ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

REDES DE SENSORES INALÁMBRICOS PARA IOT

ALGORITMO PARA LA ASIGNACIÓN AUTOMÁTICA DE DIRECCIONES A NODOS EN REDES LORAWAN MULTISALTO CON TOPOLOGÍA LINEAL

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

SEBASTIÁN DAVID BOLAÑOS ENRÍQUEZ
sebastian.bolanos@epn.edu.ec

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

DMQ, abril 2023

CERTIFICACIONES

Yo, SEBASTIÁN DAVID BOLAÑOS ENRÍQUEZ 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.

SEBASTIÁN DAVID BOLAÑOS ENRÍQUEZ

Certifico que el presente trabajo de integración curricular fue desarrollado por SEBASTIÁN DAVID BOLAÑOS ENRÍQUEZ, bajo mi supervisión.

CARLOS ROBERTO EGAS ACOSTA

DIRECTOR

DECLARACIÓN DE AUTORÍA

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

SEBASTIÁN DAVID BOLAÑOS ENRÍQUEZ

CARLOS ROBERTO EGAS ACOSTA

DEDICATORIA

Este trabajo va dedicado a todos aquellos quienes estuvieron a mi lado a lo largo de este proceso. A mi familia quienes supieron estar de inicio a fin, y me apoyaron a seguir adelante, sin ustedes esto hoy tal vez sería posible, pero no sería un propósito tan importante.

Este trabajo va dedicado a mis padres que me enseñaron a ir más allá y a no rendirme, si no seguir avanzando con más fuerzas.

Esto va dedicado para ustedes familia.

AGRADECIMIENTO

Gracias a todos los que creyeron siempre en mí.
Gracias a mi papá por esforzarse tanto para yo llegar hasta este momento, gracias a mi mamá, que con sus oraciones siempre me bendijo.

Quiero agradecer a mis tías y familiares que me supieron apoyar aún en la distancia.

Hoy cierro un ciclo más, no sin antes reconocer toda la ayuda que he tenido también por parte de todos mis profesores, sin su instrucción, esto no sería posible; igualmente a mis compañeros, con quienes pude compartir varios conocimientos y trabajos, gracias por enseñarme que en quipo todo es mejor.

Gracias a todos y cada uno de ustedes, que en cada parte de mi vida han ido sumando con conocimiento, con amor, con esfuerzo y de más.

ÍNDICE DE CONTENIDO

CERTIFICACIONES ..... **I**
DECLARACIÓN DE AUTORÍA ..... **II**
DEDICATORIA ..... **III**
AGRADECIMIENTO ..... **IV**
ÍNDICE DE CONTENIDO ..... **V**
1 INTRODUCCIÓN **1**
1.1 OBJETIVO GENERAL 2
1.2 OBJETIVOS ESPECÍFICOS 2
1.3 ALCANCE 2
1.4 MARCO TEÓRICO 3
1.4.1 INTRODUCCIÓN AL INTERNET DE LAS COSAS (IoT) 3
1.4.2 REDES INALÁMBRICAS DE SENSORES 4
1.4.2.1 Elementos de una red inalámbrica de sensores 5
1.4.2.2 Topología lineal de una red de sensores inalámbricas 5
1.4.3 REDES LoRa 7
1.4.3.1 Propiedades de una red LoRa 7
1.4.3.2 Estructura de la trama LoRa 8
1.4.4 PROTOCOLO LoRaWAN 8
1.4.4.1 Clases de nodos finales LoRaWAN 9
2 METODOLOGÍA **11**
2.1 LoRa-E5 DEV BOARD 11
2.1.1 ELEMENTOS PARA CONFIGURACIÓN DEL NODO LORA-E5 14
2.2 DISEÑO DEL ALGORITMO 14
2.2.1 DIAGRAMA DE FLUJO DEL ALGORITMO 15
2.2.2 CODIFICACIÓN DEL ALGORITMO 17
2.2.2.1 Codificación para la asignación de dirección del nodo coordinador 18
2.2.2.2 Codificación para la asignación de la nueva dirección del nodo receptor 19
2.3 HARDWARE PARA LA IMPLEMENTACIÓN 20
2.3.1 PROGRAMADOR DE MICROCONTROLADORES 20
2.3.1.1 ST-Link V2 21
2.3.2 ARDUINO UNO 21

2.4 SOFTWARE PARA LA IMPLEMENTACIÓN del algoritmo 22
2.4.1 ADICIÓN DE TARJETAS STM32 EN ARDUINO IDE 22
2.4.2 ACTUALIZACIÓN DE FIRMWARE PARA ST-LINK V2 27
2.4.3 CONFIGURACIÓN INICIAL DEL NODO LORA-E5 28
2.5 IMPLEMENTACIÓN DEL PROTOTIPO 32
3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES 37
3.1 RESULTADOS 38
3.1.1 ESCENARIO 1: ENCENDIDO DEL NODO COORDINADOR 38
3.1.2 ESCENARIO 2: ENCENDIDO DEL NODO COORDINADOR Y UN NODO .. 40
3.1.2.1 Nodo a 10 metros de distancia 41
3.1.2.2 Nodo a 1 kilómetro de distancia 43
3.2 CONCLUSIONES 46
3.3 RECOMENDACIONES 47
4 REFERENCIAS BIBLIOGRÁFICAS 48
5 ANEXOS 51

RESUMEN

El presente Trabajo Curricular expone una solución para una asignación dinámica de direcciones para los nodos, lo que puede afectar tanto al rendimiento de la topología como al no tener un orden establecido en sus direcciones al momento de controlarlas, siendo una solución la creación de un algoritmo que permita asignar automáticamente las direcciones de los nodos en una topología lineal.

El primer capítulo brinda un pequeño marco teórico para informar al lector temas de interés relacionado al tema, donde se discuten conceptos como las redes inalámbricas de sensores enfocándose en los elementos que componen dicha red y explicando una de sus estructuras que es la topología lineal, redes LoRa y el protocolo LoRaWAN.

El segundo capítulo trata de los elementos necesarios para construir un prototipo que asigne automáticamente las direcciones de los nodos. Pero para ello, primero se realiza un análisis del nodo LoRa-E5 con el que se trabaja. Seguido de ello se genera un diagrama de flujo para entender la forma en la que se debe trabajar. Finalmente se implementa el prototipo con dos nodos, explicando la forma en la que se consigue preparar el software y el hardware. Se realizan las pruebas necesarias.

En el tercer capítulo se lleva a cabo la presentación de los resultados obtenidos del prototipo implementado en el capítulo anterior. Se detallan dichos resultados en tablas para un mejor entendimiento, comentando cada uno de ellos. Para finalizar, se mencionan las conclusiones y recomendaciones encontradas a lo largo del Trabajo Curricular realizado.

PALABRAS CLAVE: Asignación, direcciones, automático, LoRa-E5, DevAddr, STM32.

ABSTRACT

The present Project exposes a solution for a dynamic assignment of addresses for the nodes, which can affect both the performance of the topology and not having an established order in their addresses when controlling them, being a solution the creation of an algorithm that allows to automatically assign the addresses of the nodes in a linear topology.

The first chapter provides a small theoretical framework to inform the reader of topics of interest related to the subject, where concepts such as wireless sensor networks are discussed, focusing on the elements that make up said network and explaining one of its structures, which is linear topology, networks LoRa and the LoRaWAN protocol.

The second chapter deals with the necessary elements to build a prototype network that automatically assigns the addresses of the nodes. But for this, first an analysis of the LoRa-E5 node with which it works is carried out. Following this, a flowchart is generated to understand the way in which it should work. Finally, the prototype with two nodes is implemented, explaining the way in which the software and hardware can be prepared. The necessary tests are carried out.

In the third chapter the previous results obtained from the prototype implemented in the previous chapter are presented. These results are detailed in tables for a better understanding, commenting on each of them. In the end, the conclusions and recommendations found along of the Project carried out that are mentioned.

KEYWORDS: Assign, address, automatic, LoRa-E5, DevAddr, STM32.

1 INTRODUCCIÓN

A lo largo de los años se ha ido incrementando poco a poco el uso de aplicaciones con IoT, donde se trata de crear un sistema conectado entre dispositivos y usuarios, cambiando así la manera en la que vivimos para mejorar las actividades diarias, tanto en lo personal como en lo laboral [1]. Parte de este incremento viene ligado a la conexión mediante redes inalámbricas de sensores, que ha tenido un crecimiento acelerado para brindar las satisfacciones necesarias de una manera automatizada [2].

Las redes inalámbricas de sensores, al estar conectados entre sí, pueden tener distintas topologías de acuerdo con su uso, por ejemplo, la topología lineal, que puede ser utilizada en aplicaciones de sensado en estructuras lineales, como el monitoreo de oleoductos, tuberías de agua y otras aplicaciones futuras que posean una infraestructura lineal a gran escala [3].

Para ello, una tecnología que ha ido tomando fuerza poco a poco ha sido LoRa, la cual nos permite tener una conexión inalámbrica mediante el uso de nodos sensores. Estos nodos tienen la característica de que consumen un nivel bajo de batería, haciendo que se genere un ahorro en cuanto a estructura física (cableado en transmisión de datos y cableado de energía) e implementación, además de tener una capacidad de comunicación entre otros nodos en rangos de las decenas de kilómetros (entre 10 [km] y 15 [km], dependiendo las condiciones externas) [4].

Pero no todo lo que brinda son ventaja, pues a su vez existen ciertas limitaciones en la utilización cientos de nodos en las topologías lineales, ya que en varios casos son redes multisalto para lo cual es necesario un protocolo de enrutamiento adecuado. Es decir, al ser una topología lineal, solamente existe una ruta, por lo que no es necesario un protocolo de enrutamiento a nivel de red debido a que este necesita grandes cantidades de energía y se tiene retardos mayores [3].

Para afrontar dichos retos se emplean diferentes soluciones con las que es posible automatizar los nodos brindados en la topología, como lo es la asignación automática de direcciones globales mediante el protocolo LoRa, que trabaja a nivel de enlace, con lo cual es posible conseguir una conectividad eficiente entre los nodos para mejorar su rendimiento y optimizar los recursos utilizados [5].

1.1 OBJETIVO GENERAL

Generar un algoritmo que permita asignar automáticamente las direcciones de los nodos en una red inalámbrica LoRaWAN con topología lineal.

1.2 OBJETIVOS ESPECÍFICOS

1. Estudiar las redes inalámbricas de sensores, enfocado a la tecnología LoRa y su protocolo LoRaWAN.
2. Seleccionar un nodo LoRa que se adecúe a las características técnicas necesarias para cumplir los requisitos del algoritmo.
3. Diseñar un algoritmo que asigne automáticamente las direcciones de un nodo LoRa que forma parte de una topología lineal.
4. Implementar el algoritmo en un prototipo de red LoRaWAN con topología lineal multisalto.
5. Evaluar el prototipo implementado mediante las pruebas necesarias para recopilar la información deseada.

1.3 ALCANCE

El Trabajo de Integración Curricular se centra en la creación de un algoritmo que nos permita asignar automáticamente las direcciones en los nodos de una red LoRaWAN multisalto con topología lineal. Para ello se debe investigar los temas relacionados para realizar un algoritmo e implementarlo, y así conocer la factibilidad que brindan los mismos.

En primer lugar, se realizará una recopilación de información bibliográfica acerca de las redes inalámbricas de sensores, enfocado a la tecnología LoRa y su protocolo LoRaWAN, así como las tramas que este protocolo brinda para acceder al canal.

Además, se estudiará las características de una topología lineal y a partir de ello se diseñará un algoritmo que genere una asignación automática de las direcciones en un nodo LoRa, obteniendo así un diagrama de flujo que nos permita entender de mejor manera la secuencia del algoritmo.

Una vez realizado el diseño, es necesario conocer las características que hacen posible la implementación de dicho algoritmo, por lo que se realizará una investigación que pueda suplir las dudas y condiciones del algoritmo.

En base a la información recabada de las características necesarias del diseño se trabajará en la búsqueda de un nodo adecuado con el que se realizará la implementación.

Finalmente, se evaluará el algoritmo en la topología adecuada, con un nodo en específico, y se realizarán pruebas para la verificación de su aplicación y posterior a ello obtener un análisis de los resultados obtenidos. Con ello se podrá realizar diferentes conclusiones en base a las redes inalámbricos de sensores, la tecnología LoRa y su asignación automática de direcciones.

1.4 MARCO TEÓRICO

1.4.1 INTRODUCCIÓN AL INTERNET DE LAS COSAS (IOT)

Actualmente el Internet es usado globalmente donde es posible enlazarse a distintos dispositivos mediante la red, lo cual nos lleva a la necesidad de crear escenarios de conexión entre dispositivos y sensores que se usan cotidianamente, pero con una capacidad informática enorme [6].

El término “Internet de las Cosas”, o más conocido como IoT (por sus siglas en inglés) fue nombrado por Kevin Ashton en 1999. Este científico lo describe como un sistema a través de Internet donde los dispositivos permitan conectarse entre sí mediante sensores. El término lo utilizó para expresar la manera en la que es posible conectar un sistema de almacenamiento y recuperación de datos remotos (RFID) con el que pudiera mantener un conteo y rastreo óptimo de la mercadería de las empresas corporativas sin necesidad de que intervenga el ser humano [7].

El IoT se ha desplegado en distintas aplicaciones, siendo usado con mayor frecuencia el ámbitos tecnológicos, industriales, ingenierías e inclusive económicos y políticos. Debido a esto varias organizaciones mundiales han propuesto distintas definiciones, entre ellas las más reconocidas son la IEFT (Internet Engineering Task Force), IAB (Internet Architecture Board) y la ITU (International Telecommunication Union) [6]. A pesar de que cada organización ha brindado su definición, la más reconocida fue brindada por la IEEE Communications Magazine, la cual describe a la IoT como un marco donde todas las cosas tienen una presencia y representan cierta caracterización en Internet. Es decir, que la

finalidad de la IoT es ofertar nuevos servicios con los que sea posible una conexión entre el mundo virtual con el mundo físico, donde las comunicaciones 'máquina a máquina' (M2M) simbolizan una comunicación para la interacción de las cosas (dispositivos) con las aplicaciones de la nube [7].

Dicha tecnología se encuentra integrada en una gran variedad de dispositivos, sensores de red y sistemas, que sacan ventaja de los avances tecnológicos como la miniaturización electrónica, la potencia de cómputo y las interconexiones de red para presentar nuevas y variadas capacidades que hace años no era factible [6].

1.4.2 REDES INALÁMBRICAS DE SENSORES

El objetivo principal de una red de señores inalámbricas es entender el mecanismo de control e implementar una monitorización a los fenómenos existentes en el entorno.

Anteriormente, la conectividad entre sensores se llevaba a cabo mediante la utilización de una red cableada. Sin embargo, con el pasar de los años la tecnología en las comunicaciones ha ido incrementando a tal punto que distintos tipos de dispositivos cuentan con comunicaciones inalámbricas donde pueden ser utilizados en distintas ubicaciones y a un costo relativamente bajo, siendo además dispositivos cada vez más compactos, pero con un rendimiento eficaz, con consumo de energía bajo y multifuncionales. Gracias a todas las características mencionadas anteriormente nace la idea de las redes inalámbricas de sensores o WSN (Wireless Sensor Network) [8].

Una WSN es una agrupación de dos o más dispositivos autónomos denominados nodos que pueden almacenar datos, los cuales tienen la capacidad de comunicarse entre sí mediante el uso de sensores para poder conducir la información por toda la red de forma inalámbrica y con ello realizar las distintas actividades planificadas [9].

Existen ciertos parámetros que influyen en la red de sensores inalámbrica, por lo que se deben considerar a la hora de realizar algún diseño. Algunos de estos parámetros son [10]:

- **Energía:** Los sensores de la red utilizan baterías, por lo que se debe monitorear su nivel de energización para poder recargarlas o a su vez reemplazarlas en caso de ser necesario. Así, el tiempo de operatividad de la red se prolongará. En caso de que el sensor no se encuentre procesando información es posible mantenerlo en un modo de hibernación para extender la vida útil del elemento.
- **Redundancia:** Si se quiere que la red sea robusta y confiable es necesario que la misma garantice la transmisión y recepción de la información. Puede ser con nodos

con tecnología que permita recorrer la información por distintos caminos, siendo así una forma de certificar un enlace redundante.

- **Escalabilidad:** Se sugiere que la red, al momento de ser diseñada, tenga espacio para seguir creciendo en caso de que su aplicación lo necesite, por lo que se debe estimar un cierto porcentaje de ampliación de la red sin que esta afecte a la funcionabilidad.

- **Topología:** Dependiendo la aplicación, la forma en la que se unan los nodos influye en gran medida al rendimiento de la red, mejorando o afectando su redundancia, la energía que consume u otros factores. Puede ser topología lineal, estrella, árbol, entre otras.

1.4.2.1 Elementos de una red inalámbrica de sensores

Una red de sensores inalámbricas se compone de los siguientes elementos [8]:

- **Sensor:** Elemento que captura la información para ser procesada y posterior a ello transformada en una señal eléctrica.

- **Nodo:** Reúne la información capturada por el sensor a través de sus puertas de datos. Posterior a ello procede a enviar los datos hacia un equipo en específico donde existe la opción de ser almacenado o a su vez realizar cualquier actividad planificada con dichos datos.

- **Estación base:** Reúne, almacena y expone la información de cada nodo mediante una interfaz gráfica.

- **Gateway:** Elemento que interconecta la red de sensores inalámbrica con otra red de sensores inalámbrica o inclusive con una red externa a la misma.

1.4.2.2 Topología lineal de una red de sensores inalámbricas

La topología de una red se describe como la forma en la que los nodos se encuentran enlazados entre sí. Las redes inalámbricas de sensores pueden utilizar distintos tipos de topologías dependiendo la necesidad de la aplicación. Aunque usualmente se utiliza topologías tipo estrella, malla o árbol, estas presentan un protocolo de enrutamiento complejo a la hora de implementar en estructuras lineales (monitoreo de oleoductos, tuberías de agua y otras aplicaciones futuras que posean una infraestructura lineal a gran escala) [3].

Al aplicar una topología lineal la red es susceptible a ciertos riesgos (baja redundancia, poca robustez, entre otros) por lo que es necesario aplicar diversas técnicas para mitigar

los fallos. Al manipular la topología podemos reducir los riesgos, siendo esta una de las técnicas de gran utilidad para la topología lineal, como una arquitectura multisalto. Mediante el multisalto podemos asegurar que, en caso de que el nodo siguiente falle, la conexión no se pierda y vaya a un nodo en óptimas condiciones, asegurando así la transmisión exitosa de la información [3]. La topología lineal con arquitectura multisalto tiene dos estructuras principales: topología de un nivel (plana) o topología de varios niveles (jerárquica) [11]. Podemos observar gráficamente en la Figura 1.1 que existe una topología lineal con multisalto, la cual puede ser armada de manera simple para un uso básico donde solamente sea necesario enviar datos a través de nodos y evitar que se pierda la información generando saltos si alguno de ellos falla, o a su vez realizarlo más complejo, donde prima la seguridad y robustez de la red siendo controlado a su vez por nodos de mayor rango jerárquico. Cada nodo tiene su rango donde alcanzará el salto máximo que este puede brindar.

