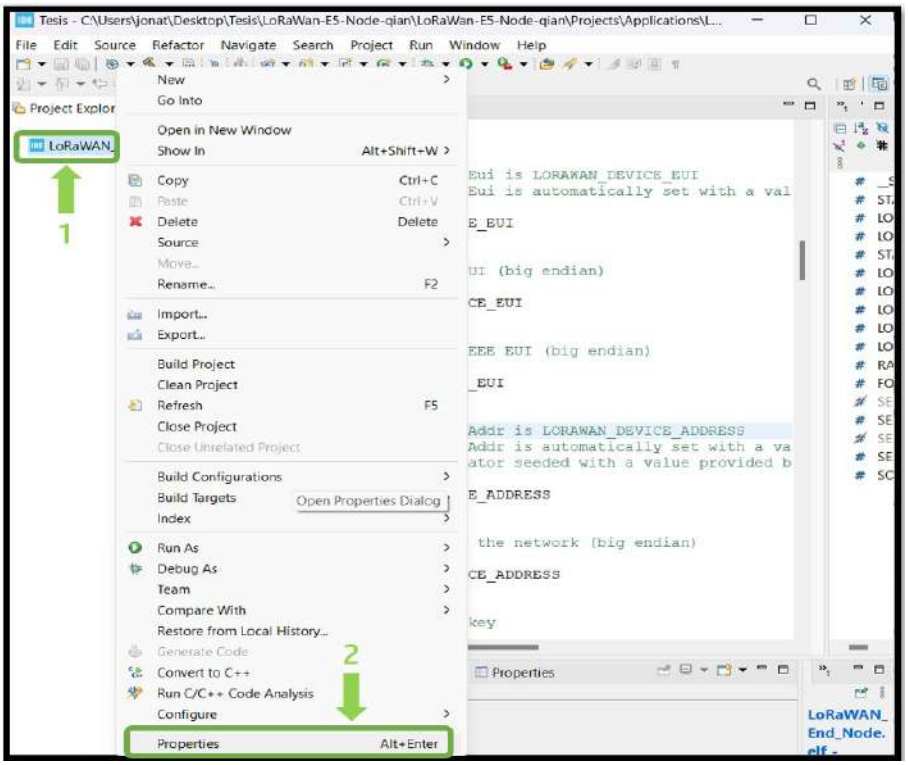


Figura 2.21: Configuración archivo .project

2. En la figura 2.22, se debe Navegar hacia la opción `C/C++ Build > Settings > MCU Post build outputs`, donde se debe seleccionar las opciones de ***Convert to binary file (-O BINARY)*** y ***Convert to Intel Hex(-O BINARY)***, para obtener el archivo .hex el que sera cargado al programador STLink v2.

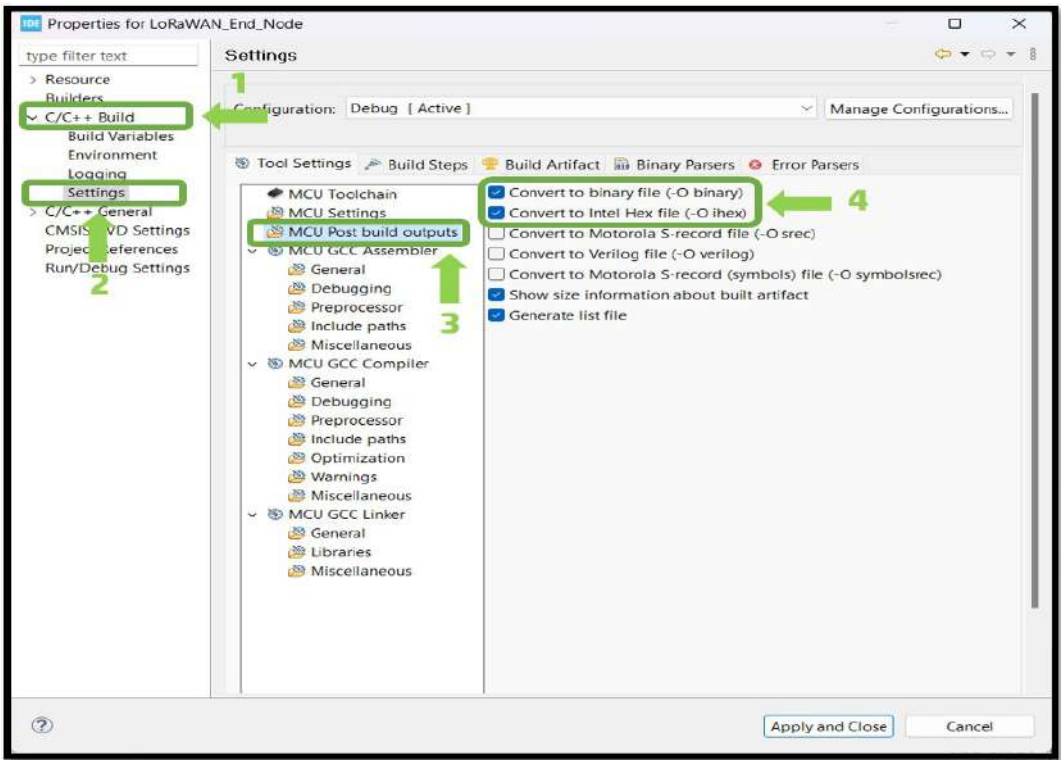
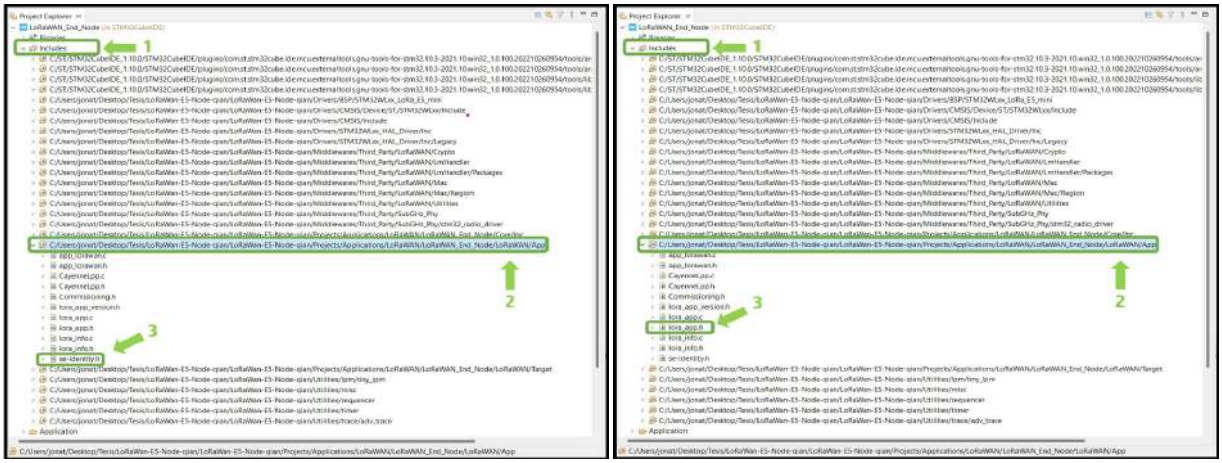


Figura 2.22: Configuración archivo -O hex

3. Seguido se debe compilar el programa, esto se realiza en la opción **Build and Debug** para ver que el archivo cargado no tiene ningún error. Después se debe abrir el archivo **se-identify.h** y el archivo **lora_app.h**. En el primer archivo se debe configurar los parámetros de la placa: Dispositivo EUI (DevEUI), Aplicación EUI (AppEUI), Aplicación CLAVE (APP_KEY), los que después serán registrados en la TTN, para ello se deberá agregar como se explico previamente un nuevo END DEVICE con los parámetros mencionados. En el segundo archivo se debe configurar la Región LoRaWAN [33].



(a) Archivo se-identify.h configuraciones **(b)** Archivo lora_app.h configuraciones.

Figura 2.23: Configuración de archivos LoRa.

La configuración de los archivos se pude observar en el capitulo 5 Anexo 1, A continuación se presenta una parte de los cambios realizados en los archivos mencionados:

```
/*!
 * Device address on the network (big endian)
 */
#define LORAWAN_DEVICE_ADDRESS                          ( uint32_t )0
    x0100000A

/*!
 * Application root key
 */
#define LORAWAN_APP_KEY                                 2B,7E,15,16,2
    8,AE,D2,A6,AB,F7,15,88,09,CF,4F,3C

/*!
 * Network root key
 */
#define LORAWAN_NWK_KEY                                 2B,7E,15,16,2
    8,AE,D2,A6,AB,F7,15,88,09,CF,4F,3C

/*!
 * Forwarding Network session key
 */
#define LORAWAN_NWK_S_KEY                               2B,7E,15,16,2
    8,AE,D2,A6,AB,F7,15,88,09,CF,4F,3C

/*!
 * Application session key
 */
#define LORAWAN_APP_S_KEY                               2B,7E,15,16,2
    8,AE,D2,A6,AB,F7,15,88,09,CF,4F,3C
```

Código 2.3: Código para configurar los parámetros ID de la placa Wio-E5

```
/* LoraWAN application configuration (Mw is configured by lorawan_conf.h
    ) */
#define ACTIVE_REGION                                   LORAMAC_REGION_US915

/*!
 * CAYENNE_LPP is myDevices Application server.
```

```
 *  */
/*#define CAYENNE_LPP*/

/*!
 * Defines the application data transmission duty cycle. 10s, value in [
     ms].
 */
#define APP_TX_DUTYCYCLE                            30000
```

Código 2.4: Código para configurar los parámetros de Frecuencia de la placa Wio-E5

4. Una vez realizada las modificaciones se debe volver a compilar el archivo para así general los archivos .hex. Finalmente se debe subir el firmware mediante el programador STLink V2, para ello se debe abrir la herramientas de windows CMD con permisos de administrador, y colocar los comandos para acceder a STM32CubeProgrammer y conectar el programador para obtener la ventana que se muestra en la figura 2.24,