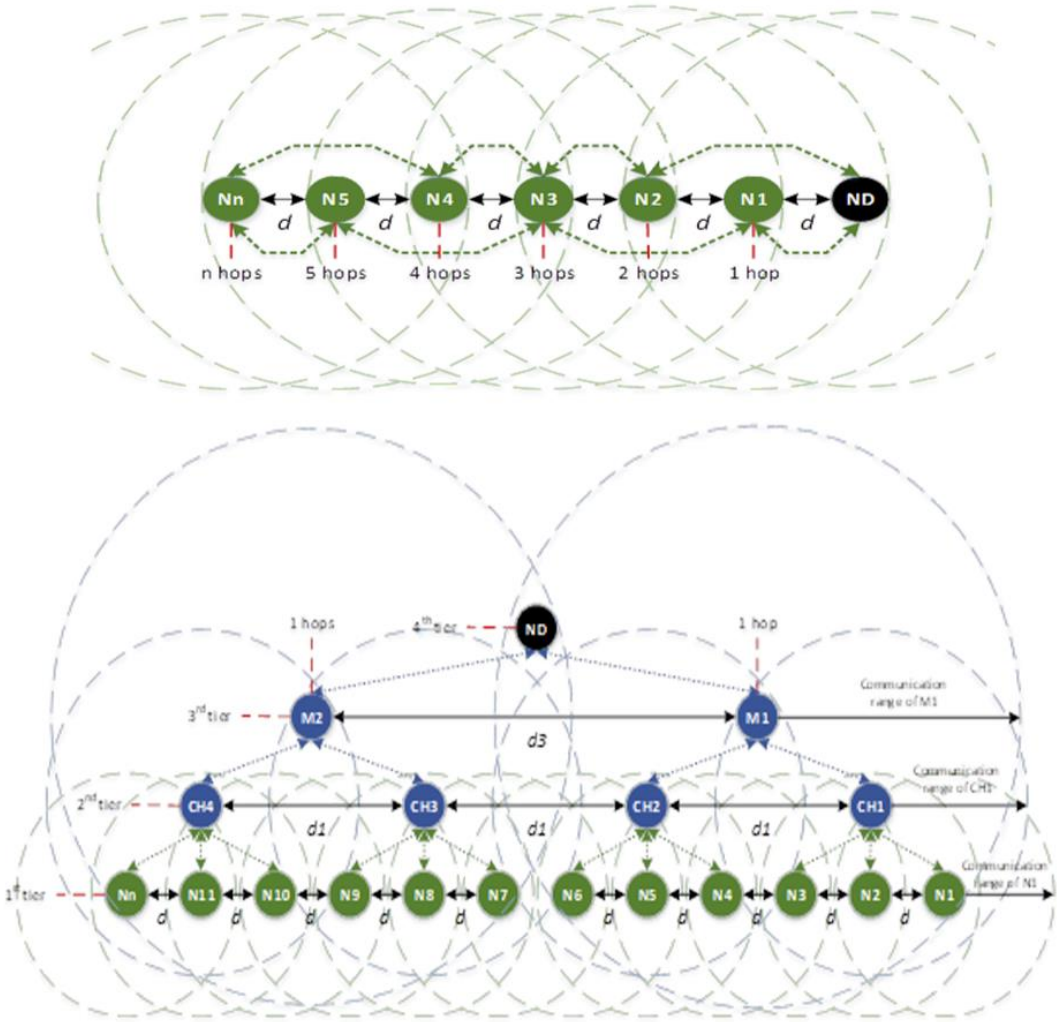

Figura 1.1. Topología lineal plana (superior) y topología lineal jerárquica (inferior) [11]

1.4.3 REDES LORA

LoRa es una tecnología inalámbrica catalogada como una transmisión de largo alcance y un consumo de energía bajo que trabaja en la capa física (PHY), como se puede observar en la Figura 1.3. Esta tecnología se basa en la modulación Chirp Spread Spectrum (CSS), con lo que es posible transmitir datos hasta 15 [km] (con línea de vista). Además, el Spreading Factor (SF) es ortogonal lo que ayuda a mitigar la interferencia entre señales emitidas por un mismo canal y en un mismo tiempo [12].

La capacidad de LoRa permite escoger un SF distinto para la transmisión de datos. Es decir, puede modificar los niveles de energía y la velocidad de transmisión de datos, siendo esto una optimización de la vida útil para el dispositivo. Para entender de mejor manera, el nodo más cercano transmitirá con un SF bajo porque la señal no debe viajar mucho, mientras que un nodo lejano necesitará un SF alto para que la señal llegue a su destino de manera satisfactoria [12].

LoRa tiene distintas configuraciones, como el ancho de banda o los canales, dependiendo la región donde se implemente. Por ejemplo, en América del Norte la configuración debe estar centrado en tener 64 canales, cada uno separados 200 [kHz] desde su frecuencia central, con canales de enlace ascendente empezando por los 125 [kHz], además de 8 canales de enlace ascendentes y descendentes de 500 [kHz], posibilitando así una negociación entre la velocidad de datos y la sensibilidad de esta [13].

1.4.3.1 Propiedades de una red LoRa

Al usar esta tecnología, se debe tener en conocimiento las propiedades que brinda para una mejor experiencia [14].

- Consumo de potencia mínimo
- Alta robustez
- Cobertura de largo alcance
- Resistente a distintos efectos que afectan la transmisión (efecto Doppler, mutitrayecto o desvanecimiento de la señal)
- Escalable

1.4.3.2 Estructura de la trama LoRa

La trama LoRa consta de distintos bloques, los cuales tienen datos distintos para la transmisión de datos. Además, al final tiene un bloque que ayuda a la detección de errores (Cyclic Redundancy Check o CRC) [14].

El bloque del preámbulo y la cabecera es codificado a con una tasa de código 4/8, la cual es fija. El bloque de payload y CRC en cambio puede ser variable y opcional, dependiendo esta de la tasa de código que sea escogida. En la Figura 1.2 podemos observar la estructura de la trama [15].

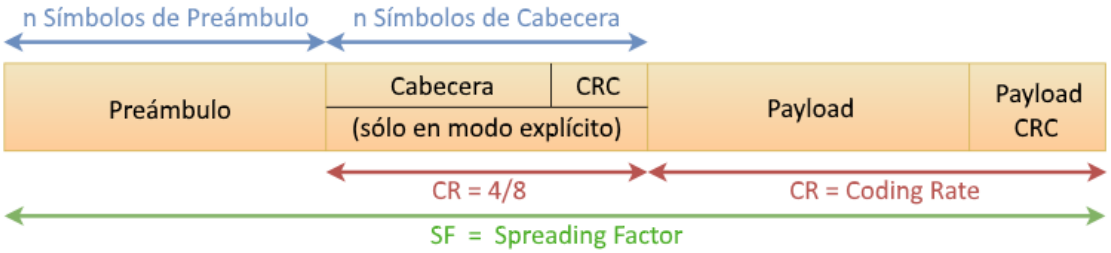

Figura 1.2. Estructura de la trama LoRa [15]

1.4.4 PROTOCOLO LORAWAN

El protocolo LoRaWAN, estandarizado por LoRa-Alliance, es un protocolo de código abierto que brinda servicios de movilidad, localización, comunicación bidireccional y seguridad. Como se puede observar en la Figura 1.3., la capa de LoRaWAN (MAC) se ubica en la parte superior de la capa de LoRa (PHY), definiendo así ciertas características fundamentales como los comandos MAC, el tipo de clase, entre otros. En general, se puede decir que define el procesamiento que el servidor de red va a realizar [16].

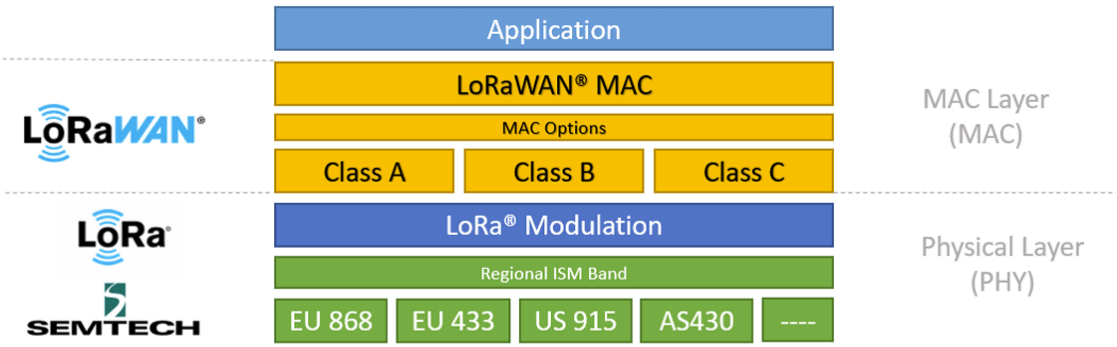

Figura 1.3. Capas de LoRa y LoRaWAN [14]

1.4.4.1 Clases de nodos finales LoRaWAN

Los nodos finales pueden comunicarse entre una o diferentes puertas de enlace, los cuales pueden funcionar en una clase de nodo LoRaWAN. Existen tres tipos de clases, siendo la Clase A la principal para la implementación del funcionamiento. Los nodos de Clase B deben ser implementados también con Clase A y B conjuntamente. Finalmente, los nodos de Clase C deben tener Clase A, Clase B y Clase C. Cualquier clase tiene su manera de comunicación entre nodos [16].

- **CLASE A (ALL-DEVICES)**

Este tipo de clase permite la comunicación en ambos sentidos entre nodos y el servidor, es decir, bidireccional entre ambos. El nodo termina la transmisión del enlace ascendente (UL) y en breve se generan dos ventanas para la recepción del enlace descendente (DL), pero para un próximo envío de datos en DL es necesario que primeramente se tenga el envío de datos en UL. Así, el nodo puede entrar en un estado inactivo hasta nuevo aviso [16].

Figura 1.4. Transmisión de la Clase A [14]

La Clase A tiene un consumo de potencia mínimo entre las otras clases, por lo que es útil para aplicaciones que necesiten una respuesta en DL una vez que se haya finalizado la respuesta en UL.

- **CLASE B (BEACON)**

Este tipo de clase permite generar ventanas de recepción en intervalos fijos de tiempo. Aquí el nodo puede tener la opción de habilitar la recepción de datos en DL agregando una ventana de recepción que se encuentre sincronizado con el nodo. Mediante esta habilitación de la ventana se puede iniciar la comunicación sin necesidad de un envío de datos en UL [16].

Esta clase posee una ventana de recepción extra con tiempo fijo, mientras que las anteriores ventanas continúan como en la Clase A, gracias a una señal de alarma, también llamada beacon, proveniente dese el Gateway que sirve para la sincronización con el nodo.

- **CLASE C (CONTINOUS)**

En este tipo de clase los nodos se encuentran activos todo el tiempo, dejando abierta la ventana de recepción en DL hasta que se encuentre un mensaje en UL. De allí se repite el proceso con una nueva ventana abierta. De esta forma es posible recibir los datos en cualquier momento, generando una red con latencia mínima para la comunicación [16].

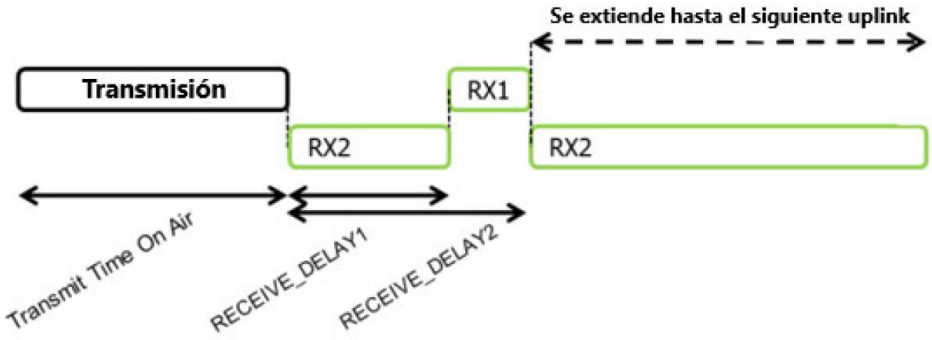

Figura 1.5. Transmisión de la Clase C [17]

2 METODOLOGÍA

Pueden existir cientos de nodos en una topología lineal y con ello tener una asignación dinámica de direcciones, por lo que este afectaría al rendimiento y a su vez no se tendría un orden con el que se pueda controlar a dichos nodos.

En este capítulo se presenta una solución al problema mencionado anteriormente, que es realizar un algoritmo que permita asignar automáticamente las direcciones. Aquí se detalla el proceso en el que se alcanzarán los objetivos especificados en el capítulo antecedente.

Luego de una revisión entre distintos nodos, se selecciona el nodo LoRa-E5 debido a sus ventajas que este presenta en sus características técnicas, como por ejemplo que el módulo de esta placa (STM32WLE5JC) posee una memoria amplia comparado con sus antecesores, lo que le permite trabajar con mayores capacidades sin tener problemas de colgarse, así como sus distintas opciones que tiene para poder ser controlado, cumpliendo con uno de los objetivos específicos de este Trabajo.

Se realiza una breve descripción del nodo LoRa-E5, indicando las características principales del porqué fue seleccionado para la implementación. Seguido de esto se presenta el diseño del algoritmo, el cual nos va a indicar la manera en la que es posible llevar a cabo la solución y la forma en la que debe funcionar el algoritmo, además de un diagrama de flujo para un mayor entendimiento. Finalmente, se emplea el prototipo con dos nodos LoRa-E5, donde se muestran los pasos a seguir, las herramientas necesarias (tanto hardware como software) y la codificación para que este algoritmo sea implementado y con el cual se realizarán las pruebas pertinentes para lograr la meta de este Trabajo de Integración Curricular, que es poder analizar la asignación automática de las direcciones de los nodos en una red inalámbrica LoRaWAN con topología lineal.

2.1 LORA-E5 DEV BOARD

Esta es una placa que puede procesar la tecnología LoRaWAN, la cual utiliza un módulo Wio-E5 STM32WLE5JC (combinación de chip LoRa RF y MCU), haciéndolo perfecto para implementar proyectos pequeños con aplicaciones IoT gracias a sus GPIO que aceptan distintos protocolos e interfaces de datos [18]. Podemos observar la placa en la Figura 2.1.

Esta placa es la primera en la que se logra combinar un chip LoRa RF con un chip MCU en uno solo, alimentado así con un núcleo ARM Cortex-M4 y un chip Semtech SX126X. Gracias a este último chip es posible que la placa sea compatible con el protocolo LoRaWAN, adaptándose a las frecuencias y modulaciones de la tecnología [19].

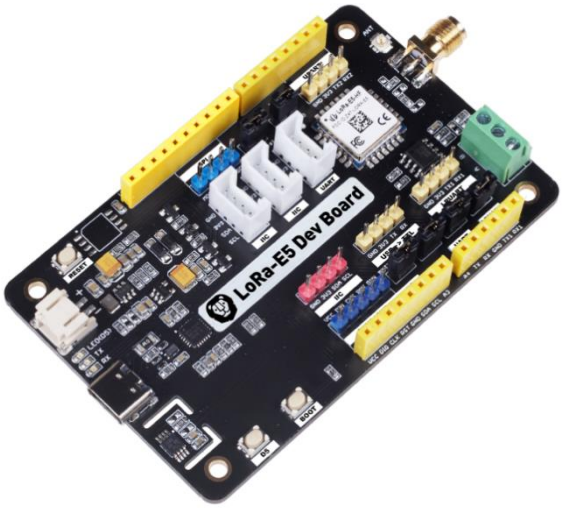

Figura 2.1. LoRa-E5 Dev Board [18]

A continuación, podemos ver las características que este dispositivo presenta, lo cual le hace apto para realizar el trabajo esperado.

Tabla 2.1. Características de LoRa-E5 Dev Board [18]

PARAMETROS	ESPECIFICACIONES
TAMAÑO	85.6 * 54 [mm]
VOLTAJE DE SUMINISTRO	3-5 [V] (Batería externa) 5 [V] (USB tipo C)
VOLTAJE DE SALIDA	3.3 [V] 5 [V]
POTENCIA DE SALIDA	Alrededor de +20.8 [dBm] (a 3.3 [V])
FRECUENCIA	EU868 US915 AU915 AS923 KR920 IN865

PROTOCOLO	LoRaWAN
SENSIBILIDAD	-116.5 [dBm] (SF5) -121.5 [dBm] (SF7) -136 [dBm] (SF12)
INTERFACES	USB tipo C JST2.0 Groove*3 (I2C*2 / UART*1) RS485 SMA-K IPEX
MODULACION	LoRa (G)FSK (G)MSK BPSK
TEMPERATURA	-40 [°C] hasta 85 [°C]

Existen distintas maneras de controlar este nodo gracias a su chip con unidad de microcontrolador. Entre ellas las más comunes son [20]:

- Mediante comandos AT
- Conectando el nodo a otra placa vía UART (Universal Asynchronous Receiver-Transmitter)
- Conectando directamente el nodo y usando una aplicación con SDK (Software Development Kit)
- Conectando a un programador vía SWD (Serial Wire Debug)

Dependiendo de la capacidad del usuario se pueden utilizar las diferentes formas para obtener el mejor resultado posible. Para este trabajo se utilizan dos formas de control (una de ellas consta de dos métodos). La primera será usando vía SWD donde se configurará

el nodo para que pueda asignar la dirección. La segunda será mediante conexión UART para poder conocer los resultados obtenidos con los comandos AT disponibles de la placa LoRa-E5. Alguno de los comandos AT que son de utilidad y que se usará posteriormente en este trabajo son los siguientes [21]:

- AT+ID=DevAddr
- AT+POWER
- AT+MODE
- AT+CLASS

2.1.1 ELEMENTOS PARA CONFIGURACIÓN DEL NODO LORA-E5

El principal dispositivo es el nodo LoRa-E5, pero este a su vez necesita de otros elementos para poder implementar la solución planteada. Para ello se necesita el siguiente hardware:

- LoRa-E5 Dev Board
- Computador
- ST-Link V2
- Arduino Uno
- Cables de conexión macho/hembra

Los elementos principales, como el programador ST-Link V2 y la placa Arduino Uno, serán explicados en la Sección 2.3 de este capítulo.

2.2 DISEÑO DEL ALGORITMO

Una vez encendido el nodo LoRa-E5, hay que tener en cuenta que por default el nodo proporciona una dirección predeterminada, llamada DevAddr. La trama de la dirección del nodo está formada por 32 bits (4 octetos) con los que se puede identificar al nodo en la red (ID). Para mayor entendimiento estas tramas se expresan en hexadecimal.

DevAddr Range	Operator
00000000 - 01FFFFFF	Private/experimental nodes
02000000 - 03FFFFFF	Private/experimental nodes

Figura 2.2. Rangos de operación (en hexadecimal) para los ID de nodos LoRaWAN [22]

Como se aprecia en la Figura 2.3, se empieza encendiendo los nodos LoRa que se encuentren a disposición, hay que tener en cuenta que previo a ello se debe asignar cuál será el primer nodo coordinador. Mediante comandos AT también se puede configurar y así para entregar el ID igual a 1 a dicho nodo. Seguido de eso el nodo empieza una transmisión para conocer los nodos que se encuentren en su rango y luego recibe las peticiones de cada nodo, guardando así en un arreglo el ID y el nivel de RSSI de cada mensaje. Dependiendo el nivel de RSSI se asignará el ID, es decir, mientras mayor sea la potencia con la que llegue el mensaje quiere decir que dicho nodo se encuentra más cercano al nodo coordinador, por lo que a ese nodo se le asignará el ID igual a 2 y así seguirá sucesivamente hasta que se le asigne al nodo con RSSI más bajo (nodo más lejano). Al último nodo se lo puede convertir en coordinador para que nuevamente repita las acciones de asignar direcciones a los siguientes nodos.

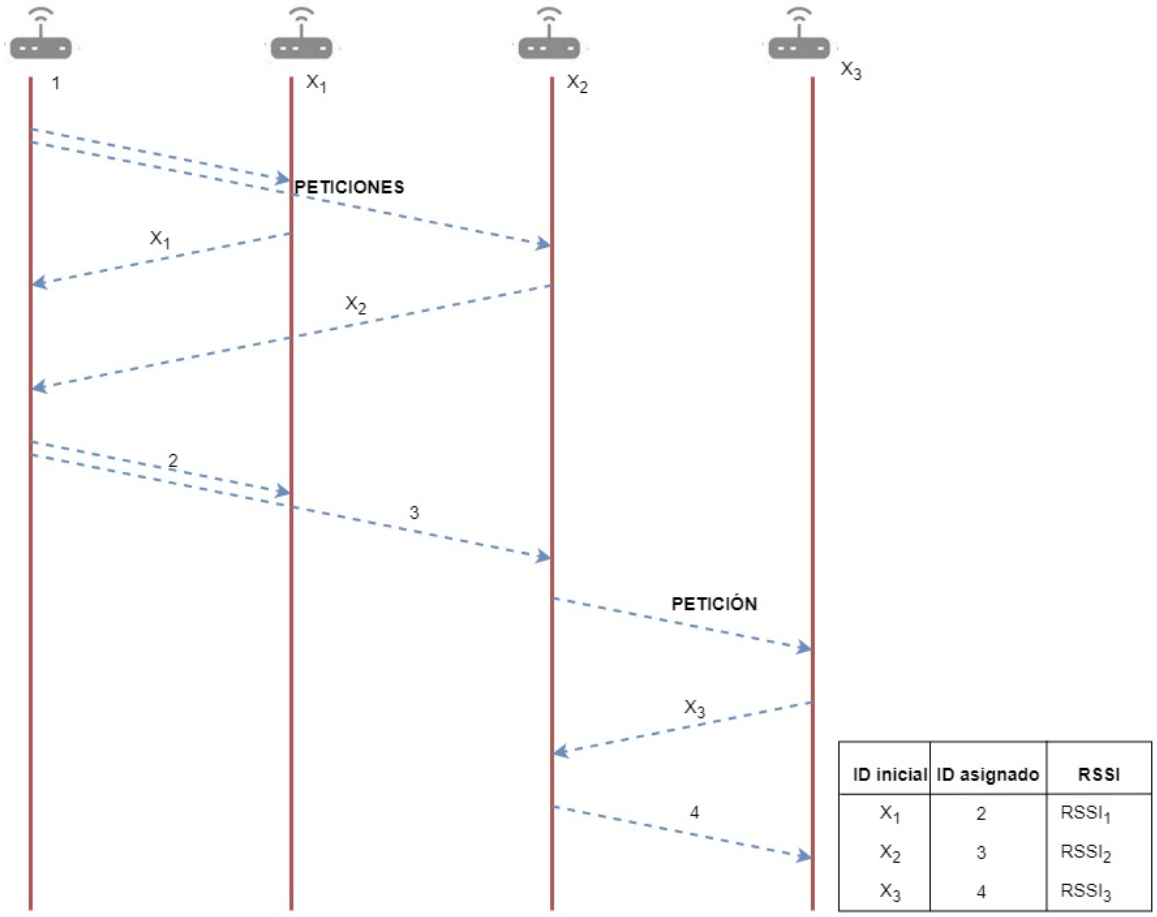

Figura 2.3. Diagrama de la asignación de direcciones

2.2.1 DIAGRAMA DE FLUJO DEL ALGORITMO

Una vez explicado la manera en la que se asignan los identificadores a los nodos de manera automática, se presenta un diagrama de flujo donde se entiende los pasos que se

seguirán para conseguir la asignación automática de direcciones. Aquí podemos observar que, el nodo coordinador asigna direcciones a los nodos que se encuentren dentro del rango, y en caso de que ya no logre encontrar más, se repetirá todo el proceso con un nuevo nodo coordinador. Esto es para abarcar una mayor distancia con distintos nodos hasta que se finalice la topología.

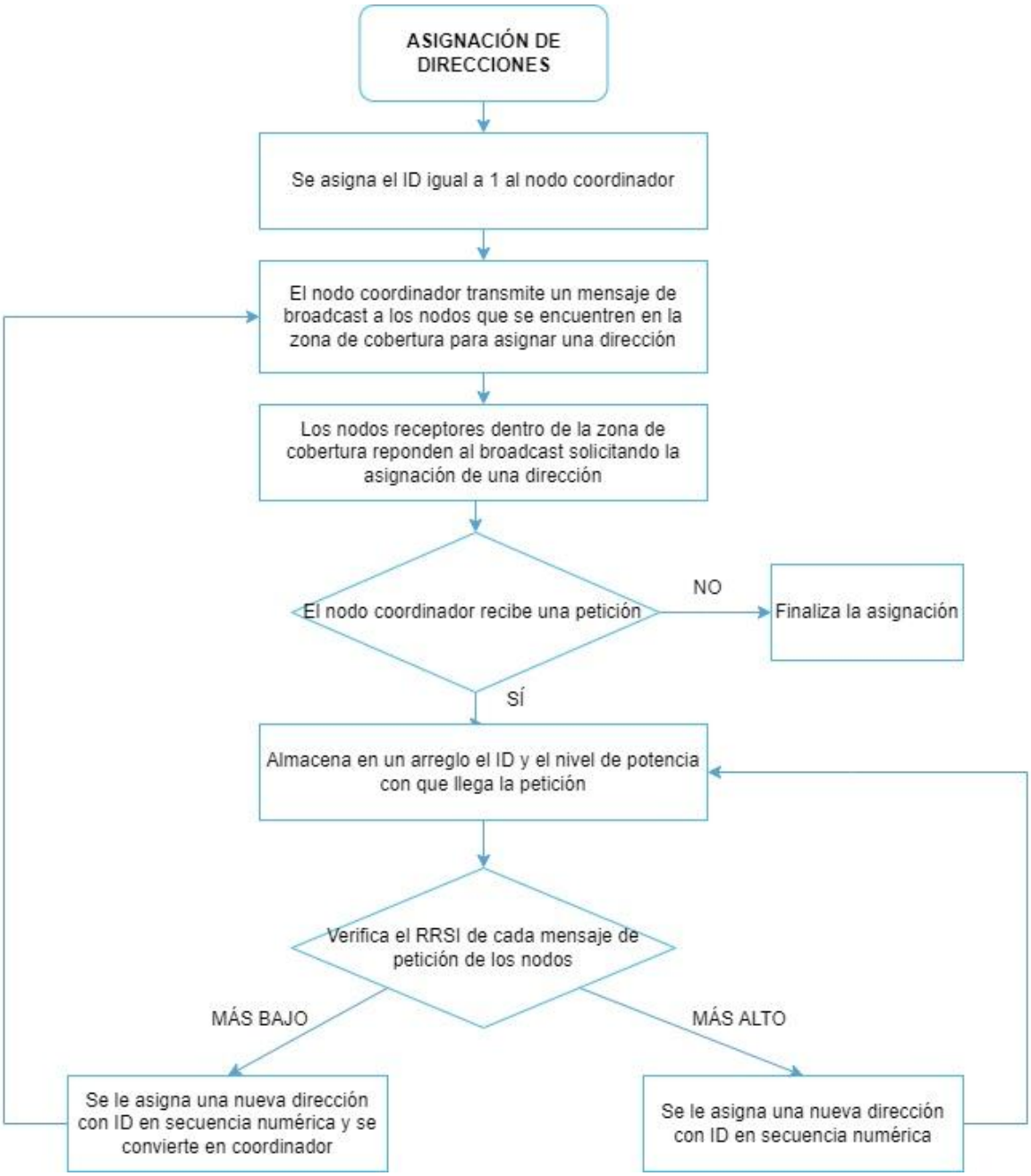

Figura 2.4. Diagrama de flujos para la asignación de direcciones

2.2.2 CODIFICACIÓN DEL ALGORITMO

A continuación, se codifica el diagrama de flujo presentado en la sección anterior. Para ello, se realizan dos códigos, uno que debe ser implementado en el nodo coordinador y el otro que debe ser implementado en el nodo receptor.

En ambos códigos se usan las mismas librerías y se definen los mismos parámetros para que puedan conectarse entre sí.

```
#include <LoRa.h>
#include <LoRaNode.h>
#include <SPI.h>

unsigned long tiempo_espera;

#define ClassA "A"
#define Tx_Power 14
#define Tx_Preamble 8
#define BW 125000
```

Código 2.1. Definición de parámetros

Podemos observar que se tiene librerías necesarias para una transmisión mediante LoRa, lo cual es fundamental para facilitar la codificación. Además, los parámetros elegidos son apropiados para la transmisión de datos debido a que no es necesario un mayor consumo de recursos. Se puede destacar que se define la Clase A para que tengamos primero una comunicación ascendente y luego de que se finalice se empieza con la comunicación descendente. Otra configuración necesaria es la del ancho de banda (BW) ya que el nodo LoRa-E5 solamente soporta tres capacidades diferentes, siendo 125[kHz], 250[kHz] o 500[kHz], por lo que se escoge la de menor capacidad ya que con ello es suficiente. Otro parámetro que se debe tomar muy en cuenta es la longitud del preámbulo que debe tener un preámbulo de recepción mayor al preámbulo de transmisión.

```
void setup() {
  Serial.begin(115200);
  if (!nodo_coordinador.begin(US915)) {
    Serial.println("Falla de encendido del nodo coordinador");
  }

  Serial.println("Encendido del nodo coordinador");
  lora_e5.setClass(ClassA);
  lora_e5.setTxPower(Tx_Power);
  lora_e5.setSignalBandwidth(BW);
  lora_e5.setPreambleLength(Tx_Preamble);
```

Código 2.2. Inicialización del nodo

En la sección de SETUP se inicializa el nodo y se configura los parámetros anteriormente definidos, cada uno con su atributo.

2.2.2.1 Codificación para la asignación de dirección del nodo coordinador

En este apartado se brinda una breve explicación de lo fundamental del código para que el nodo coordinador pueda asignar automáticamente una dirección hacia otro nodo que se encuentre en la zona.

```
int connected=nodo_coordinador.joinABP(devAddr);
```

Código 2.3. Modo del nodo

Seguimos en la función SETUP, donde podemos observar que estamos configurando al nodo coordinador con el modo ABP (Activation By Personalization), el cual permite almacenar la dirección (DevAddr) y las claves de sesión (NwkSKey y AppSKey) directamente en el nodo. Aquí asignamos directamente la dirección al nodo coordinador. En este modo no es necesario un proceso de emparejamiento para que el nodo adquiera la información para enviar los paquetes o mensajes dentro de una red LoRaWAN.

```
nodo_coordinador.beginPacket();
nodo_coordinador.send(mensaje_broadcast);
error=nodo_coordinador.endPacket(true);
```

Código 2.4. Envío de broadcast

Ya en la sección LOOP se empieza la transmisión y asignación de direcciones automática, donde el coordinador envía un mensaje a los nodos que se encuentren en la zona de cobertura.

```
while (nodo_coordinador.available()) {
  uint8_t recibido[]=nodo_coordinador.read();
  nodo_coordinador.recvMSG(recibido, 3000, &from)
  Serial.print("Peticion de: 0x");
  Serial.println(lora_e5.from, HEX);
  Serial.print("RSSI: ");
  Serial.println(lora_e5.lastRssi(), DEC);
```

Código 2.5. Recibo de petición

Luego, una vez enviado el mensaje de broadcast se espera un momento y empiezan a llegar las peticiones de los nodos que se encuentran en la cobertura, donde se podrá observar la dirección que posee actualmente (en hexadecimal) y la potencia con la que se recibe la petición.

```
nodo_coordinador.beginPacket(from);
nodo_coordinador.send(ID);
error=nodo_coordinador.endPacket(true);
```

Código 2.6. Envío de la nueva dirección

Posterior a ello se asigna una nueva dirección secuencial al nodo que va solicitando, todo esto hasta que ya no se tengan más peticiones.

```
nodo_coordinador.beginPacket(from);
nodo_coordinador.send("Se asigna nuevo nodo coordinador");
nodo_coordinador.send(coord_new);
nodo_coordinador.endPacket(true);
```

Código 2.7. Asignación de nuevo coordinador

Finalmente, al último nodo que se le asignó una dirección se envía un mensaje informando que va a ser el nuevo coordinador.

El código completo se lo puede observar en la sección de ANEXOS, como Anexo I.

2.2.2.2 Codificación para la asignación de la nueva dirección del nodo receptor

En este apartado se brinda una breve explicación de lo fundamental del código para que el nodo pueda recibir y tener asignado automáticamente una dirección que viene del nodo coordinador y, de ser el caso, convertirse en el nuevo coordinador para que los demás nodos que no estuvieron en la primera zona de cobertura también puedan tener una asignación automática de direcciones.

```
uint8_t recibido[]=nodo.read();
uint8_t tamano=sizeof(recibido);
nodo.recvMSG(recibido,&tamano, 3000, &from);
if (tamano==48){
  Serial.print("Broadcast de: 0x");
  Serial.println(lora_e5.from, HEX);
```

Código 2.8. Recepción de broadcast desde el nodo coordinador

Una vez encendido el nodo, este procede a recibir el mensaje de broadcast enviado desde el nodo coordinador. Para que lo recepte primero se compara el tamaño del mensaje, puesto que se puede confundir con la petición de algún otro nodo.

```
nodo.beginPacket(from);
nodo.send("Peticion para asignación");
error=nodo.endPacket(true);
```

Código 2.9. Envío de petición

Al confirmar que se trata del mensaje de broadcast indicando que se asignará una nueva dirección, el nodo procede a enviar un mensaje de petición solicitando la asignación.

```
int nuevo ID=nodo.read();
lora_e5.setDevAddr(nuevo_ID, HEX);
```

Código 2.10. Envío de petición

Finalmente, el nodo lee la nueva dirección enviada por el coordinador para así poder configurarlo con el atributo correspondiente.

```
int coordinador=nodo.read();
if(coordinador==nuevo_ID){
```

Código 2.11. Nuevo coordinador

En caso de que el nodo vaya a ser el nuevo coordinador, se hace una comparación entre la dirección que el coordinador envía y la dirección actual del nodo. Una vez comprobado que sea el mismo nodo, entonces se procede a utilizar el código anteriormente explicado.

El código completo se lo puede observar en la sección de ANEXOS, como Anexo II.

2.3 HARDWARE PARA LA IMPLEMENTACIÓN

2.3.1 PROGRAMADOR DE MICROCONTROLADORES

El programador de microcontroladores es un dispositivo con un software, mediante el cual es posible traducir los lenguajes del código (C, C++, Java, Ensamblador, entre otros) a un lenguaje de máquina para que sea quemado en el microcontrolador desde el computador y lo guarda en un archivo hexadecimal. Dicho programador hace el papel de una interfaz entre el computador y el microcontrolador [23]. Un esquema simple se puede observar en la Figura 2.5.

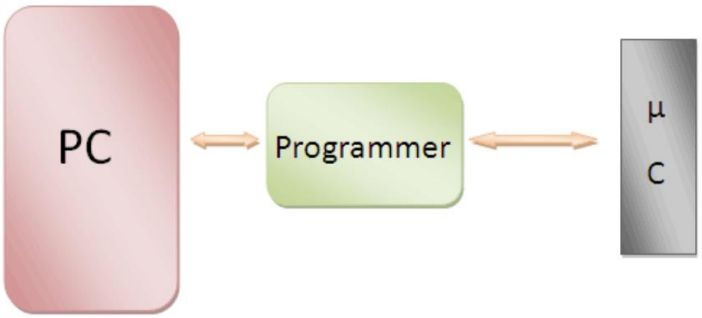

Figura 2.5. Diagrama de bloques de un programador de microcontroladores [23]

2.3.1.1 ST-Link V2

El ST-Link V2 es un dispositivo programador para microcontroladores tipo MCU (MicroController Unit) adecuado para conexiones con módulos STM8 y STM32, el cual se conecta al computador mediante interfaz USB y a las placas o tarjetas de desarrollo mediante conectores I2C “macho” [24].

Existen dos versiones de este dispositivo, siendo el de la Figura 2.6 el que se usó en este trabajo, donde se encuentra explicado los pines de este para su conexión.

Figura 2.6. ST-Link V2 y sus pines de conexión [24]

2.3.2 ARDUINO UNO

La placa de Arduino UNO es un microcontrolador basado en el módulo ATmega32P con entradas/salidas digitales y analógicas. Al ser de código abierto, este puede ser programado directamente, y a su vez servir de puente para controlar otras placas conectadas a ella. Esta placa puede ser utilizada como interfaz de entrada o interfaz de salida, dependiendo las aplicaciones en las que se lo haya configurado [25].

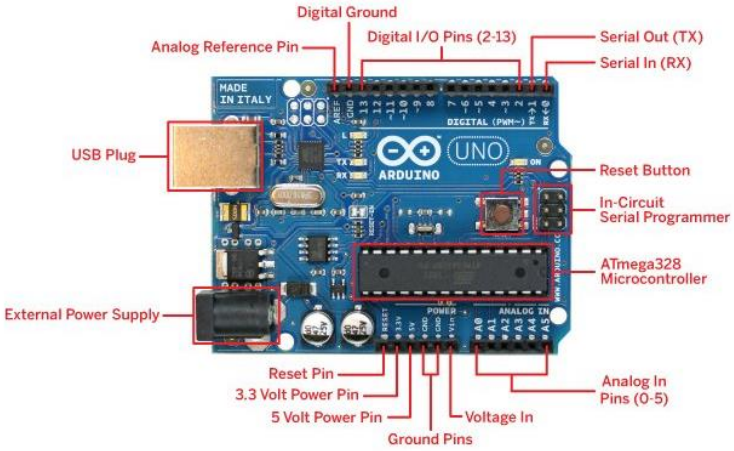

Figura 2.7. Arduino Uno y sus pines de conexión [25]

2.4 SOFTWARE PARA LA IMPLEMENTACIÓN DEL ALGORITMO

En esta sección se explicará la preparación previa que se debe realizar al software para poder utilizar el IDE de Arduino para quemar el código en la placa, y que se pueda obtener una conexión exitosa entre el nodo LoRa-E5 y el programador ST-Link V2.

2.4.1 ADICIÓN DE TARJETAS STM32 EN ARDUINO IDE

mediante Internet o a su vez desde Microsoft Store. En este caso se optó por utilizar el programa de Microsoft Store, el cual nos proporcionó la versión 1.8.57.0, que de momento no consume mucho almacenamiento en la PC.

Figura 2.8. Programa Arduino IDE

En dicho IDE solo se encuentran los entornos de las placas Arduino, por lo que se debe añadir un entorno que contenga el desarrollo Arduino STM32 con soporte para LoRa-E5. Para ello lo primero que se debe hacer es ir a la pestaña de Archivo, ubicado en la barra superior, y elegir la opción Preferencias, como indica la Figura 2.9.