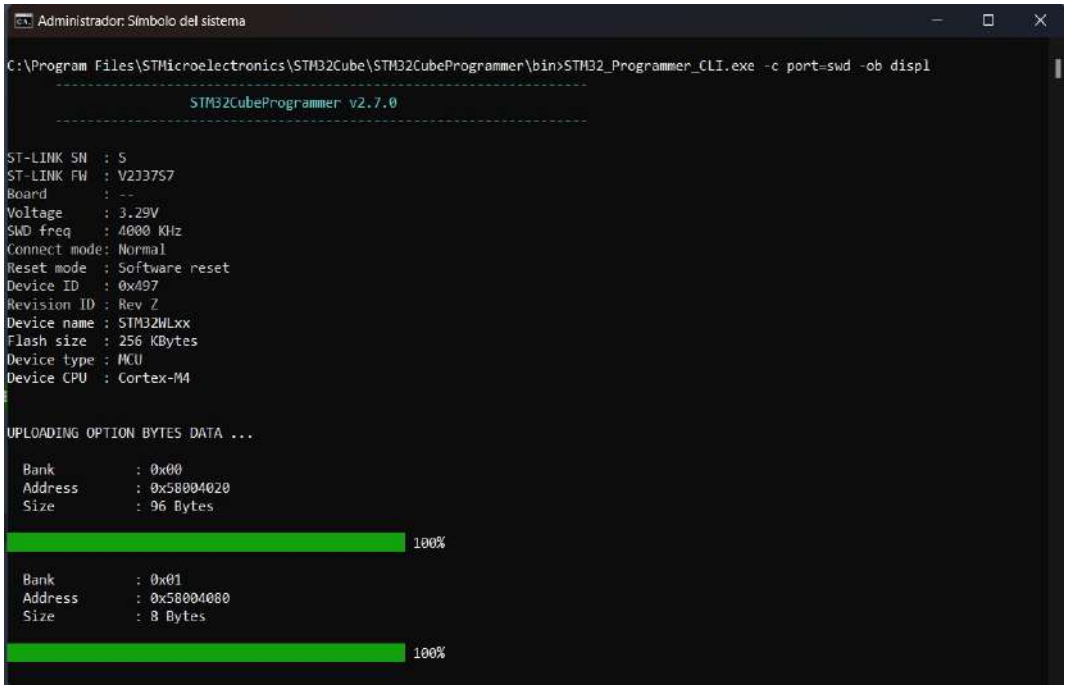


Figura 2.24: STM32CubeIDE mediante CMD

5. Finalmente, se debe colocar el siguiente código `STM32_Programmer_CLI.exe -c port =swd -d` seguido de la ubicación del archivo .hex en este caso es: `C:\Users\jonat\Des ktop\Tesis\LoRaWan-E5-Node-qian\LoRaWan-E5-Node-qian\Projects\Application s\LoRaWAN\LoRaWAN_End_Node\STM32CubeIDE\Debug\LoRaWAN_End_Node.hex`, mostrando el resultado de la figura 2.25, y un mensaje de **FILE DOWNLOAD COMPLETE.**

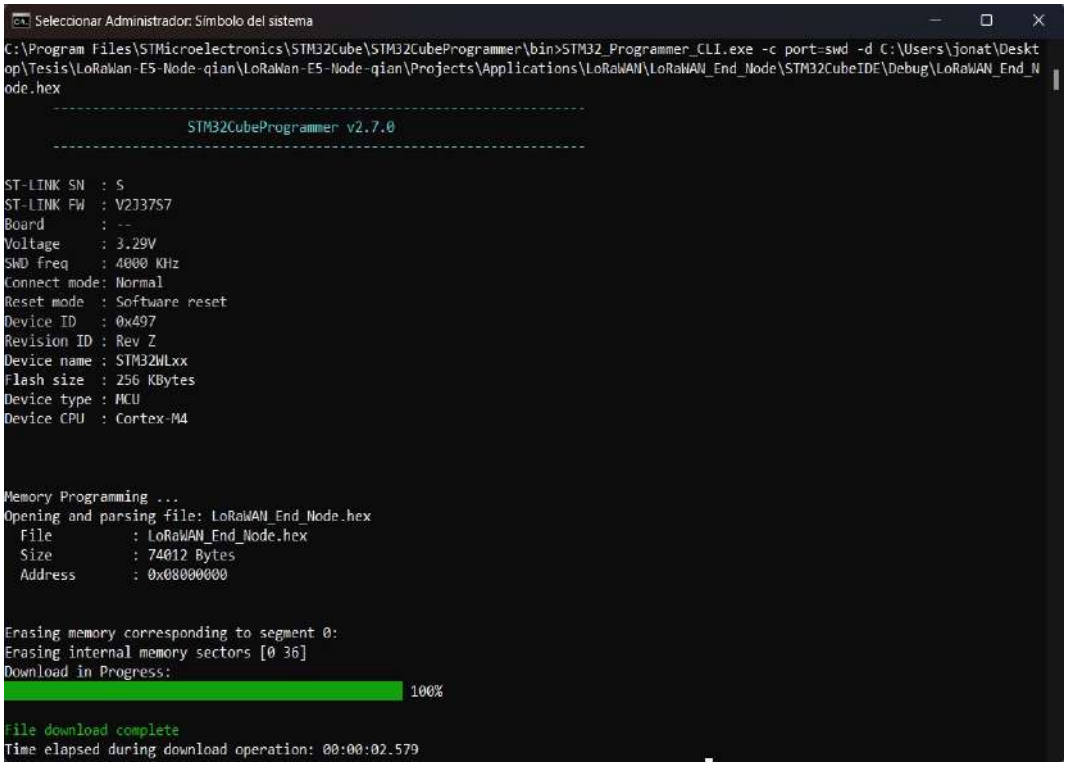


Figura 2.25: Firmware para conexión Wio-E5 con TTN mediante STM32CubeProgrammer.

7. Para observar la conexión se puede realizar mediante el serial de Arduino ID, seleccionar el puerto y establecer la velocidad de comunicación en 115200 baudios, dando como resultado lo que se observa en la figura 2.26.

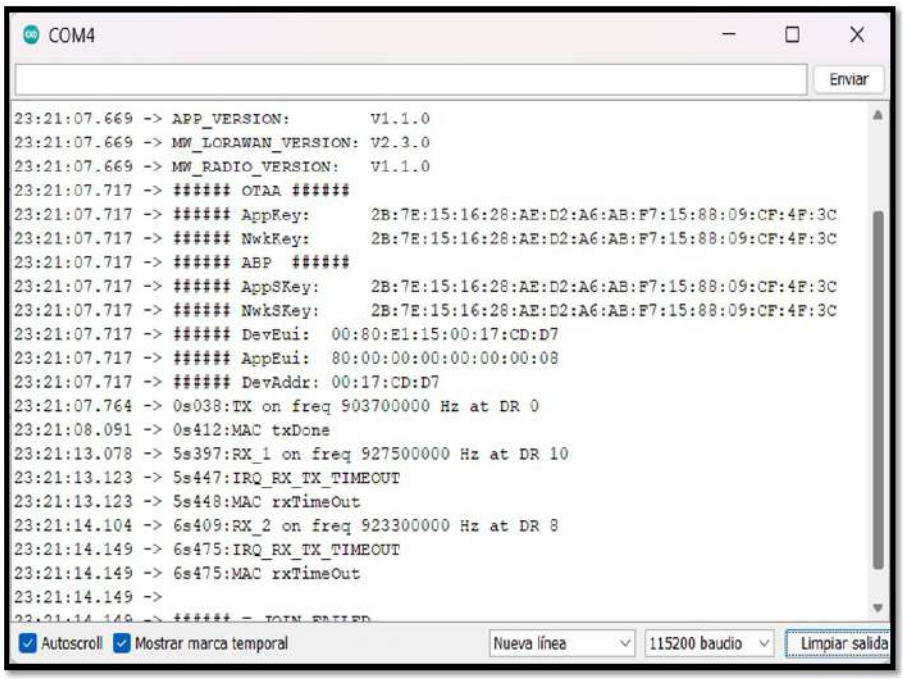


Figura 2.26: Cambio ID, lectura mediante Arduino.

8. La figura 2.27, muestra la conexión a la TTN (The Things of Networks), mediante OTAA, (Over-The-Air-Activation), donde en un recuadro color verde se observa **MCPS Indication**, que se utiliza para iniciar la transmisión de un mensaje de datos desde un dispositivo final. Seguido, en un recuadroceleste se observar **MCPS Confirm** para confirmar la recepción de un mensaje de datos enviado por el dispositivo final (end-device) al servidor de red.

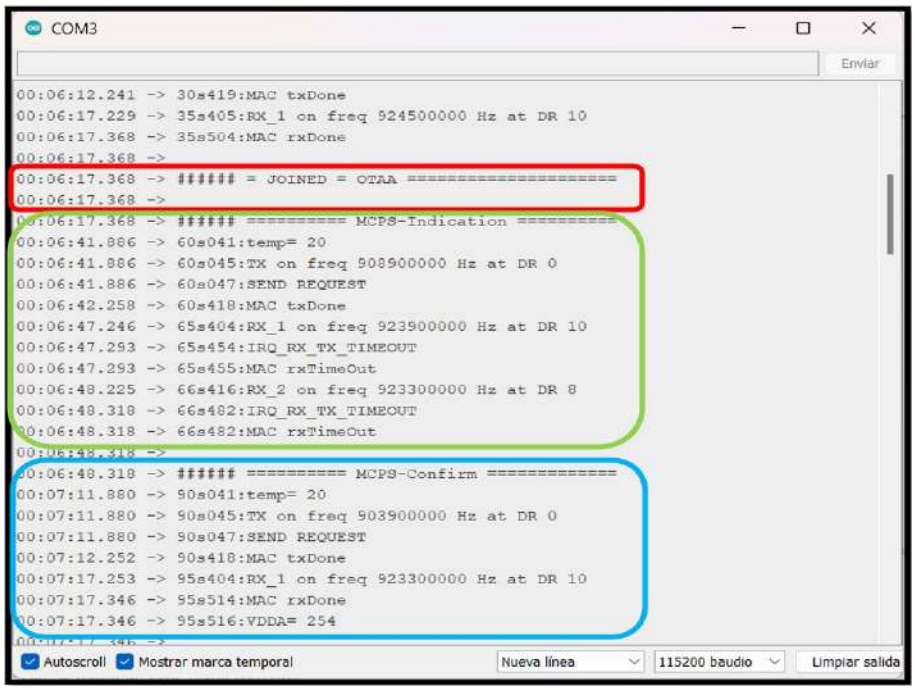


Figura 2.27: Conexión a la TTN mediante OTAA.

En la figura 2.28, se observa el Live Data de la aplicación creada en la TTN, mostrando el registro y el payload enviado desde el End Device.

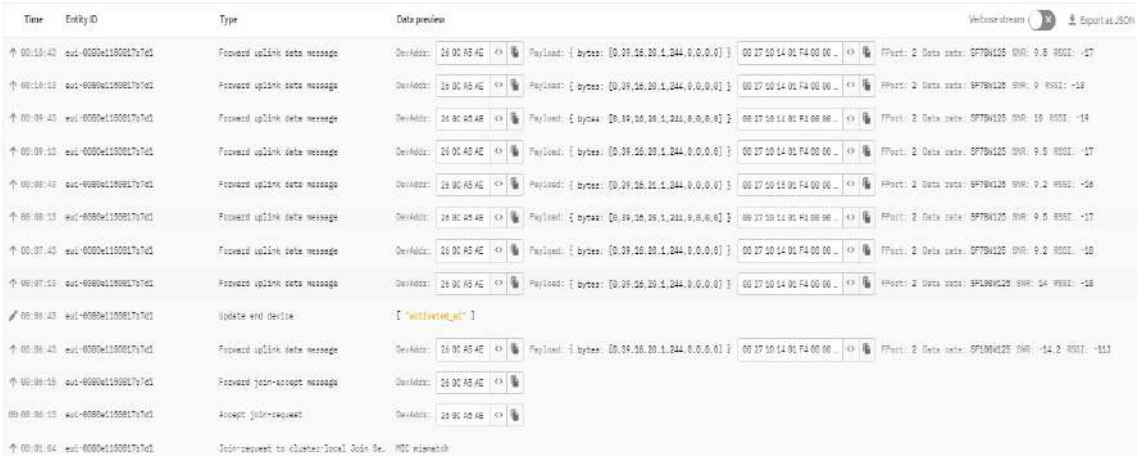


Figura 2.28: a la TTN mediante OTAA.