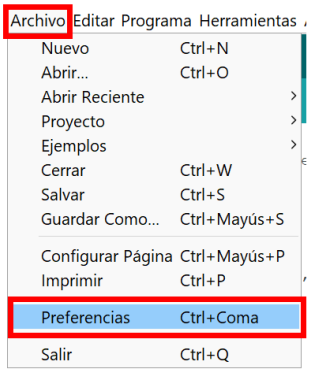

Figura 2.9. Menú Archivo de Arduino IDE

Una vez seleccionado la opción Preferencias se despliega una nueva ventana, en la cual se debe ingresar la URL del gestor de tarjetas adicional. Si se desean agregar varios se debe aplastar en el cuadro de la derecha y en cada línea nueva poner un URL distinto. Para nuestro caso se utilizó el siguiente URL:

https://github.com/stm32duino/BoardManagerFiles/raw/main/package_stmicroelectronics_index.json

Como podemos observar en la Figura 2.10, el formato de la URL es .json, lo que quiere decir que es un formato que nos permite estructurar los datos como texto y se puede intercambiar información entre las aplicaciones.

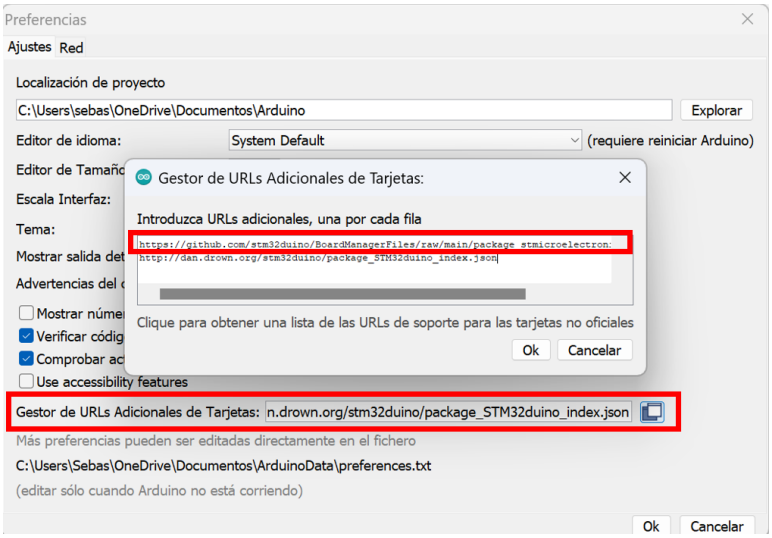

Figura 2.10. Ventana de Preferencias para añadir el gestor de tarjetas

Una vez ingresado la URL procedemos a la descarga de la tarjeta STM32 ingresando a la pestaña de Herramientas, ubicado en la barra superior, y en el menú se elige la opción Placa para luego abrir el Gestor de Tarjetas.

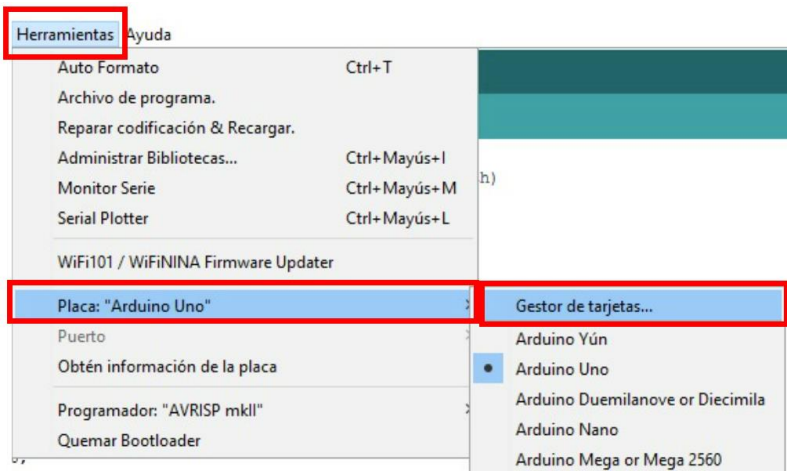

Figura 2.11. Menú Herramientas de Arduino IDE

Allí se despliega una nueva ventana en la cual se debe buscar “STM32 MCU based boards”, implemetado por STMicroelectronics. En la descripción podemos observar las placas STM32 que nos facilita este gestor por lo que verificamos que sí será de utilidad

para nuestro trabajo. Se debe dar click en el botón Instalar y empieza la descarga del gestor.

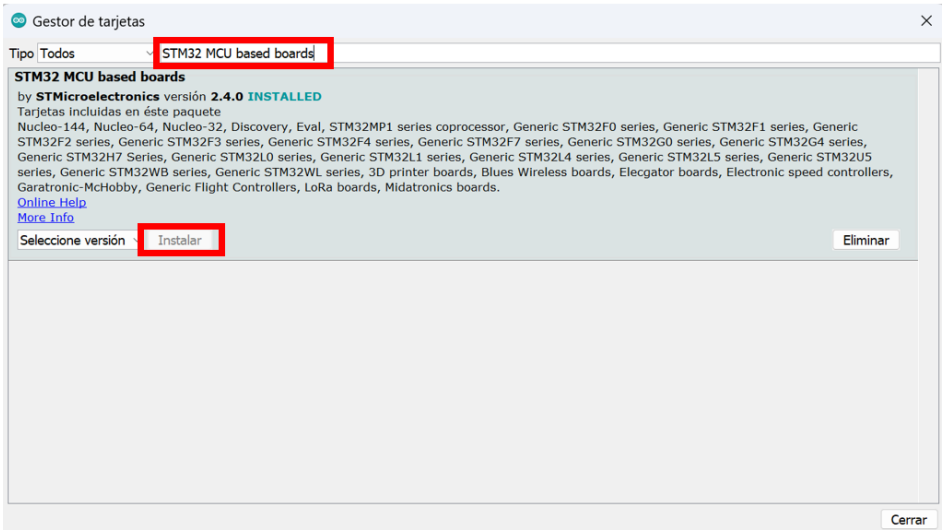

Figura 2.12. Menú Descarga del gestor de tarjetas STM32 MCU Based Boards

Se debe tener en cuenta que este gestor ocupa alrededor de 7 GB debido a que son varias placas importadas, por lo que es muy importante tener ese espacio para que la descarga pueda ser exitosa. En mi caso se logró implementar el gestor con la capacidad en su límite máximo (como se observa en la Figura 2.14) lo cual disminuye el rendimiento del computador. Se debe tener muy en cuenta ese factor antes de utilizar este gestor.

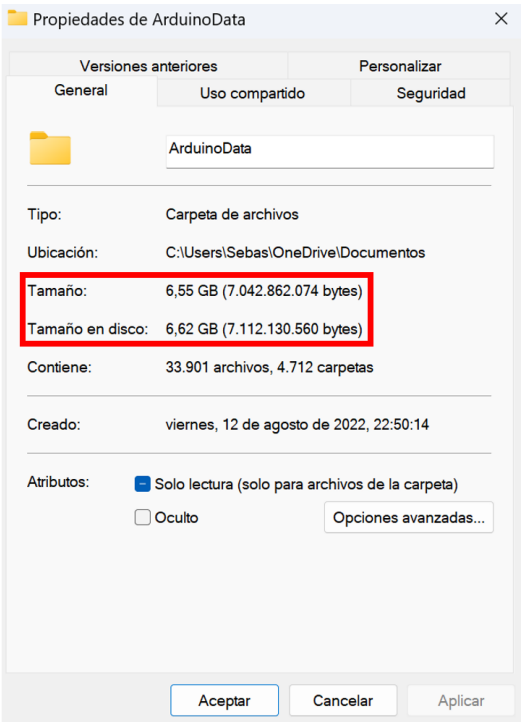

Figura 2.13. Tamaño de uso en la capacidad

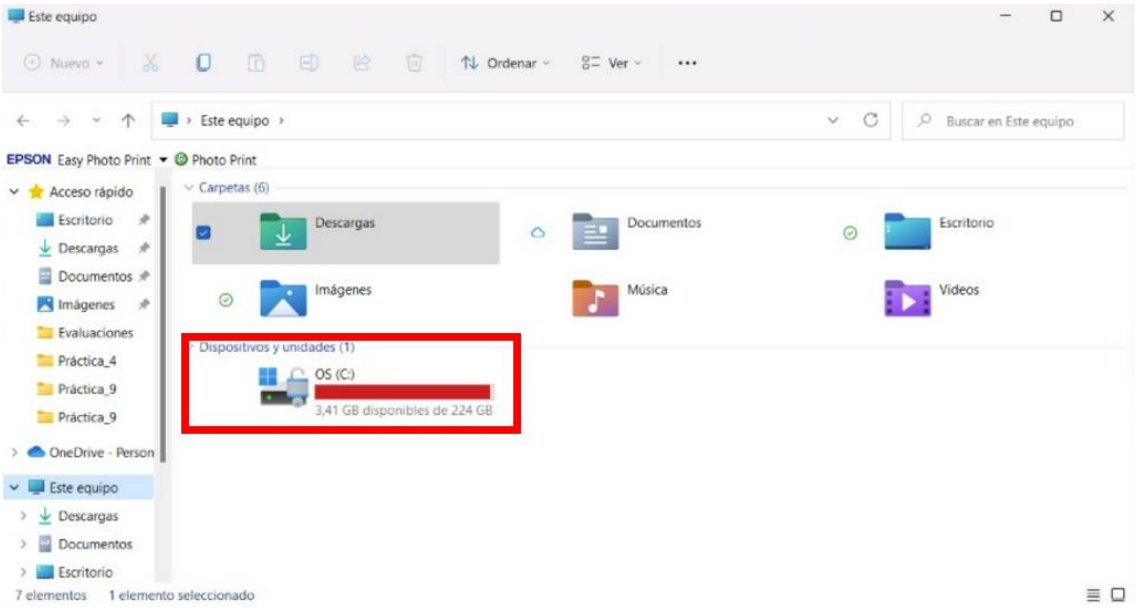

Figura 2.14. Capacidad actual del computador

Ahora que ya se encentra instalado el paquete, podemos utilizar las placas STM32 que deseemos. Existen distintos modelos de STM32 por lo que debemos elegir el correcto. En nuestro caso la placa LoRa-E5 utiliza el módulo STM32WLE5JC. Entonces nos vamos a la opción de Placa y buscamos la serie STM32WL en STM32 board groups, como indica la Figura 2.15.

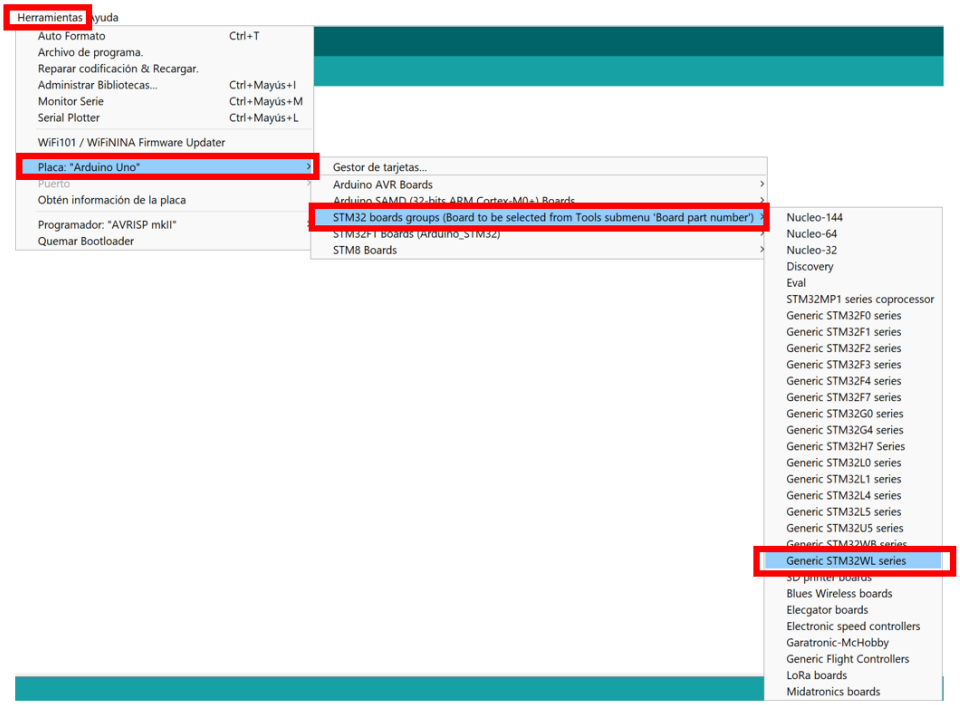

Figura 2.15. Selección de serie para placa STM32

Solo elegimos la serie, por lo que se despliega un menú más extenso donde se puede seleccionar más elementos. Los cambios importantes para nuestro trabajo son el número de placa, seleccionando WLE5JCx, y el método de cargado que va a ser SWD debido a que se está utilizando el ST-Link V2 (conectado al nodo LoRa mediante dicha comunicación). En la Figura 2.16. podemos observar las opciones seleccionadas.

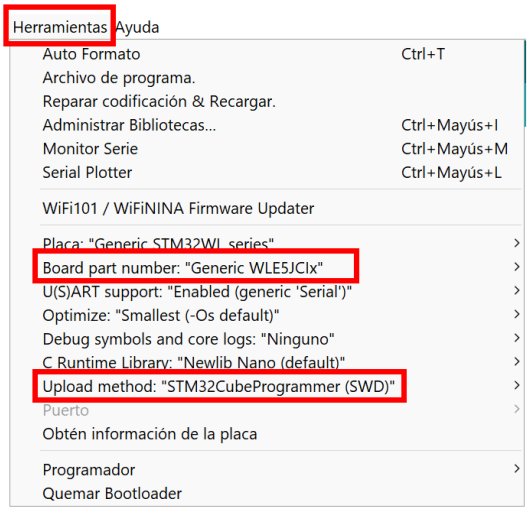

Figura 2.16. Selección de características

Aparte de preparar el software en el IDE, también se deben clonar una arquitectura distribuida de git. Primero, para utilizar los comandos en el terminal de Windows es necesario descargar e instalar la aplicación Git del siguiente link:

http://git-scm.com/download/win

Posterior a ello se debe ir al terminal e ingresar a la carpeta de hardware de Arduino y crear la carpeta stm32, mediante los siguientes comandos:

cd OneDrive/Documentos/Arduino/hardware

mkdir -f stm32

Una vez creado se ingresa a la carpeta stm32 y allí es donde se clona la arquitectura git para el uso del hardware. Se ingresa el siguiente comando:

git clone https://github.com/bertrik/Arduino_Core_STM32

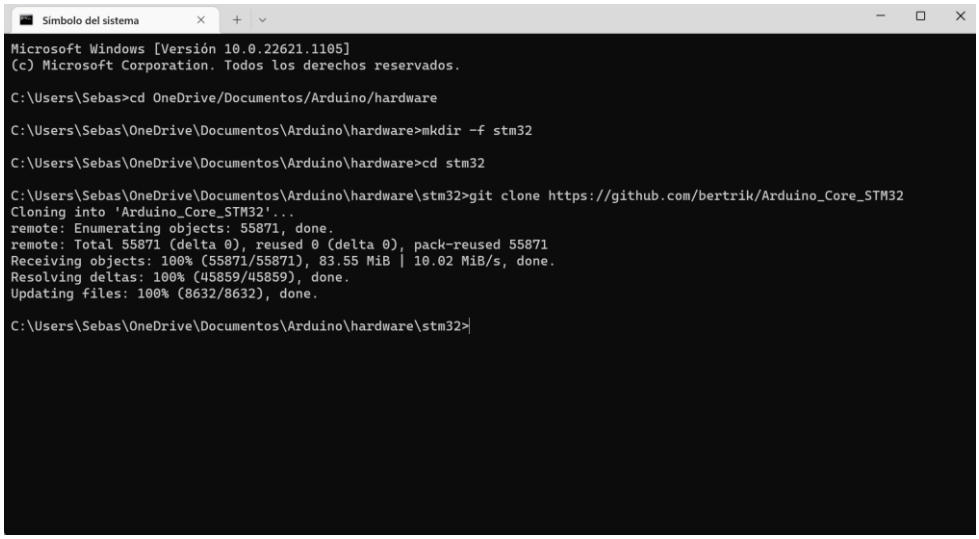

Figura 2.17. Ingreso de comandos para clonar GIT

Una vez realizado todo ese proceso podemos decir que el software de Arduino IDE se encuentra listo para la implementación del código del prototipo.

2.4.2 ACTUALIZACIÓN DE FIRMWARE PARA ST-LINK V2

La primera vez que se conecta el ST-Link V2 y para que opere normalmente se necesita ver si su driver se encuentra actualizado. Por lo que se debe ingresar al Administrador de Dispositivos y verificar que el controlador STM32 STLink no se encuentre con un signo de alerta (como muestra la Figura 2.18). En caso de que se visualice dicho símbolo, solamente es necesario actualizarlo para arreglar el problema, caso contrario no funcionará el programador.

Figura 2.18. Controlador STM32 STLink

Ahora, se debe verificar el firmware del ST-Link V2, por lo que se debe utilizar el programa STM32 ST-LINK Utility y encontrar el actualizador.

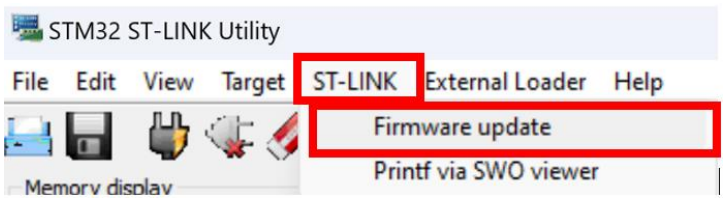

Figura 2.19. Menú ST-LINK de STM32 ST-LINK Utility

Este te redirige a una nueva ventana donde se actualizará a la versión más reciente. En nuestro caso se actualizó de la versión V2.J29.S7 a la versión V2.J40.S7 (ver la Figura 2.20).

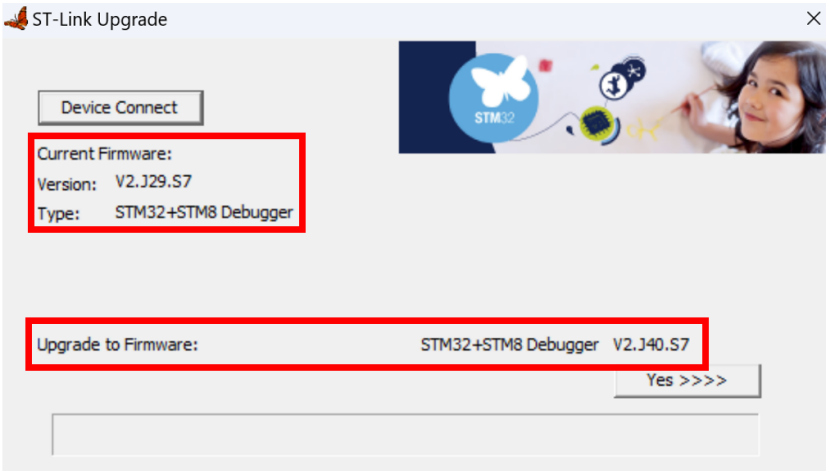

Figura 2.20. Ventana de actualización de Firmware para ST-Link V2

2.4.3 CONFIGURACIÓN INICIAL DEL NODO LORA-E5

Para poder controlar el nodo LoRa-E5 es necesario realizar distintas configuraciones previas con las que se pueda quemar el programa deseado.

Primero se debe conectar el nodo al computador mediante el ST-Link V2 con la conexión de pines de la Figura 2.21.

ST-LINK V2	LoRa-E5
PIN2 - SWDIO	DIO -SWDIO
PIN4 - GND	GND
PIN6 - SWCLK	CLK - SWDLK

Figura 2.21. Conexión de pines entre LoRa-E5 y ST-Link V2 [26]

Seguido a la conexión se debe “forzar” su encendido. Para ello se utiliza nuevamente el programa STM32 ST-LINK Utility. Inicialmente saldrán mensajes de que no es posible conectar el nodo porque no lo encuentra (especificaciones en rojo), por lo que se debe mantener presionado el botón de RESET en el nodo y allí y se da click en el ícono CONNECT TO THE TARGET. Luego se suelta el botón del nodo y en ese momento es cuando el programa puede leer al nodo (especificaciones en azul). Se debe realizar esta operación cada que se encienda y se conecte el nodo.

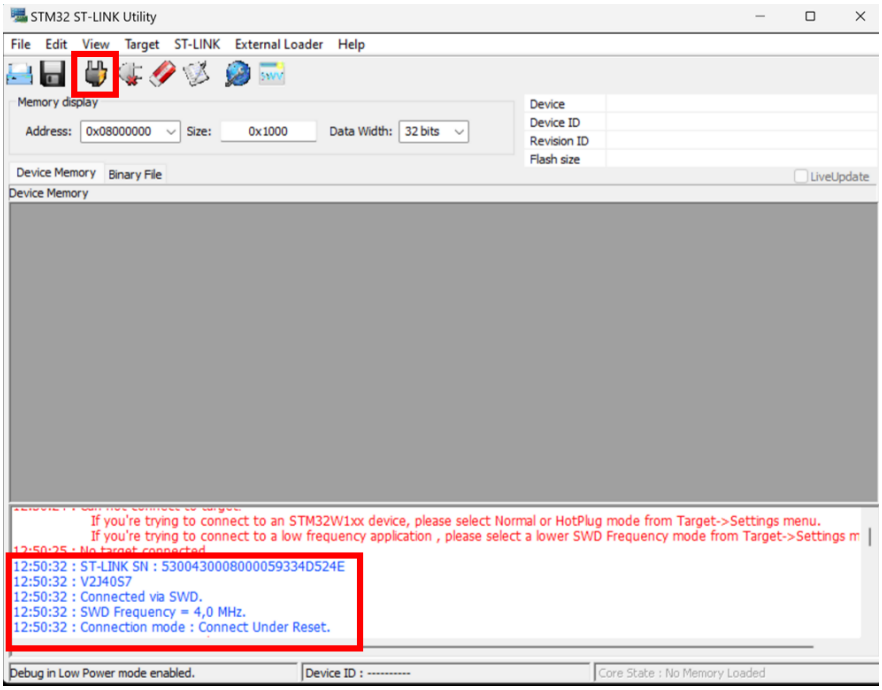

Figura 2.22. Lectura exitosa del nodo con ST-Link V2

También podemos comprobar que efectivamente exista dicha conexión mediante comandos en el terminal de Windows. Se debe ingresar a la carpeta BIN de STM32CubeProgrammer. Una vez dentro se ingresa el comando *STM32_Programmer_CLI.exe* el cual nos muestra la forma en la que se debe utilizar el comando, así como una lista los comandos y argumentos que podemos ingresar y los cambios

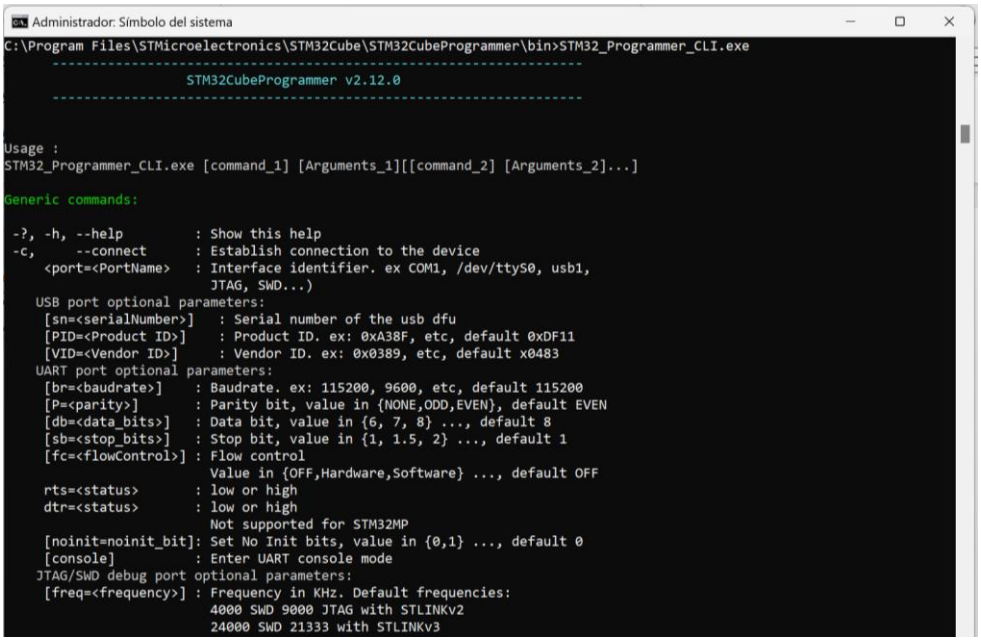

Figura 2.23. Uso del comando *STM32_Programmer_CLI.exe*

Al usar los argumentos *-c port=swd -ob displ* podemos conocer el estado de la conexión.

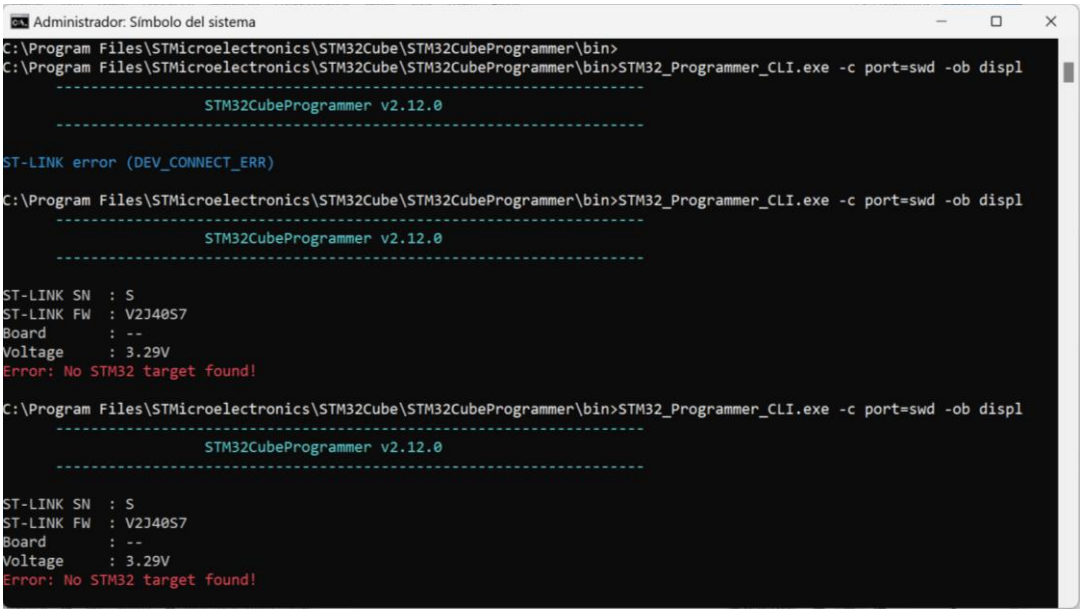

Figura 2.24. Conexión fallida entre nodo y ST-Link V2

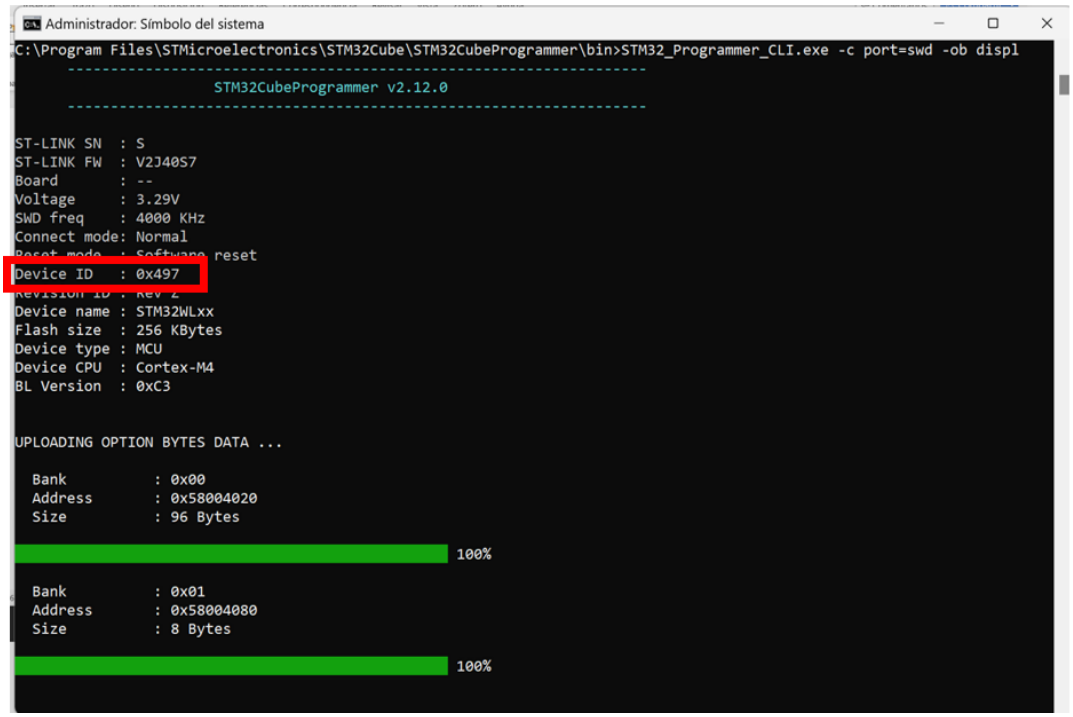

Figura 2.25. Conexión exitosa entre nodo y ST-Link V2

En la Figura 2.24 podemos observar que aún no encuentra la placa, por lo que no brinda información, pero una vez realizada la configuración anterior, volvemos a enviar el mismo comando y ya nos entrega la información vista en la Figura 2.25. Aquí observamos que la dirección del nodo es 0x497, el cual será cambiado más adelante.

Con dicha configuración simplemente nos aseguramos de que se pueda leer el nodo. En caso de que quememos el programa nos saldrá un error mencionando que la placa aún no permite quemarla debido a su seguridad. Esto se debe a que el nodo viene con una protección de lectura, la cual se puede cambiar mediante comandos en el terminal de Windows. Para observar si el nodo se encuentra con protección de lectura se utiliza el comando *STM32_Programmer_CLI.exe -c port=swd -ob displ.*

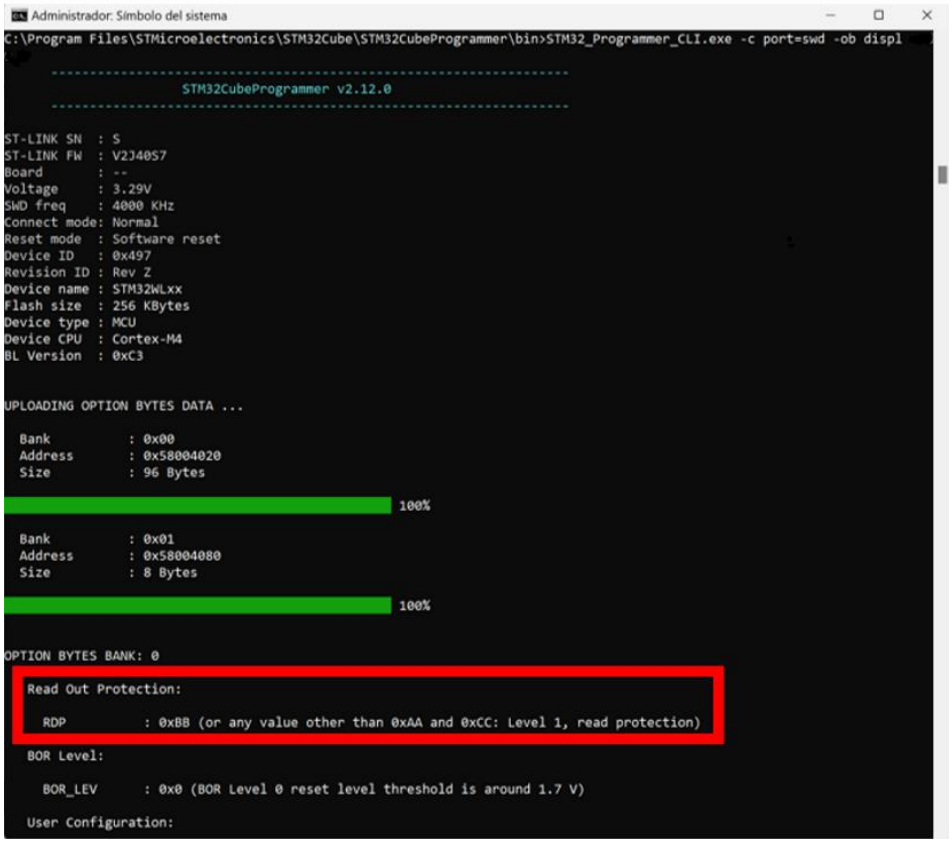

Figura 2.26. Protección de lectura activada

En la Figura 2.26 podemos observar el resultado del comando, donde en el apartado READ OUT PROTECTION es de nivel 1, es decir, aún mantiene su protección. Al hacer uso del comando mencionado en la Figura 2.23 podemos buscar el argumento adecuado que permita cambiar la protección, donde *-rdu* es el argumento adecuado.

```
-rdu,   --readunprotect: Remove memory's Read Protection by shifting the RDP
                         level from level 1 to level 0.
```

Figura 2.27. Argumento para modificar la protección

Al usar el comando *STM32_Programmer_CLI.exe -c port=swd -ob displ -rdu 0* estamos indicándole al nodo que se cambie la protección a nivel 0, es decir, que se desactive la protección de lectura.

Figura 2.28. Desactivación de la protección de lectura

Podemos volver a correr el comando de la Figura 2.26 para poder comprobar que efectivamente se haya desactivado la protección.

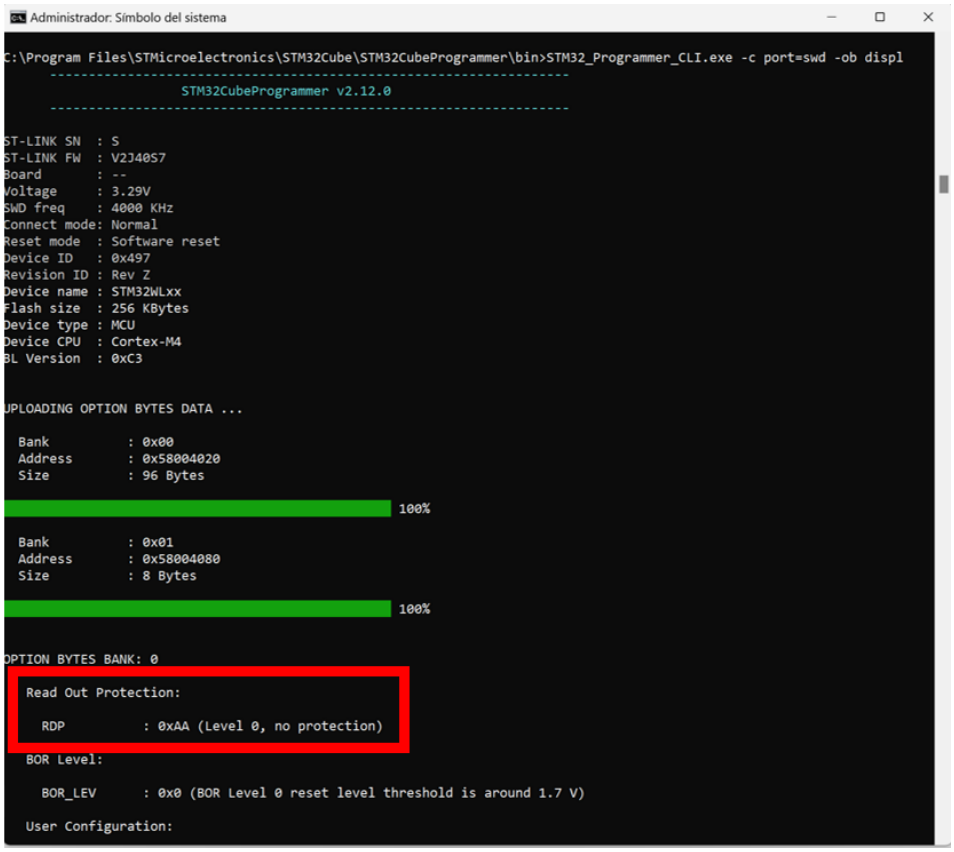

Figura 2.29. Protección de lectura desactivada

2.5 IMPLEMENTACIÓN DEL PROTOTIPO

Mediante el prototipo podremos realizar las pruebas de funcionamiento del algoritmo para la asignación automática. Para la implementación del prototipo disponemos de dos nodos LoRa-E5, los cuales uno actuará como coordinador (con dirección 0x1) y otro nodo será el que se asigne automáticamente (con dirección 0x2) con los códigos listados en Anexo I y Anexo II respectivamente.

Previamente se deben realizar las preparaciones indicadas en secciones anteriores.

Se conecta el nodo LoRa-E5 al ST-Link V2 como se muestra en la Figura 2.21, y allí se conecta al puerto USB del computador. En mi caso estoy utilizando un HUB para tener una mayor cantidad de puertos USB donde conectar, pero hay que tener en cuenta que esto no interfiere con el proceso de la implementación. En la Figura 2.30 podemos observar la conexión.