2.2.3 Diseño del diagrama de flujo para la medición de retardos en al retransmisión.

Los nodos Wio E5 son capaces de retransmitir (relay) mensajes de otros nodos en la red. Esta función de retransmisión es importante para mejorar el alcance y la confiabilidad de la red LoRaWAN, especialmente en áreas donde la señal inalámbrica puede ser débil o intermitente. Cuando un nodo envía un mensaje a través de la red LoRaWAN, la estación base recibe el mensaje y lo envía al servidor de red. Si el servidor de red determina que el mensaje no se recibió correctamente, puede enviar una solicitud de retransmisión (retransmission request) a los nodos cercanos para que vuelvan a enviar el mensaje. La figura 2.29, se muestra la Red LoRaWAN implementada, mostrando: End devices (Conexión Arduino Nano- LoRa E5), Gateway (RG191), Network Server (TTN) y Dashboards (Serial Arduino).

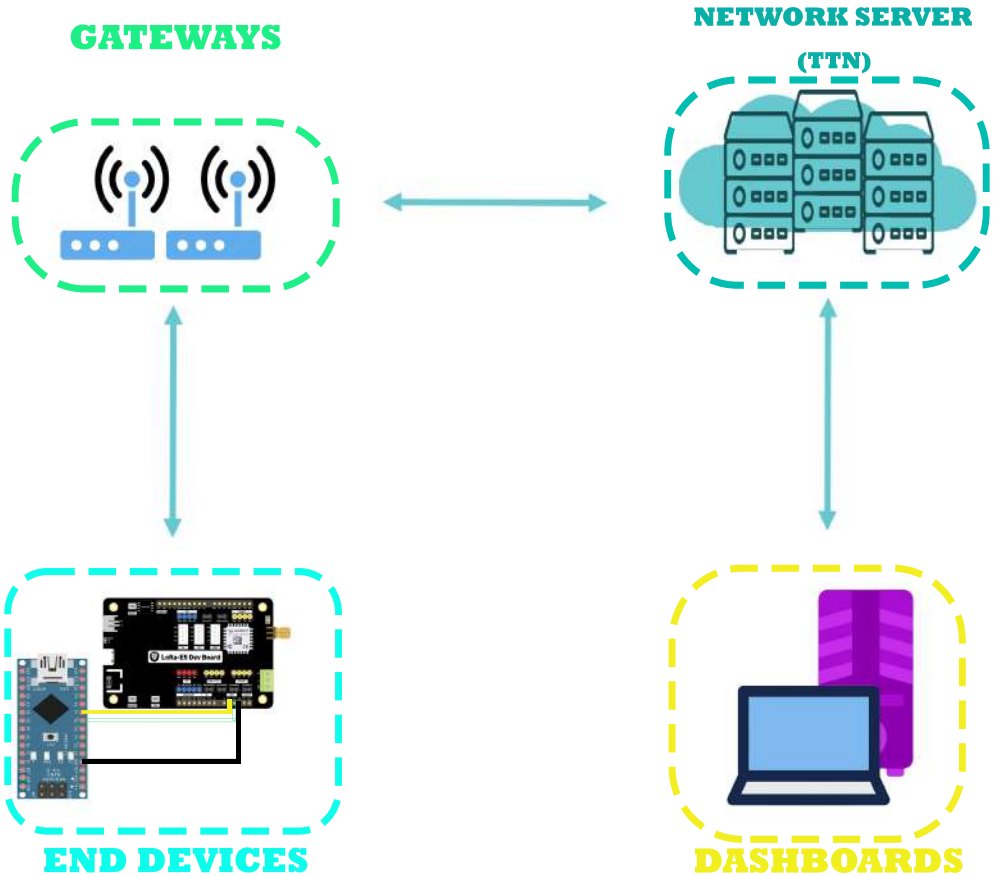


Figura 2.29: Prototipo de la Red LoRaWAN

Los nodos Wio E5 pueden recibir estas solicitudes de retransmisión y reenviar el mensaje original. Para hacerlo, los nodos Wio E5 deben estar configurados para operar en modo

de retransmisión y estar ubicados lo suficientemente cerca del nodo original como para recibir el mensaje. La figura 2.30 es el diagrama de flujo implementado para el la medición de retardos en retransmisión de una red LoRaWAN, donde básicamente se inicializa la Biblioteca LoRaWAN en Arduino IDE, posteriormente se configura los parámetros del nodo para la retransmisión y establecer un tiempo para el envió de mensajes, espera el envió y la conexión al Gateway con las credenciales respectivas, el servidor central enviará un mensaje de confirmación al nodo a través del gateway después de recibir el mensaje. El nodo esperará una respuesta de ACK durante un tiempo determinado y, si no recibe una respuesta, asumirá que el mensaje no se entregó correctamente y retransmitirá el mensaje.

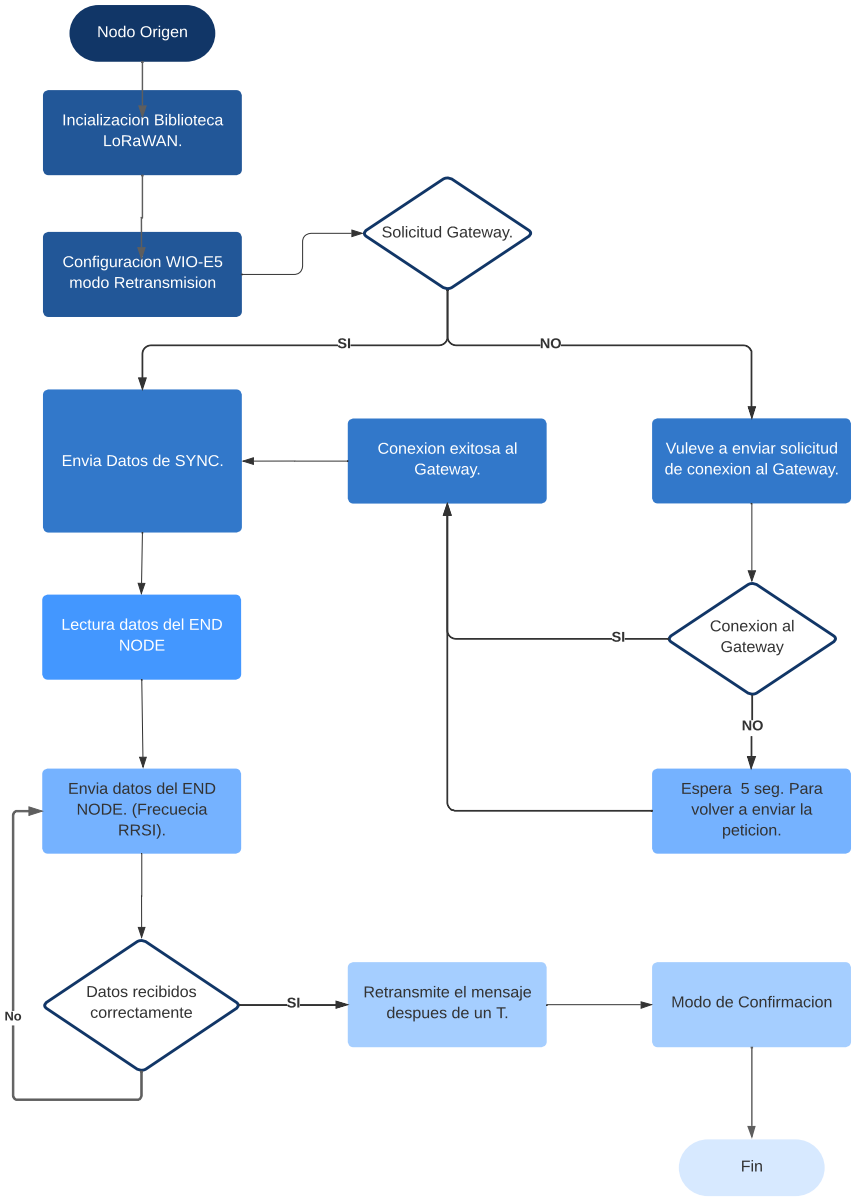


Figura 2.30: Diagrama de Flujo para la implementación del código

2.3 PROGRAMACIÓN DE LOS NODOS PARA LA MEDICIÓN DE RETARDOS EN LA RED LORAWAN.

2.3.1 Arduino IDE y Librerías para LoRaWAN

Partiendo del diagrama de flujo observado en la anterior sección, mediante el programa Arduino IDE y las diferentes librerías para LoRaWAN y sus microprocesadores embebidos como **LoRa.h, lmic.h**,se creara la codificación para la comunicación en la red LoRaWAN, donde la librería LoRa.h proporciona una interfaz fácil de usar para configurar y utilizar los módulos LoRa y enviar y recibir datos de forma inalámbrica. También proporciona una serie de funciones para establecer la frecuencia de transmisión, la potencia de salida, el ancho de banda, el factor de propagación, el CRC y otros parámetros de configuración.

LMIC se puede usar en una amplia variedad de microcontroladores, como Arduino, ESP32, STM32, entre otros, y puede ser adaptado a diferentes radios LoRa. La librería se ha convertido en una herramienta popular para desarrollar dispositivos IoT de baja potencia que se conectan a la red LoRaWAN. Por otro lado, la librería **LMIC** sirve para enviar datos de sensores a través de The Things Network o una red de datos similar basada en LoRaWAN. Donde para implementar se necesita de las siguientes características.

- Un procesador de 32 bits (ARM, XTENSA, etc).
- Una radio LoRa basada en SX1276.
- Un entorno de tiempo de ejecución de Arduino [34].

2.3.2 Codificación del diagrama de flujo

La medición del retardo en transmisión LoRaWAN, se empleara de acuerdo a dos métodos de confirmación el iACK y eACK teniendo dos códigos diferentes para cada uno de estos. A continuación se presentara una breve explicación de los lineas de código mas importantes:

❒ En general se debe configurar las variables y se incluyen las bibliotecas que se utilizarán en el código. En particular, se incluyen la biblioteca LMIC, que proporciona las funciones para el manejo de la comunicación LoRaWAN.

```
//CODIGO PARA CALCULAR EL RETARDO EN UNA RED LORAWAN
//MEDIANTE LAS SIGUIENTES LIBRERIAS.
#include <Arduino.h>
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
```

Código 2.5: Código para inicialización definición de librerías.

❒ Se definen las credenciales de la red LoRaWAN a la que se conectará el nodo. Estas credenciales incluyen la dirección única del dispositivo (DEVEUI), la dirección única de la aplicación (APPEUI) y la clave de aplicación (APPKEY

```

// Definir las credenciales de la red LoRaWAN configuradas mediante
    STM32CubeIDe
static const u1_t PROGMEM DEVEUI[8]={ 0x00, 0x80, 0xE1, 0x15, 0x00, 0x17
    , 0xCD, 0xD7 };
static const u1_t PROGMEM APPEUI[8]={ 0x80, 0x00, 0x00, 0x00, 0x00, 0x00
    , 0x00, 0x08 };
static const u1_t PROGMEM APPKEY[16]={0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE
    , 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F,0x3C };
```

Código 2.6: Código para inicialización de credenciales.

En esta sección se definen el objeto de transmisión y un buffer para los datos que se transmitirán.

```
// Definir el objeto de transmisi n
static osjob_t sendjob;
static uint8_t txBuffer[255];
static uint32_t start_time = 0;  // Variable para guardar el tiempo de
    env o del retardo
```

Código 2.7: Código para definir el objeto a transmitir.