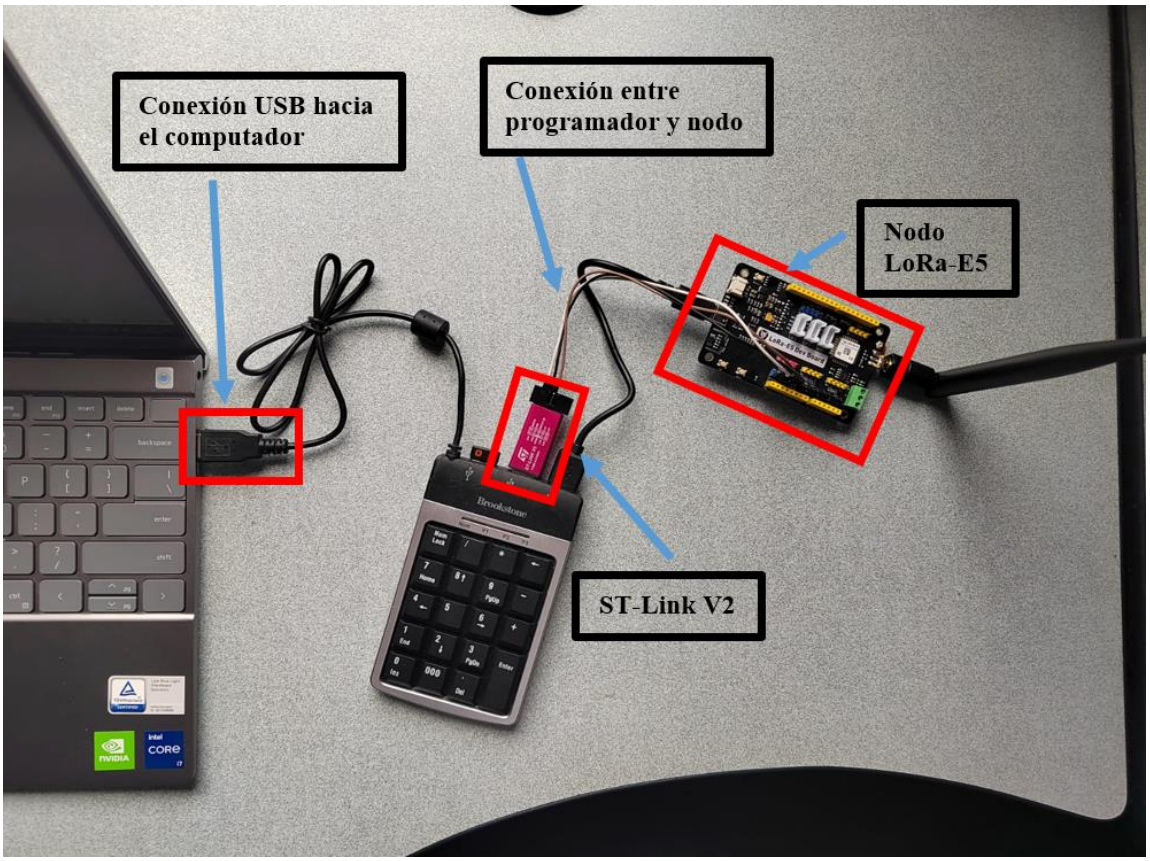

Figura 2.30. Conexión del prototipo

Allí se enciende el LED del ST-Link V2. Inicialmente se enciende con color rojo que quiere decir que no se encuentra en conexión. Se debe realizar la conexión como lo descrito en la Figura 2.21. Una vez realizada la conexión exitosa el LED cambiará su color a azul.

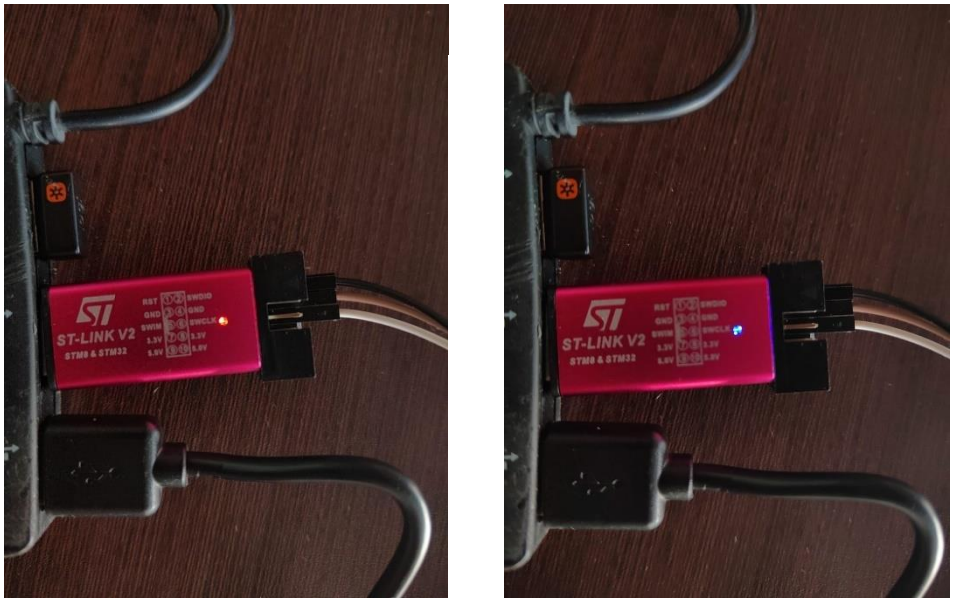

Figura 2.31. Conexión fallida en rojo (izquierda) y conexión exitosa en azul (derecha)

Una vez tengamos el LED azul ya podremos cargar los códigos generados para la asignación de direcciones. En el momento en que el programador se encuentre traduciendo al lenguaje del microcontrolador, este se encontrará parpadeando intermitente entre el color rojo y color azul. En la parte inferior del IDE de Arduino podemos ir viendo los paquetes que se van utilizando, los mismos que fueron cargados al momento de realizar la preparación del IDE.

Al momento en que se sube el programa, como se observa en la Figura 2.32, en la ventana inferior nos sale un mensaje donde se indica que la operación se logró exitosamente. Además, en la parte inferior derecha aparece las características de la placa configuradas inicialmente donde estamos cargando el programa.

Figura 2.32. Programa subido al nodo coordinador

Además, si ampliamos la ventana podemos observar que se está emulando mediante STMCubeProgrammer v2.12.0 y nos brinda la información del nodo conectado en ese momento.

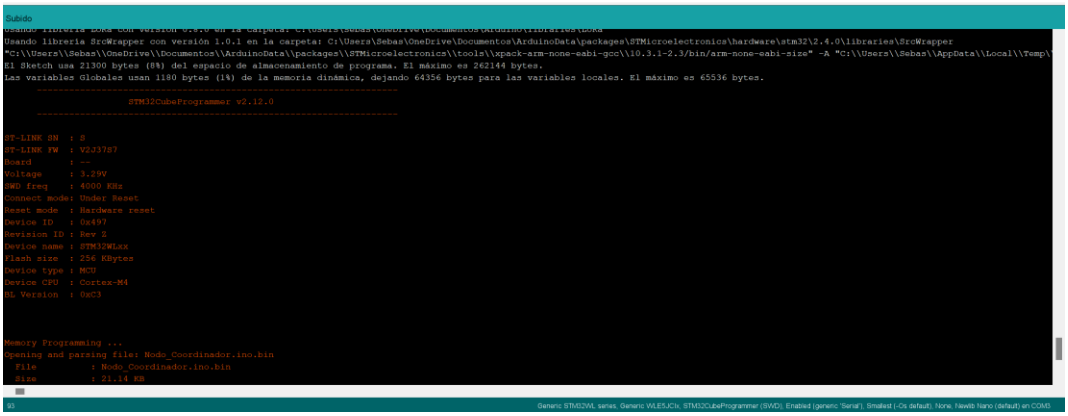

Figura 2.33. Emulación de STMCube Programmer

A continuación, se presenta el prototipo completo, es decir, la conexión del nodo coordinador y la conexión del nodo al cual se asignará la dirección automáticamente.

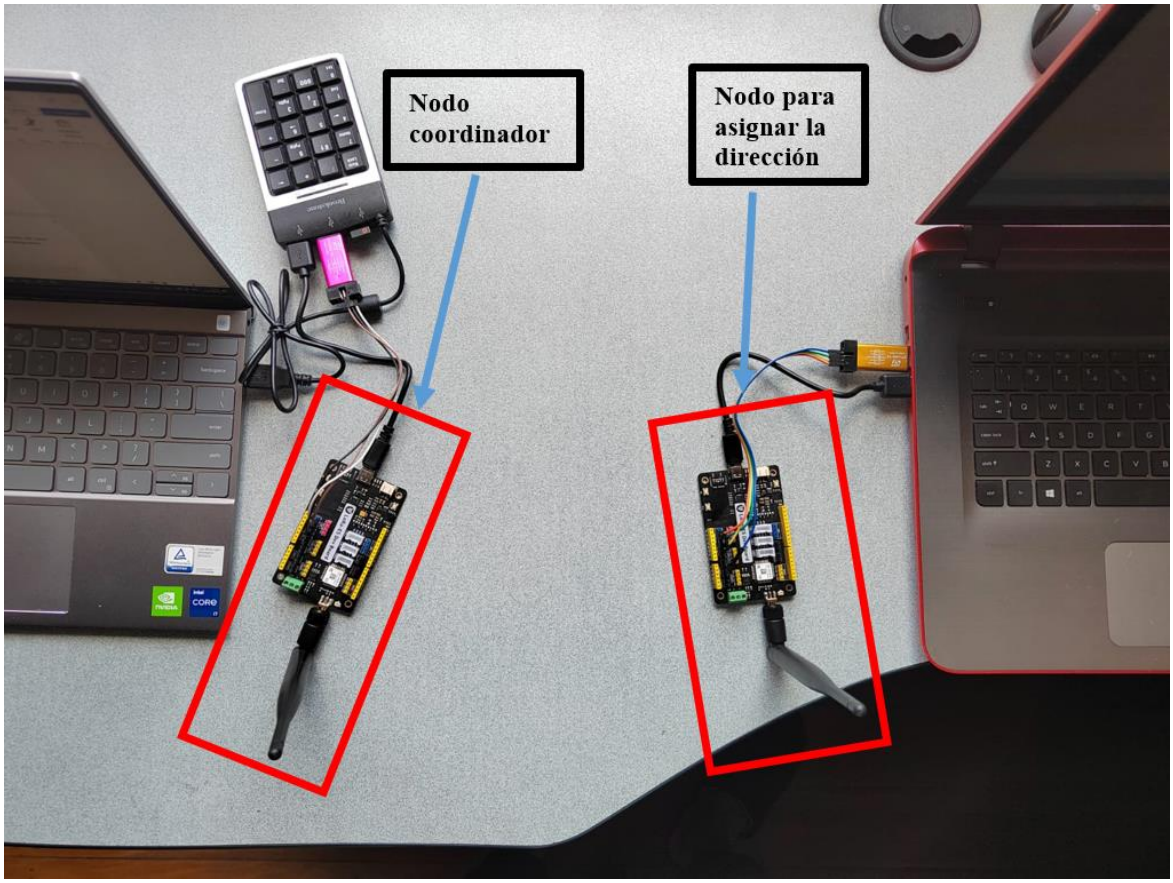

Figura 2.34. Prototipo para la asignación automática

Cabe mencionar que es suficiente implementar el prototipo con dos nodos, donde exista un nodo coordinador y un nodo al que se le asigna la dirección para demostrar que se encuentra en correcto funcionamiento de asignación. Sin embargo, el código del algoritmo se encuentra realizado para que, en caso de que se llegue a aumentar otro nodo, el

coordinador sea capaz de continuar asignando las direcciones automáticamente, y si existe otro nodo fuera del rango, el último nodo asignado se convierta en el nuevo coordinador y pueda continuar asignando direcciones a los nodos posteriores al mismo, cumpliendo así la asignación automática de direcciones en una topología lineal.

En las pruebas se va a realizar dos escenarios, donde no existe ningún nodo al que asignar dentro del rango (solamente se enciende el nodo coordinador) y otro escenario donde el nodo coordinador encuentra a un nodo para asignar la dirección.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En el Capítulo 3 podemos observar los resultados obtenidos en las pruebas al haber implementado nuestro algoritmo generado en un prototipo con nodos LoRa-E5. Para obtener los resultados se usará una conexión UART entre la placa Arduino Uno y el nodo LoRa-E5 que se le asignó una dirección automática. Para ello se usan los comandos AT mediante código.

```
void setup() {
  pinMode(Pin_TX, OUTPUT);
  Serial.begin(115200);
  delay (2000);
  Serial.println("PRUEBAS DE TRABAJO CURRICULAR");
  Encendido();
  delay(3000);
}
```

Código 3.1. Configuración de la placa

Se define al pin conectado como salida para que pueda enviar los comandos AT desde la placa Arduino hacia el nodo LoRa-E5. Además, se configura la velocidad de transmisión del puerto serial a 115,2 [kbps].

```
void Encendido(){
  uint8_t respuesta=0;
  respuesta = Envio_ComandoAT("AT", 2000);
  if (respuesta == 0) {
    digitalWrite(Pin_TX,HIGH);
    delay(3000);
    digitalWrite(Pin_TX,LOW);
    while(respuesta == 0){
      respuesta = Envio_ComandoAT("AT", 2000);
    }
  }
```

Código 3.2. Función ENCENDIDO

La función ENCENDIDO utiliza el comando AT, donde debe dar una respuesta de OK al momento en que esta sea ingresada. Con esto nos aseguramos de que el nodo pueda obtener respuestas para cualquier comando AT. En caso de que no se tenga una respuesta se espera 2 segundos y posterior a ello vuelve a intentar enviar el comando.

```
int8_t Envio_ComandoAT(char* ComandoAT, unsigned int Tiempo_agotado) {
  uint8_t x=0, respuesta=0;
  char response[100];
  unsigned long anterior;
  memset(response, '\0', 100); // Inicializa el string
  delay(100);
  // Limpia el buffer de entrada
  while( Serial.available() > 0) Serial.read();
  Serial.println(ComandoAT);
  respuesta=1;
  anterior = millis();

  while((respuesta == 0) && ((millis() - anterior) < Tiempo_agotado));
  return respuesta;
}
```

Código 3.3. Función ENVIO_COMANDOAT

La función ENVIO_COMANDOAT nos permitirá ingresar el comando deseado desde el código y esperar una respuesta. En caso de que no se encuentre habilitado y haya excedido el tiempo de espera, este dejará de enviar momentáneamente.

```
void loop() {
  while( Serial.available() > 0) Serial.read();
  delay(100);
  Envio_ComandoAT("AT+CLASS", 2000);
  delay(1000);
  Envio_ComandoAT("AT+ID=DevAddr", 2000);
}
```

Código 3.4. Código para uso de los comandos AT

En esta sección se envían dos comandos AT. Uno para verificar la clase con la que el nodo se encuentra configurado, y el otro comando es para solicitar la dirección del nodo que fue asignado de manera automática.

El código completo se lo puede observar en la sección de ANEXOS, como Anexo III.

3.1 RESULTADOS

3.1.1 ESCENARIO 1: ENCENDIDO DEL NODO COORDINADOR

En este escenario planteamos que ningún nodo se encuentre dentro del rango del nodo coordinador. Por lo que para simular dicho evento podemos hacer que el otro nodo no se encienda. Así, el nodo coordinador, luego de haber buscado a otro nodo para asignar una dirección, enviará un mensaje comentando que no pudo encontrar ningún nodo candidato para asignar la dirección secuencial.

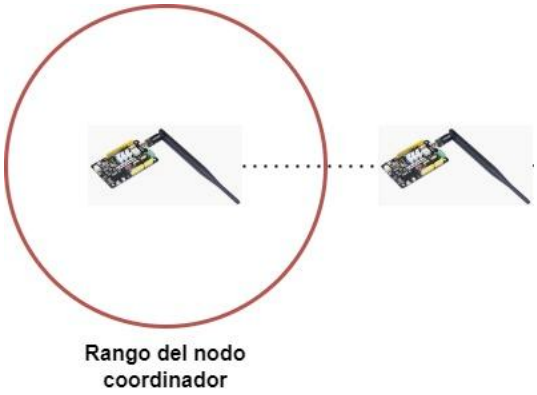

Figura 3.1. Escenario 1: Encendido del nodo coordinador

Utilizando el Código I, podemos observar el monitor serial con el resultado esperado.

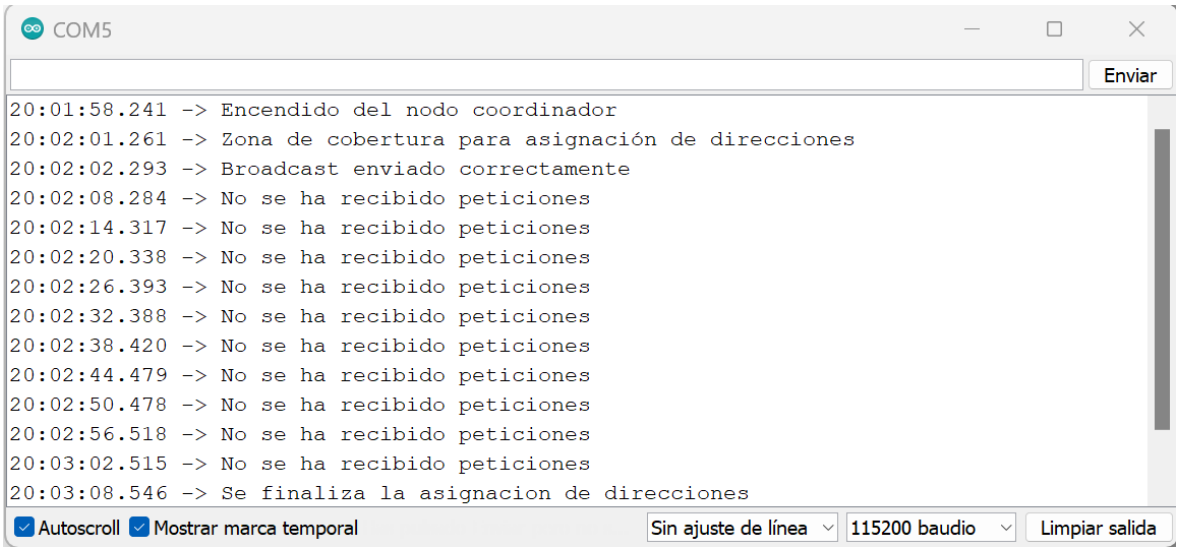

Figura 3.2. Monitor serial del nodo coordinador

El nodo coordinador empieza con su encendido y posterior a ello analiza la zona de cobertura que puede extenderse dependiendo de la potencia de transmisión empleada. Posterior a ello envía un mensaje de broadcast y espera las peticiones, pero al no tener una respuesta envía un mensaje al monitor informando que aún no se han recibido peticiones. Se creó el código para que reciba peticiones cada 6 segundos durante un minuto. Sin embargo, no pudo recolectar ninguna petición y procede a finalizar la asignación.

Además, también es posible observar las configuraciones realizadas en el nodo coordinador mediante el Código III.

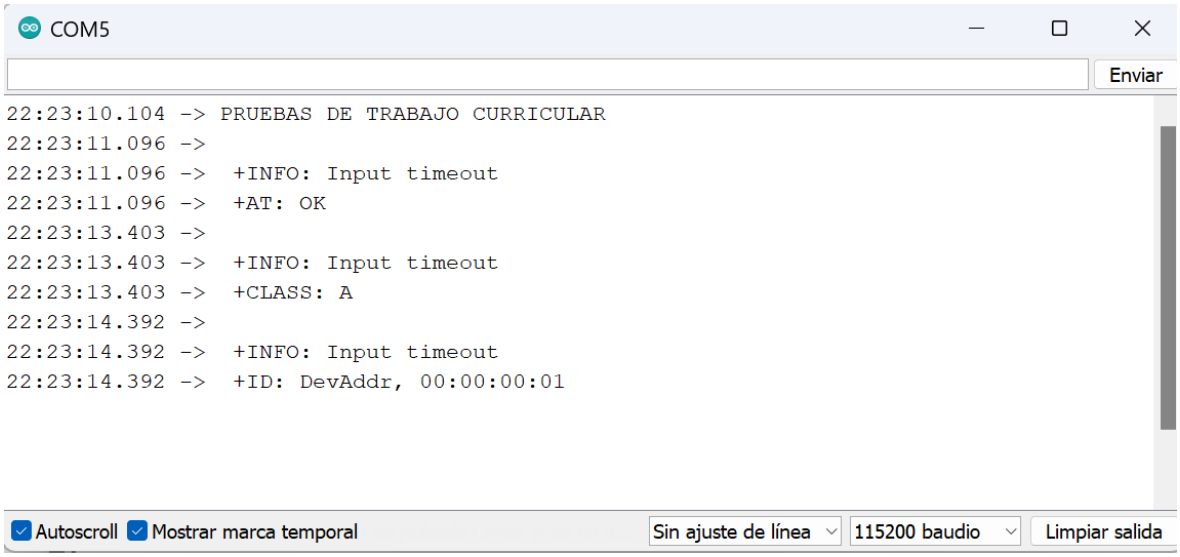

Figura 3.3. Comandos AT de la dirección

Aquí podemos observar que primero se recibe una respuesta de OK para comprobar que exista conexión entre los comandos y el nodo. Seguido de ello solicitamos que nos brinde la clase con la que se encuentra configurado, que es la clase A, y también la dirección del nodo coordinador, que resulta ser la misma configurada inicialmente, que es 0x00000001 o para un mejor entendimiento es 0x001.