Esta es la función que se encargará de enviar los datos al servidor LoRaWAN. En primer lugar, se define el tipo de mensaje como çonfirmed"(se espera ACK), y luego se llama a la función **LMIC_setTxData2** para enviar los datos al servidor. Finalmente, se muestra un mensaje en la consola serial indicando que el mensaje ha sido enviado en la Frecuencia definida.

```
// Guardar el tiempo actual en la variable start_time
start_time = millis();

// Preparar el mensaje a enviar
memcpy(txBuffer, "Hola, vengo desde End Node 1", 28); // mensaje a enviar
LMIC.frame[0] = 0x02;  // Tipo de mensaje confirmed (espera un iACK)
LMIC_setTxData2(1, txBuffer, sizeof(txBuffer), 1);  // Enviar el mensaje

// Imprimir el mensaje enviado y la frecuencia utilizada
Serial.print("Mensaje enviado: ");
for (int i = 0; i < sizeof(LMIC.frame); i++) {
  Serial.print(LMIC.frame[i], HEX);
  Serial.print(" ");
}
Serial.print("en la frecuencia ");
Serial.println(LMIC.freq);

// Esperar la respuesta de iACK durante 5 segundos
os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(5), do_send);


}
```

Código 2.8: Código para definir el objeto a transmitir.

❒ En función **do_send** se incluye un temporizador que cuenta el tiempo que transcurre desde el envío del mensaje hasta la recepción del iACK o eACK. En la variable **start_time** al inicio del código se usa mediante la función millis() para obtener el tiempo actual en milisegundos. Luego, dentro de la función **do_send**, antes de enviar el mensaje, se guarda valor actual de **millis()** en la variable definida. Después, cuando llega el iACK o eACK, se resta el valor actual de millis() con el valor de **start_time** para obtener el tiempo transcurrido. Finalmente, se imprimir el valor del tiempo transcurrido en el serial de arduino.

```
void onEvent (ev_t ev) {
  switch(ev) {
    case EV_TXCOMPLETE:
      Serial.println("Transmisi n completa");
```

```
5         break;
-       case EV_RXCOMPLETE:
-         Serial.print("Recepci n completa. RSSI=");
-         Serial.println(LMIC.rssi);
-         break;
10      case EV_JOINED:
-         Serial.println("Unido a la red LoRaWAN Jona-Tesis");
-         break;
-       case EV_ACK_COMPLETE:
-         Serial.println("iACK recibido despues de");
15        Serial.print(millis() - start_time);
-         Serial.println(" ms");
-         break;
-       default:
-         Serial.println("Evento desconocido");
20        break;
-     }
- }
```

Código 2.9: Código para definir el objeto a transmitir.

El código completo se encuentra detallado en el capitulo 5 Anexo 2.

Hay que detallar que para la evaluación de los eACKS el nodo de extremo espera explícitamente una respuesta de confirmación de entrega del gateway después de haber enviado un mensaje. Si el gateway recibe el mensaje correctamente, enviará una confirmación de entrega (ACK) al nodo de extremo.

2.3.3 Detalles del envió de paquetes para la medición de retardos

Como se sabe el retardo de retransmisión es el tiempo que el nodo espera antes de volver a intentar enviar un mensaje que no ha sido confirmado. Este tiempo de espera se puede ajustar mediante parámetros en la configuración del nodo. El retardo de retransmisión también puede estar influenciado por la política de retransmisión de la red LoRaWAN a la que está conectado el nodo.

En este caso, se establece en **DR_SF7**, lo que significa que se utiliza una tasa de datos de 7 (es decir, una tasa de transferencia de datos de 547 bits por segundo) y establece una tasa de datos de 125kHz con una codificación de convolución de tasa de 4/5 y un tiempo

en el aire aproximado de 1.22 segundos.

La potencia de transmisión (Txpow) es la potencia de salida de la señal de radio transmitida. En este caso, se establece en 14, lo que significa que se utiliza una potencia de transmisión de 14 dBm. Y por su parte en el codigo se establece en **os_setTimedCallback** un tiempo de respuesta para el ACK de 5 seg, este puede ser modificado según se desee tener una respuesta del gateway al nodo.

2.4 EVALUACIÓN DE RETARDOS Y COMPARACIÓN DE DATOS ENVIADOS Y RECIBIDOS.

El retardo en la retransmisión en una red LoRaWAN puede variar según varios factores, como la tasa de transmisión de datos (data rate), la potencia de transmisión, la distancia entre el dispositivo y el gateway, la congestión de la red, entre otros.

En general, el estándar LoRaWAN permite configurar el número máximo de retransmisiones y el tiempo de espera entre retransmisiones. Por ejemplo, en la especificación técnica de LoRaWAN 1.0.3, se establece que el valor predeterminado para el número máximo de retransmisiones es de 3 y que el tiempo de espera entre retransmisiones se duplica con cada retransmisión (es decir, el primer reenvío ocurre después de 1 segundo, el segundo reenvío después de 2 segundos, el tercero después de 4 segundos, etc.) [35].

La fórmula actualizada para calcular el retardo en la retransmisión en LoRaWAN 1.0.3 (sección 4.3.3) es la siguiente:

$$R_{Delay} = (T_{payload} + Tpreamble + Tsync + Theader) * 2^{(retry_{count}+1)} * (1 + Random(0, C)) \quad (2.1)$$

Donde:

- **Tpayload:** es el tiempo de duración del payload (datos) del paquete.
- **Tpreamble:** es el tiempo de duración del preámbulo del paquete.
- **Tsync** es el tiempo de duración de la sincronización del paquete.
- **Theader:** es el tiempo de duración del encabezado del paquete.
- **retry_count:** es el número de intentos de retransmisión.

- **C:** es el parámetro de tiempo de espera aleatorio exponencial (backoff) y está configurado por el servidor LoRaWAN.

- **Random(0, C):** es un número aleatorio entre 0 y C generado por el dispositivo.

La línea de codigo **"LMIC_setTxData2(1, txBuffer, sizeof(txBuffer), 1)"** se está configurando la transmisión del mensaje con un solo reenvío (retry = 1). Esto se indica mediante el valor 1 en el cuarto argumento de la función **LMIC_setTxData2()**. Por otro lado el valor de por defecto de **C** se establece en 2 en la biblioteca LMIC de Arduino, utilizando este para realizar el calculo del tiempo de retardo en la red LoRaWAN.

2.4.1 Comparativa de datos enviados y recibidos

Las figuras 2.31 y 2.32 muestran los datos enviados mediante el payload de origen hacia el payload de destino respectivamente, permitiendo calcular el valor aproximado del retardo producido en la retransmisión entre el END DEVICE (LoRa-E5) y el gateway, calculado por la función **millis()**. Y a su vez se puedo calcular EL RSSI, al momento de realizar la recepción del mensaje utilizando la función **LMIC.RSSI** de la librería mencionada, cabe mencionar que el obtener el resultado de estos valores aumentan el valor del consumo de energía de cada nodo.

```
// Guardar el tiempo actual en la variable start_time
start_time = millis();

// Preparar el mensaje a enviar
memcpy(txBuffer, "Hola, vengo desde End Node 1", 28); // mensaje a enviar

LMIC_setTxData2(1, txBuffer, sizeof(txBuffer), 1);  // Enviar el mensaje

// Imprimir el mensaje enviado y la frecuencia utilizada
Serial.print("Mensaje enviado: ");
for (int i = 0; i < sizeof(LMIC.frame); i++) {
  Serial.print(LMIC.frame[i], HEX);
  Serial.print(" ");
}
```

Figura 2.31: Mensaje enviado.

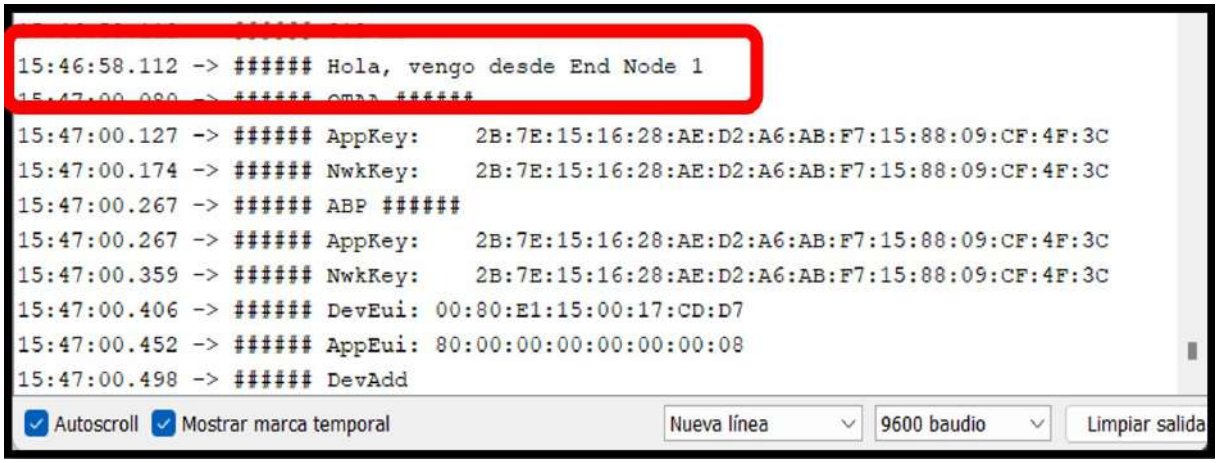


Figura 2.32: Mensaje Recibido.

La figura 2.35 permite observar con el puerto serial de ArduinoIDE el login realizado al Gateway mediante OTAA, al igual que el tiempo del retardo de la retransmisión,

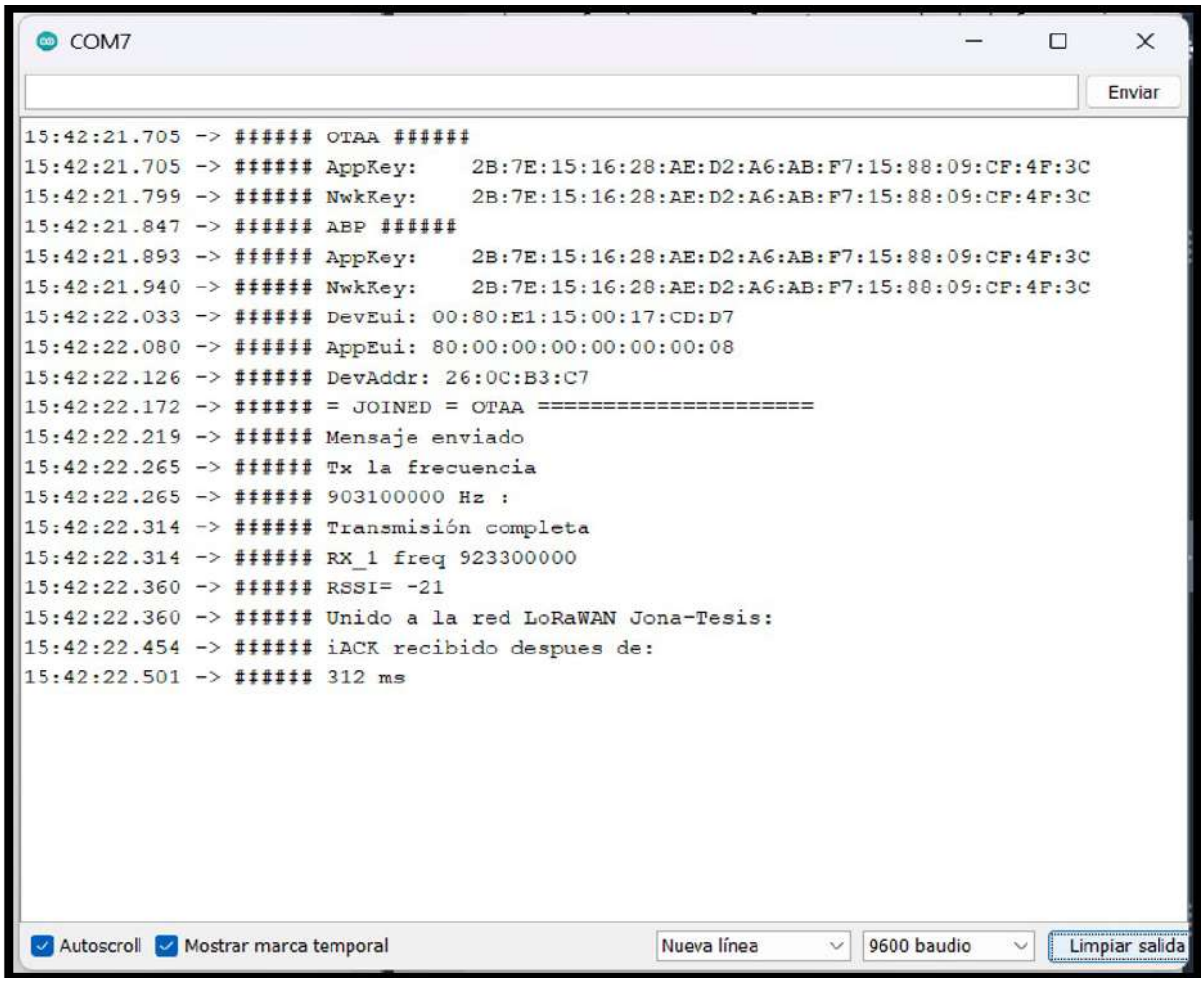


Figura 2.33: Valor obtenido de RSSI y tiempo estimado del Retardo mediante iACK.

En la figura 2.34 se puede observar el login establecido a nuestro Dashboard en la TTN mostrando valores de RSSI y a su vez detalles importantes como el valor de SNR, por otro lado el mensaje enviado se encuentra codificado debido a que si se tratase de un mensaje con información confidencial este puede ser capturado y ver lo que tiene.

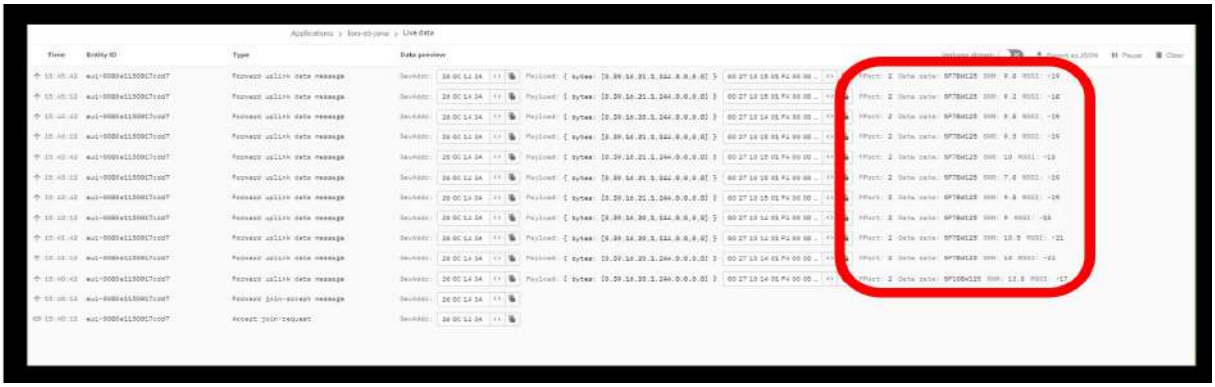


Figura 2.34: Valores obtenidos en la TTN (valoR RSSI y payload enviado)

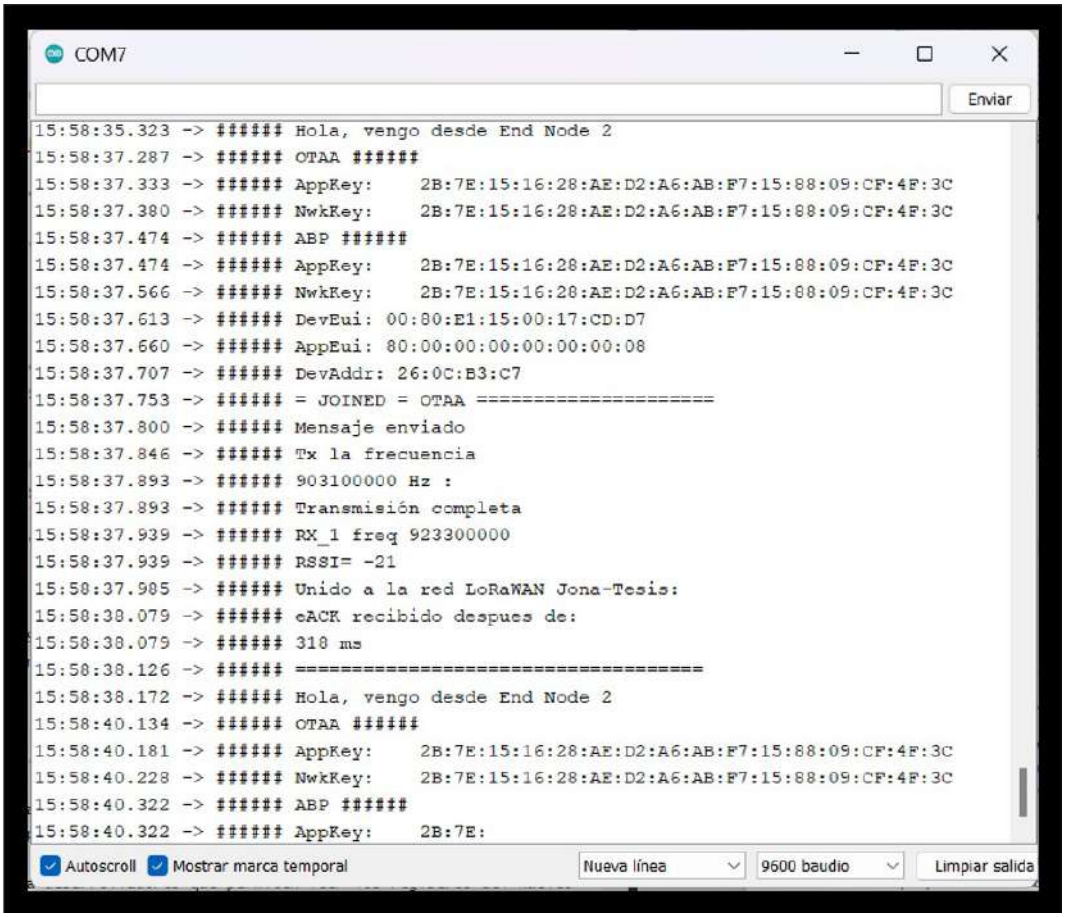


Figura 2.35: Valor obtenido de RSSI y tiempo estimado del Retardo mediante eACK.

2.4.1.1 Cálculo de retardo en la retransmisión.

Para calcular el tiempo de retardo se puede proponer un ejemplo en el cual se tiene un paquete de datos que tiene un tamaño de payload (datos) de 20 bytes, un preámbulo de 8 símbolos, una sincronización de 1 símbolo y un encabezado de 7 bytes. Además, se intentará enviar este paquete con un máximo de 3 retransmisiones y un valor de C de 4. Para calcular el retardo de retransmisión para cada intento, podemos utilizar la fórmula 2.1:

$$R_{Delay} = (T_{payload} + Tpreamble + Tsync + Theader) * 2^{(retry_{count}+1)} * (1 + Random(0, C))$$

Primero, se calcula el valor fijo del retardo de retransmisión para el primer intento, es decir cuando **retry_count = 0**

$$R_{Delay} = (36) * 2^{(0+1)} * (1 + Random(0, 4))$$

$$R_{Delay} = (36) * 2^{1} * (1 + Random(0, 4))$$

$$R_{Delay} = (36) * 2 * (1 + 1)$$

$$R_{Delay} = 144ms$$

Para el segundo intento (**retry_count = 1**, podemos calcular el retardo de retransmisión con la misma fórmula:

$$R_{Delay} = (36) * 2^{(1+1)} * (1 + Random(0, 4))$$

$$R_{Delay} = (36) * 2^{2} * (1 + 3)$$

$$R_{Delay} = (36) * 4 * (4)$$

$$R_{Delay} = 576ms$$

Finalmente, para el tercer intento **(retry_count = 2)**, podemos calcular el retardo de retransmisión:

$$R_{Delay} = (36) * 2^{(2+1)} * (1 + Random(0, 4))$$

$$R_{Delay} = (36) * 2^{3} * (1 + 2)$$

$$R_{Delay} = (36) * 8 * (3)$$

$$R_{Delay} = 864ms$$

De esta manera, se puede calcular el retardo de retransmisión para cada intento utilizando la fórmula mencionada anteriormente. Es importante tener en cuenta que el valor aleatorio generado por el dispositivo (Random) puede afectar el retardo de retransmisión y puede variar en cada intento de retransmisión.

Es importante recalcar que la La función **LMIC_setLinkCheckMode(0)** establece el modo de verificación de enlace de LoRaWAN en desactivado. La verificación de enlace es un mecanismo de control que el servidor de red utiliza para comprobar el estado y la capacidad de respuesta del dispositivo final.

Cuando está activado, el nodo LoRa envía un mensaje de solicitud de verificación de enlace **(LinkCheckReq)** al servidor de red y espera una respuesta **(LinkCheckAns)**. La respuesta incluye información sobre la calidad de la señal y la tasa de pérdida de paquetes.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En la presente sección se mostrará la implementación física realizada del prototipo, a su vez se presentará los resultados obtenidos en los diferentes escenarios variando la distancia entre el Gateway y el Nodo, estos valores seran comparados con los valores obtenidos mediante la formula mostrada en el capitulo anterior y el valor obtenido mediante el serial de Arduino IDE.

3.1 RESULTADOS

3.1.1 Implementación del Prototipo

La figura 3.1 muestra el diseño en 3D de la caja para el END DEVICE, realizada en Tinkercad la cual tiene un apartado para colocar las baterías (alimentación del nodo de 70 mm x 20 mm y a su vez un espacio que ajusta el nodo y evitar que al momento de colocar en algún ambiente lejano y exista movimiento permita que se mueva la placa, al igual una tapa que permite atornillarse junto con la carcasa y así evitar que exista filtraciones de polvo o agua.La dimensión final de la estructura es de 97mm x 100 mm.

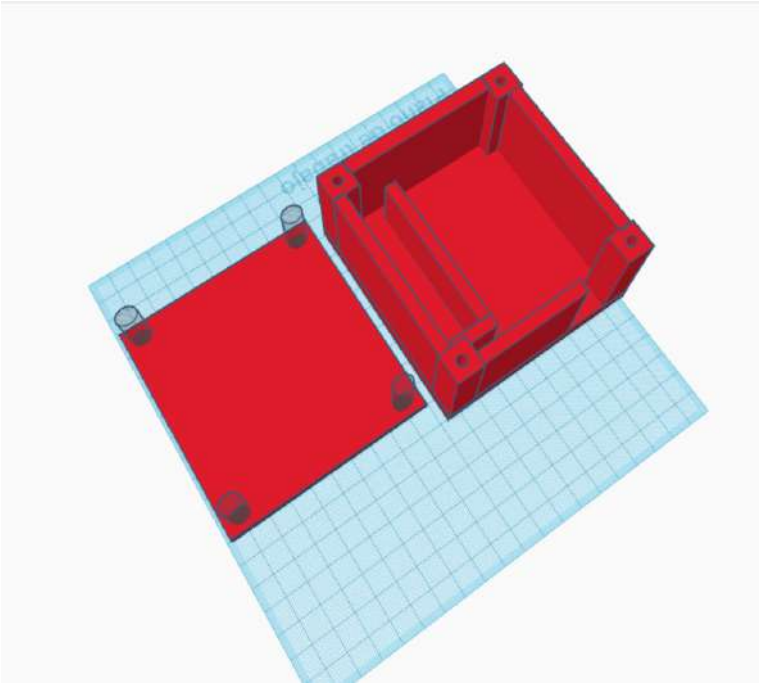


Figura 3.1: Diseño de la estructura de la caja para el End Device.