Se observar que existen cuatro bloques en la dirección, cada uno representa el octeto (8 bits) de la trama de dirección (32 bits).

3.1.2 ESCENARIO 2: ENCENDIDO DEL NODO COORDINADOR Y UN NODO

En este escenario planteamos que un nodo se encuentre dentro del rango del nodo coordinador. Se van a crear dos sub-escenarios, donde en el primer caso el nodo estará situado a 10 metros de distancia, mientras que en el otro caso estará a 1 kilómetro de distancia. Así, el nodo coordinador asignará una dirección al nodo que realizó la petición y a su vez este se convertirá en el nuevo coordinador debido a que es el último nodo dentro del rango, lo cual le permitirá a futuro asignar más direcciones a los nodos que continúen en la topología lineal.

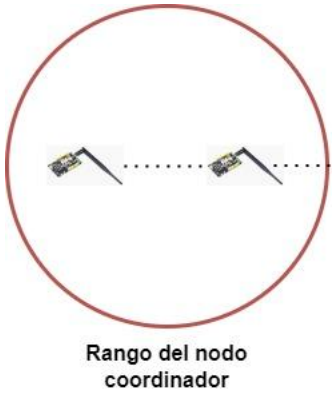

Figura 3.4. Escenario 2: Encendido del nodo coordinador y un nodo

3.1.2.1 Nodo a 10 metros de distancia

Ubicando el nodo en otra habitación con una distancia de aprox. 10 metros, se tiene los siguientes datos de haber asignado la dirección.

Utilizando el Código I, podemos observar el monitor serial con el resultado esperado.

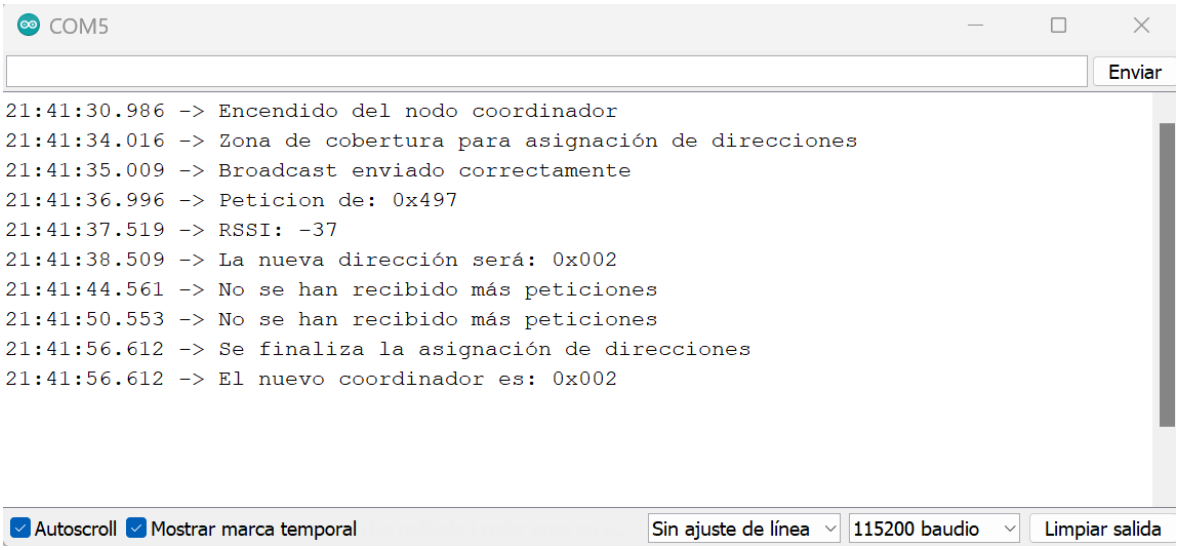

Figura 3.5. Monitor serial del nodo coordinador

Aquí podemos observar que el nodo coordinador encuentra otro nodo con dirección 0x497 que está solicitando una nueva asignación de dirección, por lo cual le corresponde el siguiente del nodo coordinador, es decir, se le asigna la dirección 0x002. Podemos observar que al estar muy cerca del nodo coordinador y con línea de vista, este tiene un RSSI de -37[dBm], siendo este valor muy excelente, donde la transmisión de datos no tendrá ningún problema. Una vez asignado la dirección, el nodo coordinador continúa en la búsqueda de más nodos para poder asignar, pero esta vez espera solamente 12

segundos por otra petición. Como no existen más peticiones entonces le asigna al último como nuevo nodo coordinador.

Utilizando el Código II, podemos observar el monitor serial con el resultado esperado.

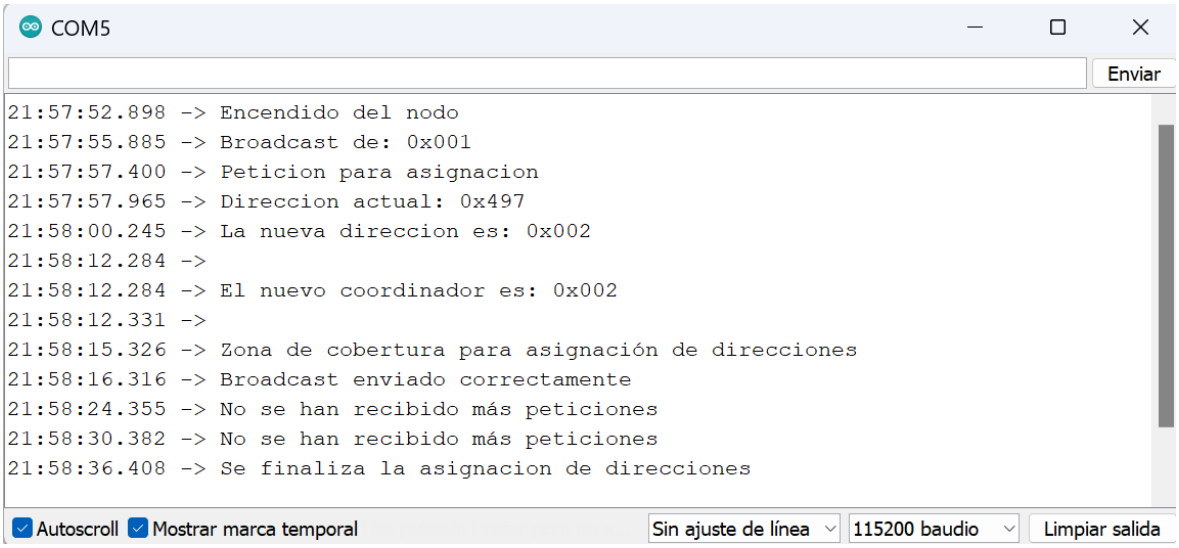

Figura 3.6. Monitor serial del nodo

En el monitor serial del nodo observamos que llega un mensaje de broadcast proveniente del nodo coordinador, a lo que el nodo le responde con una petición para que se le asigne una dirección. Inicialmente se encontraba con la dirección 0x497, pero luego de la asignación secuencial este viene a tener una dirección 0x002. Debido a que el nodo coordinador no encuentra más peticiones de otros nodos, entonces procede a asignar este nodo como el nuevo coordinador. De aquí se repite el proceso para otros nodos que se encuentren dentro del rango y que aún no hayan cambiado su dirección.

Además, también es posible observar las configuraciones realizadas en el nodo coordinador mediante el Código III.

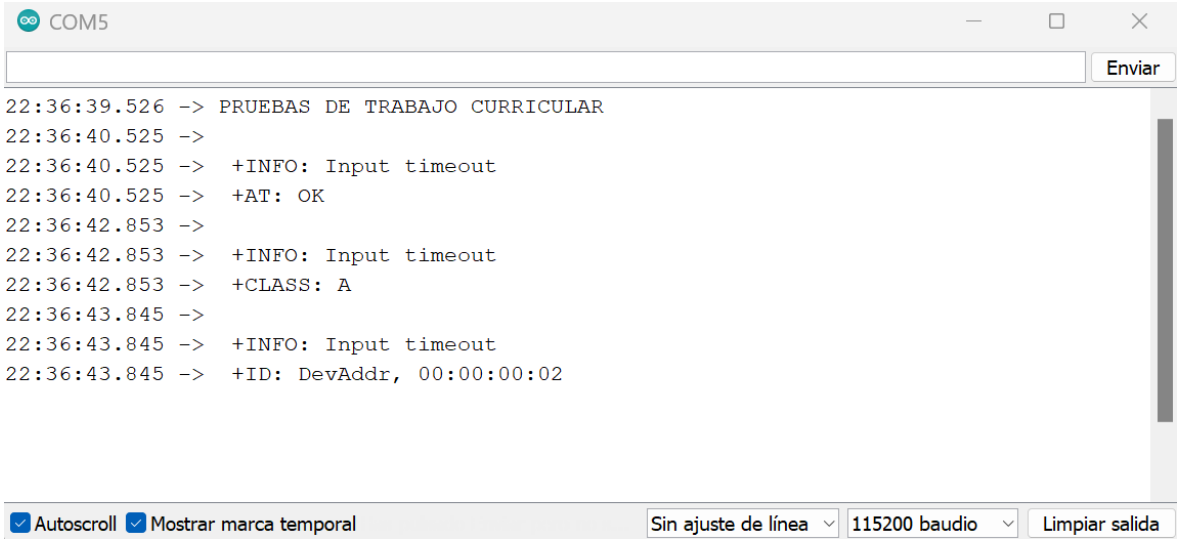

Figura 3.7. Comandos AT de la dirección

Aquí podemos observar que primero se recibe una respuesta de OK para comprobar que exista conexión entre los comandos y el nodo. Seguido de ello solicitamos que nos brinde la clase con la que se encuentra configurado, que es la clase A. También solicitamos la nueva dirección del nodo, que es la que se asignó automáticamente por el nodo coordinador, que es que es 0x00000002 o para un mejor entendimiento es 0x002.

Se observa que existen cuatro bloques en la dirección, cada uno representa el octeto (8 bits) de la trama de dirección (32 bits).

3.1.2.2 Nodo a 1 kilómetro de distancia

Ubicando el nodo a una distancia una distancia de aprox. 1 kilómetro, entre el edificio Metropolitan ubicado en la Av. Naciones Unidas e Iñaquito, y el edificio Mystique ubicado en la Rumipamba y Amazonas. Se ubican los nodos en los techos de los edificios para tener una mejor línea de vista, aunque un poco obstaculizado por otros edificios.

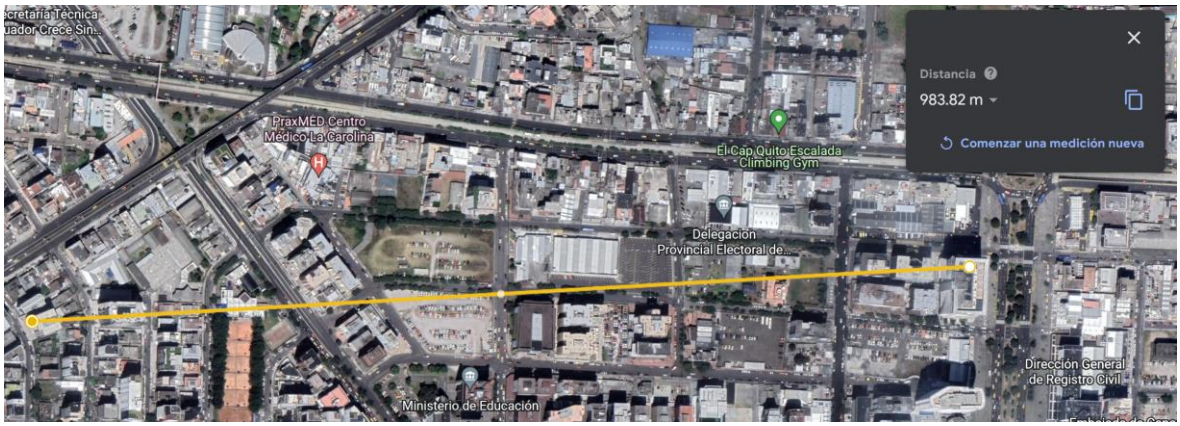

Figura 3.8. Distancia entre nodos

Utilizando el Código I, podemos observar el monitor serial con el resultado esperado.

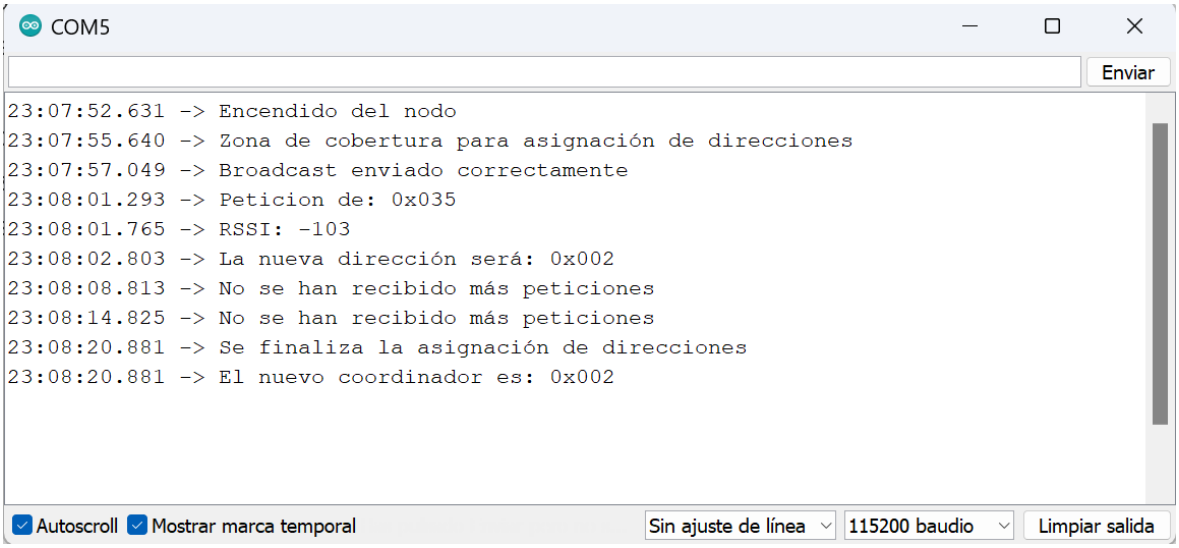

Figura 3.9. Monitor serial del nodo coordinador

Aquí podemos observar que el nodo coordinador encuentra otro nodo con dirección 0x035 (reconfigurado mediante comando AT para una nueva prueba) que está solicitando una nueva asignación de dirección, por lo cual le corresponde el siguiente del nodo coordinador, es decir, se le asigna la dirección 0x002. En este escenario los nodos ya se encuentran a una distancia mucho mayor que la primera, y con una línea de vista limitada, por lo que se tiene un RSSI de -103[dBm], siendo este posible para transmisión de pequeños paquetes, puesto que empieza a degradarse en gran cantidad. Una vez asignado la dirección, el nodo coordinador continúa en la búsqueda de más nodos para poder asignar, pero esta vez espera solamente 12 segundos por otra petición. Como no existen más peticiones entonces le asigna al último como nuevo nodo coordinador.

Utilizando el Código II, podemos observar el monitor serial con el resultado esperado.

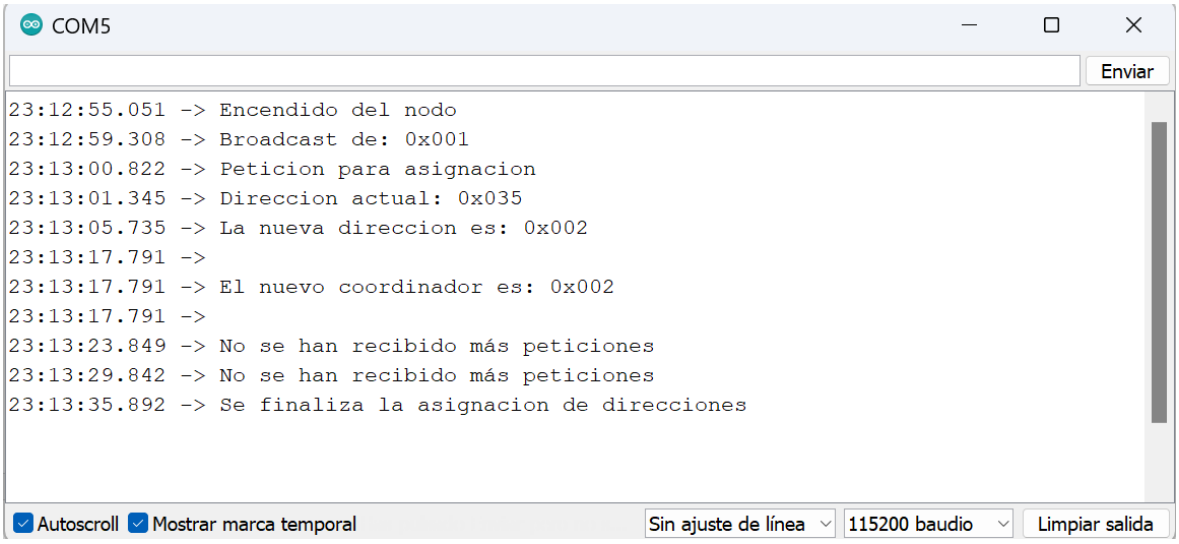

Figura 3.10. Monitor serial del nodo

En el monitor serial del nodo observamos que llega un mensaje de broadcast proveniente del nodo coordinador, a lo que el nodo le responde con una petición para que se le asigne una dirección. Inicialmente se encontraba con la dirección 0x035, pero luego de la asignación secuencial este viene a tener una dirección 0x002. Debido a que el nodo coordinador no encuentra más peticiones de otros nodos, entonces procede a asignar este nodo como el nuevo coordinador. De aquí se repite el proceso para otros nodos que se encuentren dentro del rango y que aún no hayan cambiado su dirección.

Además, también es posible observar las configuraciones realizadas en el nodo coordinador mediante el Código III.