Al momento de realizar la impresión 3D de la caja, se coloca la placa dentro de la carcasa dando como resultado lo que se observa en la imagen 3.2, donde se tiene un espacio para desmontar y montar la antena con facilidad.

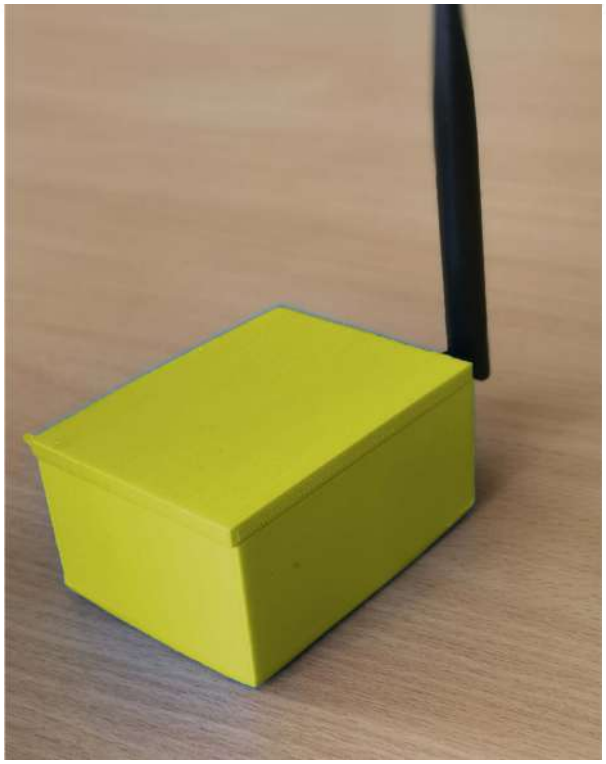


Figura 3.2: Estructura Final de la caja para el End Device

3.1.2 Escenario de Pruebas

La evaluación de retardos dentro de las retransmisión en la Red LoRaWAN se implemento en dos entornos: El primero, el cual se tenga una vista sin ningún tipo de obstáculos, y el segundo el cual la linea de vista se vea interrumpida por ciertos factores externos (edificio, árboles, etc), ambos a diferentes distancias entre el gateway y el nodo.

3.1.2.1 Entorno 1. - Av. Gónzales Suáres.

En la figura 3.3 muestra el escenario donde se llevo a cabo las pruebas realizadas en un ambiente en el cual no presenta ningún obstáculo estructural con una linea de vista recta entre el Gateway y el Nodo. Mostrando la distancia total de 200m en el recuadro del ubicado en la parte inferior derecha.

Figura 3.3: Escenario 1: Gónzales Suáres.

La figura 3.4, indica el perfil que tendrá el enlace para el escenario, visualizando casi una linea recta con tan una perdida de elevación de -2.96[m], por lo que se le considera un ambiente con una Linea de visión apta para las pruebas realizadas en el enlace.

Figura 3.4: Perfil de elevación Escenario 1

3.1.2.2 Entorno 2. - Parque la Carolina.

La figura 3.5 permite observar el entorno rural Ubicado en el Parque La Carolina donde presenta una serie de obstáculos que se consideran un factor que puede interferir en la medición de tiempos en los retardos en la retransmisión de la red LoRaWAN, como por ejemplo los árboles de mas de 4 metros de largo que no permiten visualizar correctamente el enlace. Es importante mencionar que las pruebas se realizaron a cada 20 metros hasta llegar a los 100 metros y a partir de los 200 metros se realizo 2 pruebas mas a partir de los 50m.

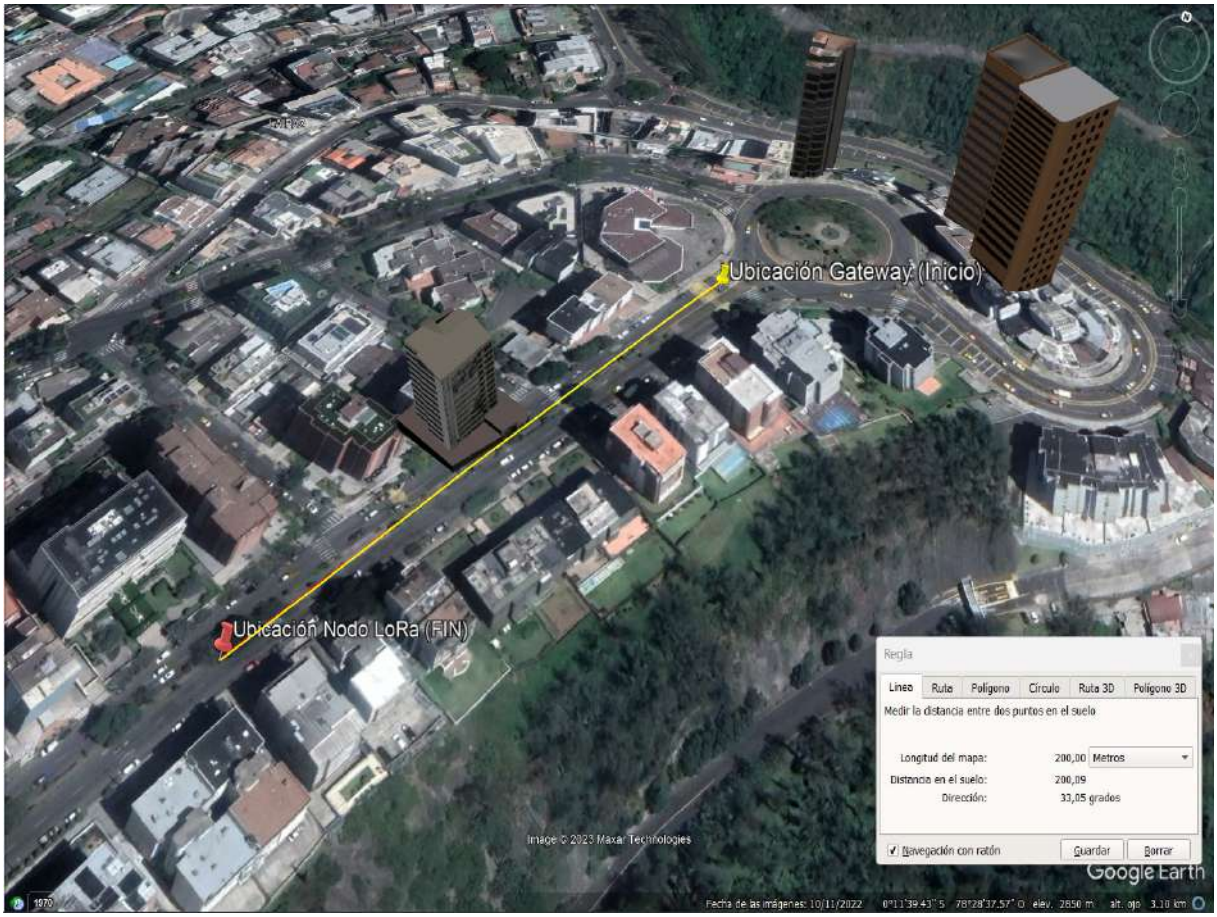


Figura 3.5: Escenario 2: Parque la Carolina.

Por otro lado, en la figura 3.6, se puede observar la diferencia en el perfil de elevación que existe mostrando un incremento de casi 6 metros y una pérdida de elevación de -2.52 [m] a una distancia total en el enlace de 200 metros aproximadamente. Mostrando que existe una irregularidad en el terreno debido a los montículos de tierra.

Figura 3.6: Perfil de elevación Escenario 2.

3.1.3 Resultados Obtenidos

Para la obtención de resultados es importante detallar algunas de las características que tomo en cuanta para la medición del retardo, entre ellos los mas relevantes son: Configuración del nodo para funcionar como clase A para una banda de frecuencia de definida en la US915 con frecuencia central en los 915MHz, también hay que tomar que en la formula 2.1 la distancia no es un factor directo en el cálculo del retraso de retransmisión y este no afecta directamente en el cálculo.

Sin embargo, la distancia puede tener un impacto indirecto en el retraso de retransmisión a través de su efecto en la calidad de la señal LoRa. Por lo cual el factor que se tomó en cuenta y afectara directamente al tiempo es el numero de retransmisiones que se coloque para la obtención del retardo. A continuación se presentara las mediciones obtenidas para cada uno de los escenarios.

3.1.3.1 Mediciones Entorno 1

Las pruebas realizadas para la medición de retardos se empleo variando el numero de retransmisiones en el código tanto para eACK e iACK, este parámetro se debe modificar en

la línea **LMIC_setTxData2(1, txBuffer, sizeof(txBuffer), 5)**, en el cuarto argumento de la función. Hay que recalcar que el numero máxima de retransmisiones que LoRaWAN acepta es de 8, y para evitar conflictos se vario retransmisiones entre 2 y 4 dependiendo de cada distancia en este caso se tendra una distancia total de 200m.

En los valores obtenidos se puede observar en la tabla 3.1 y 3.2, donde tanto para el valor de RSSI y SNR disminuyen mientras la distancia empieza a aumentar, por otro lado como resultado en los tiempos de retardo obtenidos se puede ver que en cuanto a las retransmisiones aumenta tal y como se tenia previsto.

Tabla 3.1: Valores obtenidos en el Entorno 1 mediante iACK

Distancia (m) Nodo-Gateway	Numero de Retransmisiones	RSSI (dBm)	SNR	Retardo Obtenido Serial Arduino (ms)	Retardo Calculado ms
20	1	-19	9.8	310	315.125
40	2	-25	10.8	316	326.525
60	2	-33	9.2	316	326.525
80	3	-44	12	320	343.482
100	2	-52	17	320	343.482
150	2	-75	20	321	343.482
200	3	-82	22	334	363.482

Tabla 3.2: Valores obtenidos en el Entorno 1 mediante eACK

Distancia (m) Nodo-Gateway	Numero de Retransmisiones	RSSI (dBm)	SNR	Retardo Obtenido Serial Arduino (ms)	Retardo Calculado ms
20	1	-17	7.8	323	335.125
40	2	-22	9.8	336	376.455
60	2	-33	11.2	336	376.455
80	3	-46	14.5	342	383.751
100	2	-48	18.5	342	385.485
150	2	-68	21.4	405	420.885
200	3	-76	22	420	420.885

3.1.3.2 Mediciones Entorno 2

Para las pruebas realizadas en este escenario se realizó de forma similar al anterior entorno con la diferencia que el gateway se coloco a una altura diferente esto debido a que se tenia mas obstrucción entre el gateway y el nodo.

Tabla 3.3: Valores obtenidos en el Entorno 2 mediante iACK

Distancia (m) Nodo-Gateway	Numero de Retransmisiones	RSSI (dBm)	SNR	Retardo Obtenido Serial Arduino (ms)	Retardo Calculado ms
20	1	-21	8.8	324	348.225
40	2	-23	10.8	348	348.225
60	2	-30	12.2	348	384.855
80	3	-38	13.5	378	384.855
100	2	-47	16.5	378	404.475
150	2	-58	20.4	410	404.475
200	3	-86	21.5	410	438.775

Tabla 3.4: Valores obtenidos en el Entorno 2 mediante eACK

Distancia (m) Nodo-Gateway	Numero de Retransmisiones	RSSI (dBm)	SNR	Retardo Obtenido Serial Arduino (ms)	Retardo Calculado ms
20	1	-17	10.8	360	398.125
40	2	-22	11.8	385	410.345
60	2	-25	14.2	385	410.345
80	3	-34	13.5	413	445.455
100	2	-48	18.5	413	445.455
150	2	-53	21.4	446	475.454
200	3	-72	23.5	446	475.454

Mediante los datos obtenidos en los dos entornos la medición de retardos que se pueden para el escenario 1 y 2, el iACK (ACK inmediato) proporciona una medición más precisa ya que confirma la recepción del mensaje casi inmediatamente después de que se envió y conexión a la TTN. Por otro lado, con el eACK (ACK explícito) el tiempo de confirmación puede ser variable, ya que depende del momento en que el servidor reciba el mensaje y envíe la confirmación de vuelta al nodo de extremo.

Sin embargo, en términos de fiabilidad, el eACK es más confiable ya que garantiza que el mensaje se haya entregado correctamente antes de que el nodo de extremo continúe con las siguientes operaciones. En resumen, para la medición precisa de retardos, es mejor utilizar iACK, mientras que para garantizar la fiabilidad de la entrega del mensaje, es mejor utilizar eACK.

3.2 CONCLUSIONES

- Tras realizar investigaciones de los distintos métodos de conexión para la placa LoRa, se logro determinar que la principal ventaja de la conexión tipo OTAA es la seguridad debido a que la sesión “se crea en el aire” y se renueva cada vez que el dispositivo pierde la conexión, o a su vez es apagado o reiniciado. Esto dificulta que alguien pueda robar la sesión y clonar el dispositivo y pueda obtener información en el envió de paquetes.

- Tras realizar distintas pruebas para los métodos de confirmación de entrega de mensajes (eACK e iACK), se puede concluir que son métodos utilizados por el nodo de extremo para confirmar que un mensaje a sido entregado con éxito. Donde para el eACK el nodo espera explícitamente una respuesta de confirmación de entrega del gateway después de haber enviado un mensaje, por lo que es más confiable debido a que garantiza que el mensaje se ha entregado correctamente antes de que el nodo continúe con las siguientes tareas.

- Mediante las distintas pruebas realizadas en los 2 entornos y al analizar los dashboards de la red LoRaWAN en la TTN se puede concluir que la operación de los nodos Wio E5 para la retransmisión de mensajes en una red LoRaWAN implica recibir solicitudes de retransmisión del servidor de red en este caso las peticiones del Gateway y este vuelve a enviar los mensajes originales para mejorar el alcance y la confiabilidad de la red.

- Al momento de realizar las pruebas en los escenarios se pudo concluir que los Obstáculos pueden aumentar la atenuación de la señal, que como resultado puede aumentar el número de retransmisiones necesarias para que el mensaje llegue a su destino final. Como resultado, el retardo total de la transmisión del mensaje puede verse afectado por la distancia entre los dispositivos, debido a que se puede requerir múltiples retransmisiones para superar las interferencias y obstáculos que reducen la calidad de la señal.

- Tras revisar los modos para establecer la comunicación entre el nodo y el gateway se pudo determinar que al momento de establecer el modo de verificación de enlace en desactivado (es decir 0), el dispositivo no enviará solicitudes de verificación al enlace, por lo que no recibirá respuestas de verificación de enlace del servidor de red. Esto

puede ser útil en situaciones donde la batería es limitada o cuando no es importante saber el estado de real del dispositivo.

3.3 RECOMENDACIONES

- ❒ Se recomienda trabajar con el firmware por defecto de los nodos LoRa-E5, ya que al momento de trabajar con programación en STM32, existe una gran complejidad al momento de tratar de realizar la conexión con el gateway y a su vez tratar de subir el programa mediante el programador.
- ❒ Es importante tener en cuenta que la retransmisión de mensajes puede aumentar el consumo de energía del nodo Wio E5, por lo que es posible que se deba equilibrar la necesidad de retransmisión con la vida útil de la batería del dispositivo. Por lo tanto, es importante tener en cuenta la distancia entre los dispositivos al planificar la red LoRaWAN y determinar el número de retransmisiones necesarias para garantizar una comunicación confiable.
- ❒ Se recomienda comprender la lista de comando AT prevista por LoRa debido a que al momento de subir un programa en código C, el Nodo pierde características importantes como el modo promiscuo o el modo Test que permite conectarse de manera automática al gateway de la Red LoRaWAN y evaluar el retardo de una manera mas rápida, y esto a su vez provoca que la falta de documentación para realizar esto mediante la programación en STM sea mas compleja y pueda causar problemas en la placa.
- ❒ Es importante tener en cuenta que la retransmisión de mensajes en la red LoRaWAN está controlada por el servidor de red, y la cantidad de retransmisiones permitidas depende de la configuración de la red y del perfil de dispositivo utilizado. Además, la retransmisión de mensajes consume más energía del dispositivo, por lo que se debe tener en cuenta el consumo de energía al decidir si utilizar mensajes confirmados y/o retransmisiones en su aplicación.
- ❒ Es importante tomar en en cuenta que el mensaje a enviar no puede ser más largo que el tamaño máximo de payload permitido por en la red LoRaWAN, este tamaño no debe superar los 51 bytes y si se desea enviar mensajes más grandes, se recomienda fragmentarlos y enviarlos en varias tramas.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] *¿Qué es el Internet de las cosas (IoT)?* es, Temas IoT, ene. de 2022. dirección: https://www.oracle.com/ar/internet-of-things/what-is-iot/ (visitado 10-11-2022).

[2] L. Clausin, *IoT: 4 Ejemplos de dispositivos del Internet de las Cosas en la vida cotidiana*, es, nov. de 2021. dirección: https://www.nts-solutions.com/blog/internet-de-las-cosas-ejemplos.html (visitado 15-11-2022).

[3] *Redes de sensores - TEKNIKER*, es, 2021. dirección: https://www.tekniker.es/es/redes-de-sensores (visitado 17-11-2022).

[4] D. Tibaduiza, N. Chio, C. Aparicio y M. Ortiz, «Redes de Sensores Inalámbricos,» es, pág. 1, jul. de 2007. dirección: https://www.academia.edu/5366011/Redes_de_Sensores_Inal%C3%A1mbricos (visitado 17-11-2022).

[5] *Redes LPWAN: LoRaWAN, SigFox, Helium, NB-IOT, ZigBee... – LPWAN – Todo sobre las redes LoRaWAN, Sigfox, NB-IOT*, es, 2022. dirección: https://lpwan.es/ (visitado 18-11-2022).

[6] L. A. V. Medina y B. E. Z. Soledispa, «Diseño de una red LPWAN basada en tecnología LoRa para las estaciones hidrometeorológicas",» es, pág. 92, oct. de 2018. dirección: https://www.dspace.espol.edu.ec/bitstream/123456789/47558/1/D-CD106675.pdf.

[7] C. A. G. González, F. A. Tapias y J. H. Gutiérrez, «Análisis de seguridad en redes LPWAN para dispositivos IoT,» es, *Revista Vínculos*, vol. 16, n.º 2, págs. 252-261, dic. de 2019, Number: 2, ISSN: 2322-939X. DOI: 10.14483/2322939X.15712. dirección: https://geox.udistrital.edu.co/index.php/vinculos/article/view/15712/15352 (visitado 18-11-2022).

[8] T. T. Network, *¿Qué es la tecnología LoRa y por qué es importante para IoT?* en. dirección: https://www.thethingsnetwork.org/community/santa-rosa/post/que-es-la-tecnologia-lora-y-por-que-es-importante-para-iot (visitado 18-11-2022).

[9] J. de Carvalho Silva, J. Rodrigues, A. Alberti, P. Šolić y A. Aquino, *LoRaWAN - A Low Power WAN Protocol for Internet of Things: a Review and Opportunities*. jul. de 2017.

[10] J. Vargas, *Conceptos básicos que te ayudarán a entender LoRa y LoRaWAN*, es-ES, jun. de 2022. dirección: https://www.m2mlogitek.com/conceptos-tecnicos-basicos-que-te-ayudaran-a-entender-lora-y-lorawan-low-power-wide-area-network-en-pocos-minutos/ (visitado 18-11-2022).

[11] M. A. Moya Quimbita, «Evaluación de pasarela LoRa/LoRaWAN en entornos urbanos,» es, pág. 40, 2018.

[12] Sabas, *Haciendo IoT con LoRa: Capitulo 2.- Tipos y Clases de Nodos*, en, IoT, nov. de 2017. dirección: https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be (visitado 18-11-2022).

[13] H. Alonso, *LoRaWAN – Parte 3 – Arquitectura y tramas*, en-US, jun. de 2020. dirección: https://elbosquedesilicio.es/lorawan3/ (visitado 23-11-2022).

[14] *LoRa- (Long Range) Network and Protocol Architecture & Frame Structure*, en, oct. de 2018. dirección: http://www.techplayon.com/lora-long-range-network-architecture-protocol-architecture-and-frame-formats/ (visitado 23-11-2022).

[15] R. Pérez, «Evaluación de LoRa/LoRaWAN para escenarios de Smart City,» *Ιinea] Tomado de: https://upcommons. upc. edu/bitstream/handle/2117/100922/memoria. pdf*, 2017.

[16] J. S. Duque Escobar, «SISTEMA DE MONITOREO IOT DE CALIDAD DE AIRE USANDO UNA RED DE SENSORES FIJOS Y LORAWAN EN EL CAMPUS SANGOLQUÍ DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE,» es, pág. 183, ene. de 2020. (visitado 23-11-2022).

[17] Baeldung, *Retardo de propagación frente a retardo de transmisión | Baeldung sobre informática*, en, jun. de 2021. dirección: https://www.baeldung.com/cs/propagation-vs-transmission-delay (visitado 22-11-2022).

[18] A. Mishra, *Delays in Computer Network*, en, Section: Computer Networks, sep. de 2021. dirección: https://www.geeksforgeeks.org/delays-in-computer-network/ (visitado 22-11-2022).

[19] *TTN – Aprendiendo Arduino*, mar. de 2018. dirección: https : / / www . aprendiendoarduino.com/tag/ttn/ (visitado 15-02-2023).

[20] F. Campos, *Empezando a trabajar con LoRaWAN: Nodos y Gateways:* es-ES, mar. de 2021. dirección: https://www.m2mlogitek.com/empezando-a-trabajar-con-lorawan-ii/ (visitado 22-01-2023).

[21] *Módulo Wio-E5 STM32WLE5JC - Seeed Wiki*, en-US, 2022. dirección: https://wiki.seeedstudio.com/LoRa-E5_STM32WLE5JC_Module/ (visitado 25-01-2023).

[22] L. Llamas, *Programar STM32 con IDE de Arduino y ST-Link v2*, es, sep. de 2018. dirección: https://www.luisllamas.es/programar-stm32-con-ide-de-arduino-y-st-link-v2/ (visitado 31-01-2023).

[23] «ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32,» en, n.º 4, págs. 2-3, 2016. dirección: https://www.farnell.com/datasheets/2307721.pdf.

[24] J. Damian, *Arduino Nano Pinout y características – Electrogeek*, es, mar. de 2020. dirección: https://www.electrogeekshop.com/arduino-nano-pinout-y-caracteristicas/ (visitado 15-02-2023).

[25] *Laird Sentrius™ RG191 LoRaWAN Gateway - US915*, gb, ene. de 2022. dirección: https://connectedthings.store/gb/lorawan-gateways/indoor-lorawan-gateways/laird-sentrius-rg191-lorawan-gateway-us915.html (visitado 15-02-2023).

[26] *Baterías 18650 Qué son? Cuáles son sus ventajas?* es, feb. de 2021. dirección: https://linternafrontal.pro/baterias-18650/ (visitado 31-01-2023).

[27] *Baterías 18650: Características, usos y ventajas*, es, jul. de 2019. dirección: https://www.madridiario.es/noticia/470346/recomendamos/baterias-18650:-caracteristicas-usos-y-ventajas.html (visitado 31-01-2023).

[28] *STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics*, en-US, Tecnología, 2022. dirección: https://www.st.com/en/development-tools/stm32cubeide.html (visitado 01-02-2023).

[29] *STM32CubeProg - STM32CubeProgrammer software for all STM32 - STMicroelectronics*, en-US, Tecnología, 2022. dirección: https://www.st.com/en/development-tools/stm32cubeprog.html (visitado 01-02-2023).

[30] *STSW-LINK004 - STM32 ST-LINK utility (replaced by STM32CubeProgrammer) - STMicroelectronics*, en, 2022. dirección: https://www.st.com/en/development-tools/stsw-link004.html (visitado 01-02-2023).

[31] «LoRa AT Command Manual,» es-ES, págs. 5-9, dirección: https://www.tme.eu/Document/6fd083ba11ae9feedaadc9c7221ab46e/eWBM_LoRa_AT_Command_v0.6.pdf (visitado 15-02-2029).

[32] *LoraWan - Caso práctico TTN alta de Gateway y Aplicación. Parte 2*, es, IoT, ene. de 2021. dirección: https://bconsultors.com/2021/01/07/lorawan-caso-practico-ttn-alta-de-gateway-y-aplicacion-parte-2/ (visitado 16-02-2023).

[33] *GitHub - Seeed-Studio/LoRaWan-E5-Node at qian*, en-US, mar. de 2022. dirección: https://github.com/Seeed-Studio/LoRaWan-E5-Node (visitado 16-02-2023).

[34] M. Terris, *MCCI Arduino LoRaWAN Library*, original-date: 2016-10-24T05:26:27Z, jun. de 2022. dirección: https://github.com/mcci-catena/arduino-lorawan (visitado 21-02-2023).

[35] *LoRaWAN® Specification v1.0.3*, en-US, 2018. dirección: https://hz1.37b.myftpupload.com/resource_hub/lorawan-specification-v1-0-3/ (visitado 23-02-2023).

5 ANEXOS

ANEXO 1

```
#define STATIC_DEVICE_EUI                                   0

/*!
 * end-device IEEE EUI (big endian)
 */
#define LORAWAN_DEVICE_EUI                                  { 0x70, 0xB3, 0xD
    5, 0x7E, 0xD0, 0x05, 0xA8, 0x55 }

/*!
 * App/Join server IEEE EUI (big endian)
 */
#define LORAWAN_JOIN_EUI                                    { 0x80, 0x00, 0x0
    0, 0x00, 0x00, 0x00, 0x00, 0x07 }

/*!
 * When set to 1 DevAddr is LORAWAN_DEVICE_ADDRESS
 * When set to 0 DevAddr is automatically set with a value provided by a
    pseudo
 *      random generator seeded with a value provided by the MCU platform
 */
#define STATIC_DEVICE_ADDRESS                               0

/*!
 * Device address on the network (big endian)
 */
#define LORAWAN_DEVICE_ADDRESS                              ( uint32_t )0x010
    0000A

/*!
 * Application root key
 */
#define LORAWAN_APP_KEY                                     3A,FD,F7,BC,7B,80
    ,49,7C,C8,ED,1A,20,F9,FB,90,19


/*!
```

```
 * Network root key
 */
#define LORAWAN_NWK_KEY                                 3A,FD,F7,BC,7B,80
    ,49,7C,C8,ED,1A,20,F9,FB,90,19

/*!
 * Forwarding Network session key
 */
#define LORAWAN_NWK_S_KEY                               3A,FD,F7,BC,7B,80
    ,49,7C,C8,ED,1A,20,F9,FB,90,19

/*!
 * Application session key
 */
#define LORAWAN_APP_S_KEY                               3A,FD,F7,BC,7B,80
    ,49,7C,C8,ED,1A,20,F9,FB,90,19

/*!
 * Format commissioning keys
 */
#define RAW_TO_INT8A(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p) {0x##a,0x##b,0x##c,0x
    ##d,\
                                                      0x##e,0x##f,0x##g,0x
    ##h,\
                                                      0x##i,0x##j,0x##k,0x
    ##l,\
                                                      0x##m,0x##n,0x##o,0x
    ##p}

#define FORMAT_KEY(...) RAW_TO_INT8A(__VA_ARGS__)

#if (USE_LRWAN_1_1_X_CRYPTO == 1)
#define SESSION_KEYS_LIST
                                                    \
        {
                                                    \
            /*!
                                                    \
             * Join session integrity key (Dynamically updated)
                                                    \
             * WARNING: NOT USED FOR 1.0.x DEVICES
                                                    \
```

```
-              */                                                  \
-             .KeyID     = J_S_INT_KEY,                           \
-             .KeyValue = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
    00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \
65                           0x00 },                              \
-         },                                                      \
-         {                                                       \
-             /*!                                                \
-              * Join session encryption key (Dynamically updated)\
70            * WARNING: NOT USED FOR 1.0.x DEVICES              \
-              */                                                 \
-             .KeyID     = J_S_ENC_KEY,                          \
-             .KeyValue = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
    00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \
-                           0x00 },                              \
75       },                                                      \
-         {                                                       \
-             /*!                                                \
-              * Forwarding Network session integrity key        \
-              * WARNING: NWK_S_KEY FOR 1.0.x DEVICES            \
80            */                                                 \
-             .KeyID     = F_NWK_S_INT_KEY,                      \
-             .KeyValue = FORMAT_KEY(LORAWAN_NWK_S_KEY),         \
```

```
        },                                                  \
        {                                                   \
            /*!                                            \
             * Serving Network session integrity key       \
             * WARNING: NOT USED FOR 1.0.x DEVICES. MUST BE THE SAME AS \ref LORAWAN_F_NWK_S_INT_KEY \
             */                                             \
            .KeyID     = S_NWK_S_INT_KEY,                   \
            .KeyValue = { 0x3A, 0xFD, 0xF7, 0XBC, 0x7B, 0x80, 0x49, 0x7C, 0xC8, 0xED, 0x1A, 0x20, 0xF9, 0xFB, 0x90, \
                          0x19 },                           \
        },                                                  \
        {                                                   \
            /*!                                            \
             * Network session encryption key              \
             * WARNING: NOT USED FOR 1.0.x DEVICES. MUST BE THE SAME AS \ref LORAWAN_F_NWK_S_INT_KEY \
             */                                             \
            .KeyID     = NWK_S_ENC_KEY,                     \
            .KeyValue = { 0x3A, 0xFD, 0xF7, 0XBC, 0x7B, 0x80, 0x49, 0x7C, 0xC8, 0xED, 0x1A, 0x20, 0xF9, 0xFB, 0x90, \
                          0x19 },                           \
        },                                                  \
        {                                                   \
            /*!                                            \
```

```
-               * Application session key
                                                    \
105             */
                                                    \
-               .KeyID     = APP_S_KEY,
                                                    \
-               .KeyValue = FORMAT_KEY(LORAWAN_APP_S_KEY),
                                                    \
-           },
- #else /* USE_LRWAN_1_1_X_CRYPTO == 0 */
```

Código 5.1: Código para configurar ID de la placa Wio-E5 (completo).

ANEXO 2

```
1  #include <Arduino.h>
-  #include <lmic.h>
-  #include <hal/hal.h>
-  #include <SPI.h>
5
-  // Definir las credenciales de la red LoRaWAN
-  static const u1_t PROGMEM DEVEUI[8]={ 0x00, 0x80, 0xE1, 0x15, 0x00, 0x17, 0
       xCD, 0xD7 };
-  static const u1_t PROGMEM APPEUI[8]={ 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
       00, 0x08 };
-  static const u1_t PROGMEM APPKEY[16]={0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0
       xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F,0x3C };
10
-  // Definir el objeto de transmisi n
-  static osjob_t sendjob;
-  static uint8_t txBuffer[255];
-  static uint32_t start_time = 0;  // Variable para guardar el tiempo de
       env o del retardo
15
-  // Funci n para enviar un mensaje al servidor LoRaWAN
-  void do_send(osjob_t* j) {
-
-   // Guardar el tiempo actual en la variable start_time
20   start_time = millis();
-
```

```
  // Preparar el mensaje a enviar
  memcpy(txBuffer, "Hola, vengo desde End Node 1", 28); // mensaje a enviar
  LMIC.frame[0] = 0x02;  // Tipo de mensaje confirmed (espera un iACK)
  LMIC_setTxData2(1, txBuffer, sizeof(txBuffer), 1);  // Enviar el mensaje

  // Imprimir el mensaje enviado y la frecuencia utilizada
  Serial.print("Mensaje enviado: ");
  for (int i = 0; i < sizeof(LMIC.frame); i++) {
    Serial.print(LMIC.frame[i], HEX);
    Serial.print(" ");
  }
  Serial.print("en la frecuencia ");
  Serial.println(LMIC.freq);

  // Esperar la respuesta de iACK durante 5 segundos
  os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(5), do_send);


}

void onEvent (ev_t ev) {
  switch(ev) {
    case EV_TXCOMPLETE:
      Serial.println("Transmisi n completa");
      break;
    case EV_RXCOMPLETE:
      Serial.print("Recepci n completa. RSSI=");
      Serial.println(LMIC.rssi);
      break;
    case EV_JOINED:
      Serial.println("Unido a la red LoRaWAN Jona-Tesis");
      break;
    case EV_ACK_COMPLETE:
      Serial.println("iACK recibido despues de");
      Serial.print(millis() - start_time);
      Serial.println(" ms");
      break;
    default:
      Serial.println("Evento desconocido");
      break;
  }
}
```

```
-
void setup() {
  Serial.begin(115200);
  Serial.println("Iniciando...");
-
  // Inicializar la biblioteca LoRaWAN
  os_init();
  LMIC_reset();
  LMIC_setSession (0x1, DEVEUI, APPEUI, APPKEY);
-
  // Configurar la frecuencia de transmisi n y la potencia de salida
  LMIC_selectSubBand(1);
  LMIC_setDrTxpow(DR_SF7,14);
-
  // Establecer la confirmaci n de entrega
  LMIC_setLinkCheckMode(1);

  // Configurar la funci n de devoluci n para los eventos de LoRaWAN
  LMIC_registerEventCb(onEvent);
-
  // Iniciar la transmisi n del mensaje
  os_setCallback(&sendjob, do_send);
}
-
void loop() {
  os_runloop_once();
}
```

Código 5.2: Código completo para la placa Wio-E5 (END NODE).