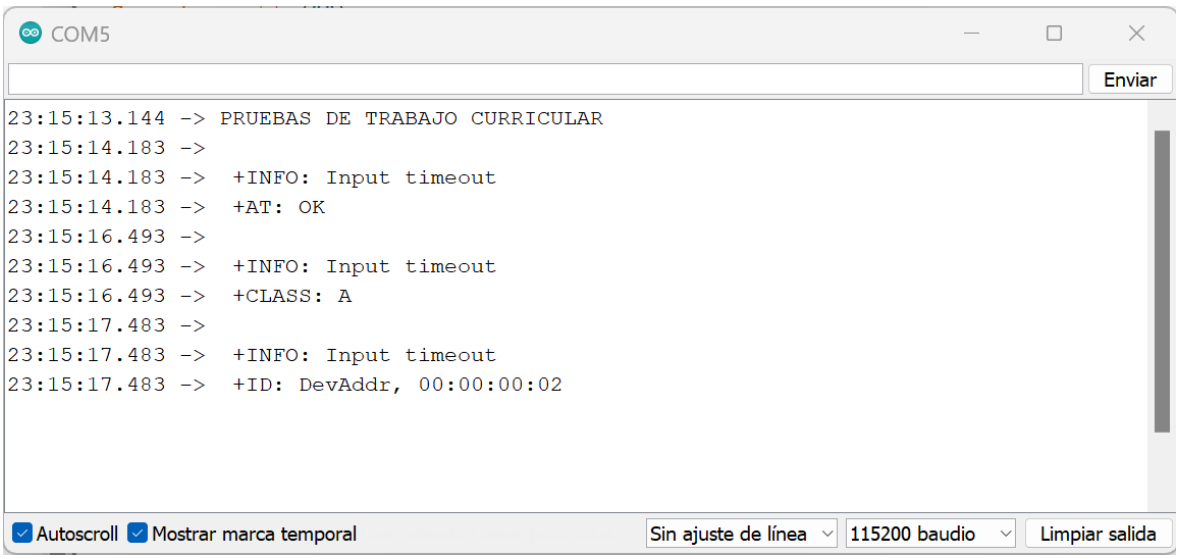

Figura 3.11. Comandos AT de la dirección

Aquí podemos observar que primero se recibe una respuesta de OK para comprobar que exista conexión entre los comandos y el nodo. Seguido de ello solicitamos que nos brinde la clase con la que se encuentra configurado, que es la clase A. También solicitamos la nueva dirección del nodo, que es la que se asignó automáticamente por el nodo coordinador, que es que es 0x00000002 o para un mejor entendimiento es 0x002.

Se observa que existen cuatro bloques en la dirección, cada uno representa el octeto (8 bits) de la trama de dirección (32 bits).

3.2 CONCLUSIONES

Cada vez se incrementa más el uso de las comunicaciones inalámbricas, y con ello se crean nuevas posibilidades para una conexión más efectiva, como por ejemplo la tecnología LoRaWAN, que nos permite tener un mayor rango de conectividad para el envío de datos con una tasa mínima, siendo útil en diferentes áreas donde no se tenga un fácil acceso para otras tecnologías.

El nodo LoRa-E5 puede trabajar en distintas configuraciones, lo que lo hace apto para aplicaciones de transmisión de datos en topologías lineales y gracias a ello puede consumir menos recursos con lo que es posible alargar su vida útil. Aparte, se puede adaptar a las necesidades del usuario para ser controlado de diversos modos, ya sea por comandos AT, codificación o lenguaje de microprocesadores.

Es posible realizar una implementación que trabaje en las capas 1 y 2 (física y de enlace de datos respectivamente) según el modelo OSI. Mediante ello se abre la posibilidad de disminuir las actividades de las otras capas superiores, reduciendo el procesamiento que afecta al rendimiento del nodo en la topología.

Los rangos de las direcciones difieren dependiendo cada región en la que se encuentre, siendo uno de estos el privado que se utiliza para realizar las pruebas en una región local. Así, nos aseguramos de que solamente se encuentre en nuestra propia topología y no interfiera con distintos operadores.

Mediante el RSSI obtenido en cada prueba, podemos decir que a medida que se vaya incrementando la distancia entre nodos, su potencia irá disminuyendo. Por lo que es importante tener una conexión entre distintos nodos con una topología lineal para no perder dicha potencia y así llegar al destino de forma segura. Mientras que, si la topología implementa una protección de multisalto, esta será mucho más robusta y resistente a fallos o pérdidas debido a la caída de algún nodo.

A medida que se incrementa la distancia, el retardo va creciendo poco a poco, haciendo que sea proporcional entre ambos parámetros. Debido a esto, el nodo coordinador tarda más tiempo en encontrar y asignar la dirección a otro nodo.

El presente Trabajo Curricular puede servir como base para conocer una de las maneras en las que el nodo LoRa-E5 puede ser configurado, ya que se encuentra explicado paso a paso la manera en la que se debe proceder para su uso.

3.3 RECOMENDACIONES

Revisar que todos los controladores y firmwares de los elementos a usarse se encuentren actualizados para que se pueda tener un correcto funcionamiento de ellos.

Transformar las direcciones de hexadecimal a decimal para tener un mejor entendimiento de los mismos y así controlar de manera más sencilla la asignación secuencial de las direcciones.

En caso de configurar el nodo en modo SWD, es sumamente importante quitar el modo de protección de lectura. Con ello es posible modificar el programa interno del nodo para poder controlarlo con libertad y lograr quemar los códigos.

En caso de replicar el presente Trabajo Curricular con una mayor cantidad de nodos, se sugiere probar con distintos escenarios como, por ejemplo, un tercer nodo que no se encuentre dentro del rango del nodo coordinador, y así el segundo nodo se convertirá en el coordinador y asignará una dirección al tercer nodo. Con esto, además de recopilar más información, se podrá demostrar que efectivamente el algoritmo realizado servirá para la cantidad de nodos existentes.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] S. Greengard, *The Internet of Things. Cambridge*, MA, USA: MIT Press, 2015.

[2] A. Cama-Pinto, E. De-La-Hoz-Franco, y D. Cama-Pinto, “Las redes de sensores inalámbricos y el Internet de las cosas”, *Wireless sensor networks and the Internet of Things*, oct. 2012. [Online]. Available: http://hdl.handle.net/11323/1546

[3] C. Egas, F. Gil-Castiñeira, y E. Costa-Montenegro, “Red inalámbrica de sensores con topología lineal sin capa de red”, *Revista de Investigación en Tecnologías de la Información*, vol. 9, núm. 17, Art. núm. 17, ene. 2021.

[4] “Regional Parameters”, *The Things Network*. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/

[5] T. Ameloot, P. Van Torre, y H. Rogier, “A Compact Low-Power LoRa IoT Sensor Node with Extended Dynamic Range for Channel Measurements”, *Sensors*, vol. 18, núm. 7, Art. núm. 7, jul. 2018, doi: 10.3390/s18072137.

[6] K. Rose, S. Eldrigde y L. Chapin, “The Internet of Things: An Overview. Undestranding the Issues and Challenges of a More Connected World”, *Internet Society of Things*, oct. 2015.

[7] J. Holler, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, y D. Boyle, *Internet of Things*. Academic Press, 2014.

[8] A. Corral, "Diseño e implementación de un entorno de simulación para redes inalámbricas de sensores" *Universidad Politécnica de Cartagena*, 2006.

[9] L. Hernandez, Y. Calderon, H. Martinez, A. Pranolo e I. Riyanto, "Design or a system for detection or enviromental variables applied in data centers", *Third International Conference on Science in Information Technology (ICSITech)*, 2017

[10] C. Aranzazu y G. Moreno, "Revisión del estado del arte de las redes inalámbricas de sensores", *Revista Politécnica*, vol. 5, n° 9, pp. 89-115, 2009.

[11] M. Lee, S. Subramaniam, A. Azman y F. Feroz, "Performance Analysis of Lineal Topology Wireless Sensor Network in Gas and Oil Industry", *IOP Conference Series: Science and Engineering*, vol. 765, num 120, mar. 2020, doi: 10.1088/1757-899X/765/1/012070

[12] A. Raychowdhury y A. Pramanik, "Survey on LoRa Technology: Solution for Internet of Things", *Technologies and Applications*, Singapore, 2020, pp. 259-271. doi: 10.1007/978-981-15-3914-5_20.

[13] S. Devalal y A. Karthikeyan, "LoRa Technology - An Overview", *Second International Conference on Electronics*, Communication and Aerospace Technology (ICECA), mar. 2018, pp. 284-290. doi: 10.1109/ICECA.2018.8474715.

[14] M. Ramos, "Estudio de la tecnología LoRaWAN mediante la implementación de un prototipo de medición de humedad y temperatura", *Escuela Politécnica Nacional,* Quito, dic. 2021

[15] "LoRa and LoRaWAN: A Technical Overview", *Semtech Corporation*, nov. 2021. [Online]. Available: https://lora-developers.semtech.com/documents/LoRa_and_LoRaWAN-A_Tech_Overview.pdf

[16] M. Knight y B. Seeber, "Decoding LoRa: Realizing a Modern LPWAN with SDR", *Proceedings of the GNU Radio Conference*, vol. 1, art. 1, sep. 2016,

[17] A. Yegin et al., "LoRaWAN protocol: specifications, security, and capabilities", *LPWAN Technologies for IoT and M2M Applications*, Eds. Academic Press, 2020, pp. 37-63. doi: 10.1016/B978-0-12-818880-4.00003-X.

[18] "Wio-E5 Development Kit", *Seeed Wiki.* [Online]. Available: https://wiki.seeedstudio.com/LoRa_E5_Dev_Board/

[19] "Wio-E5 Wireless Module (Bulk), for Long Range Applications", *Seeed Studio*, sep. 2022. [Online]. Available: https://www.seeedstudio.com/LoRa-E5-Wireless-Module-p-4745.html

[20] "Seeed Studio LoRa-E5 Dev Board", *Zephyr Project Documentation*. [Online]. Available: https://docs.zephyrproject.org/boards/doc/lora_e5_dev_board.html

[21] "LoRa E5. AT Command Specification", *Seeed Studio*. [Online]. Available: https://files.seeedstudio.com/products/317990687/res/LoRa-E5-20module-20datasheetV1.0.pdf

[22] "NetID and DevAddr Prefix Assignments", *The Things Network*. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/prefix-assignments/

[23] H. Choudhary, "Microcontroller Programmer/Burner", *Engineers Garage*. [Online]. Available: https://www.engineersgarage.com/microcontroller-programmer-burner/

[24] M. Cardozo, "Usos y aplicaciones del programador ST-Link v2", *Sunrom Electronics*. [Online]. Available: https://www.sunrom.com/usos_aplicaciones_programador_st-link-v2

[25] "Arduino Uno | Documentation", *Arduino Doc*. [Online]. Available: https://docs.arduino.cc/hardware/uno

[26] C. Stramaglia, "Seeed LoRa E5 Tutorial with ST programmer", *Hackster.io*, nov.2022. [Online]. Available: https://www.hackster.io/cstram/seeed-lora-e5-tutorial-with-st-programmer-493057

5 ANEXOS

ANEXO I. Código del nodo coordinador

ANEXO II. Código del nodo

ANEXO III. Código de verificación (para resultados)

ANEXO IV. Video del funcionamiento del prototipo para asignación de direcciones

ANEXO I

Código del nodo coordinador

```
#include <LoRa.h>
#include <LoRaNode.h>
#include <SPI.h>

unsigned long tiempo_espera;

#define DEVICE_ADDRESS "00000001" //Dirección para el nodo coordinador
String devAddr=DEVICE_ADDRESS

#define ClassA "A"      //Selección de la Clase A
#define Tx_Power 14     //Potencia adecuada para transmisión
#define Tx_Preamble 8   //Preámbulo de transmisión menor a preámbulo de recepción
#define BW 125000       //Ancho de banda acorde a soporte de LoRa-E5

LoRa nodo_coordinador;
LoRaNode lora_e5;


void setup() {
  Serial.begin(115200);
  if (!nodo_coordinador.begin(US915)) {   //Se toman los parámetros de US (con frecuencia de 915[kHz])
    Serial.println("Falla de encendido del nodo coordinador");
  }

  Serial.println("Encendido del nodo coordinador");
  lora_e5.setClass(ClassA);                       //Configuración para Clase A
  lora_e5.setTxPower(Tx_Power);                   //Configuración para potencia de 14[dBm]
  lora_e5.setSignalBandwidth(BW);                 //Configuración para ancho de banda de 125[kHz]
  lora_e5.setPreambleLength(Tx_Preamble);         //Configuración para una longitud de preámbulo de 8

  int connected=nodo_coordinador.joinABP(devAddr); //Configuración de modo Activation By Personalization (ABP)
}


void loop() {

  while(!Serial.available());
  String mensaje_broadcast="Zona de cobertura para asignacion de direcciones";

  int error;
  nodo_coordinador.beginPacket();               //Envío de broadcast a todos los nodos
  nodo_coordinador.send(mensaje_broadcast);
  error=nodo_coordinador.endPacket(true);

  if (error>0) {
    Serial.println("Broadcast enviado correctamente");
    tiempo_espera=millis();
    delay(1000);
    int a=0;

    while (a==0){
      if (!nodo_coordinador.available()){
        Serial.println("No se ha recibido peticiones");
        if(tiempo_espera>6000){                 //Tiempo de espera de que no se haya recibido ninguna respuesta
          a=2;
        }
      } else {
        a=1;
      }
    }

    int ID=2;
    while (nodo_coordinador.available()) {
      uint8_t recibido[]=nodo_coordinador.read(); //Lectura de petición
      uint8_t tamano=sizeof(recibido);            //Tamaño del mensaje de petición
      nodo_coordinador.recvMSG(recibido, &tamano, 3000, &from)
      Serial.print("Peticion de: 0x");
      Serial.println(lora_e5.from, HEX);          //ID del nodo que hace la petición
      Serial.print("RSSI: ");
      Serial.println(lora_e5.lastRssi(), DEC);    //RSSI del nodo que hace la petición
```

```
      delay(500);
      nodo_coordinador.beginPacket(from);         //Envío de nueva asignación a nodo que hizo la petición
      nodo_coordinador.send(ID);
      error=nodo_coordinador.endPacket(true);
      ID++;
      nuevo_coordinador=from;
      a=2;
    }

    if(a==2){ //Asignación de coordinador
      Serial.println("Se finaliza la asignacion de direcciones");
      Serial.print("El nuevo coordinador es: 0x");
      Serial.println(nuevo_coordinador);

      nodo_coordinador.beginPacket(nuevo_coordinador);
      nodo_coordinador.send(nuevo_coordinador);
      nodo_coordinador.endPacket(true);
    }
  }

  exit(0);
}
```

ANEXO II

Código del nodo

```
#include <LoRa.h>
#include <LoRaNode.h>
#include <SPI.h>

unsigned long tiempo_espera;

#define ClassA "A"      //Selección de la Clase A
#define Tx_Power 14     //Potencia adecuada para transmisión
#define Tx_Preamble 8   //Preámbulo de transmisión menor a preámbulo de recepción
#define BW 125000       //Ancho de banda acorde a soporte de LoRa-E5

LoRa nodo;
LoRaNode lora_e5;


void setup() {
  Serial.begin(115200);
  if (!nodo.begin(US915)) {   //Se toman los parámetros de US (con frecuencia de 915[kHz])
    Serial.println("Falla de encendido del nodo");
  }

  Serial.println("Encendido del nodo");
  lora_e5.setClass(ClassA);                       //Configuración para Clase A
  lora_e5.setTxPower(Tx_Power);                   //Configuración para potencia de 14[dBm]
  lora_e5.setSignalBandwidth(BW);                 //Configuración para ancho de banda de 125[kHz]
  lora_e5.setPreambleLength(Tx_Preamble);         //Configuración para una longitud de preámbulo de 8
}


void loop() {

  while (!Serial.available()){

    while (nodo.available()) {
      uint8_t recibido[]=nodo.read();   //Lectura de broadcast
      uint8_t tamano=sizeof(recibido);  //Tamaño de broadcast
      nodo.recvMSG(recibido,&tamano, 3000, &from);
      if (tamano==48){     //Verificar el tamaño del mensaje y no confundirse con el nodo que tambien solicita
        Serial.print("Broadcast de: 0x");
        Serial.println(lora_e5.from, HEX);  //ID del nodo coordinador

        delay(500);
        nodo.beginPacket(from); //Envío de mensaje de petición a coordinador
        nodo.send("Peticion para asignación");
        error=nodo.endPacket(true);

        delay(500);
        int nuevo_ID=nodo.read(); //Lectura de nueva asignación de dirección
        lora_e5.setDevAddr(nuevo_ID, HEX);  //Configuración de la nueva asignación de dirección
      }

      int coordinador=nodo.read();  //Lectura de nuevo nodo coodinador
      if(coordinador==nuevo_ID){  //En caso de que el ID coincida con el nuevo coordinador
          while(!Serial.available());
          String mensaje_broadcast="Zona de cobertura para asignación de direcciones";

          int error;
          nodo.beginPacket();
          nodo.send(mensaje_broadcast);
          error=nodo.endPacket(true);

          if (error>0) {
            Serial.println("Broadcast enviado correctamente");
            tiempo_espera=millis();
            delay(1000);
            int a=0;
```

```
            while (a==0){
              if (!nodo.available()){
                Serial.println("No se ha recibido peticiones");
                if(tiempo_espera>6000){
                  a=2;
                }
              } else {
                a=1;
             }
            }

            int ID=nuevo_ID+1;
            while (nodo.available()) {
              uint8_t recibido[]=nodo.read();
              nodo.recvMSG(recibido, 3000, &from)
              if(from > nuevo_ID){
                Serial.print("Peticion de: 0x");
                Serial.println(lora_e5.from, HEX);
                Serial.print("RSSI: ");
                Serial.println(lora_e5.lastRssi(), DEC);

                delay(500);
                nodo.beginPacket(from);
                nodo.send(ID);
                error=nodo.endPacket(true);
                ID++;
                nuevo_coordinador=from;
                a=2;
              }
            }

            if(a==2){
              Serial.println("Se finaliza la asignacion de direcciones");
              Serial.print("El nuevo coordinador es: 0x");
              Serial.println(nuevo_coordinador);


               nodo.beginPacket(nuevo_coordinador);
               nodo.send(nuevo_coordinador);
               nodo.endPacket(true);
             }
           }
       }
     }
  }
  exit(0);
}
```

ANEXO III

Código de verificación (pruebas)

```
#include <SoftwareSerial.h>

int8_t respuesta; //Entero de 8 bits
int Pin_TX= 1;

void setup() {
  pinMode(Pin_TX, OUTPUT); //Definición para Pin (salida)
  Serial.begin(115200);    //V_tx del puerto serial
  delay (2000);
  Serial.println("PRUEBAS DE TRABAJO CURRICULAR");
  Encendido();             //Función para verificar encendido mediante respuesta de comando AT
  delay(3000);
}

//Función para verificar que se encuentre funcional por medio de comandos AT
void Encendido(){
  uint8_t respuesta=0;
  respuesta = Envio_ComandoAT("AT", 2000); //Chequear que estén en transmisión los comandos AT (se encuentre encendido)
  if (respuesta == 0) {
    digitalWrite(Pin_TX,HIGH); //Cambio de estado del Pin (alto)
    delay(3000);
    digitalWrite(Pin_TX,LOW); //Cambio de estado del Pin (bajo)
    while(respuesta == 0){
      respuesta = Envio_ComandoAT("AT", 2000); //Se envía cada 2[s] y se espera por una respuesta
    }
  }
}

//Función para poder enviar los comandos AT directamente desde el código
int8_t Envio_ComandoAT(char* ComandoAT, unsigned int Tiempo_agotado) {
  uint8_t x=0, respuesta=0;
  char response[100];
  unsigned long anterior;
  memset(response, '\0', 100); // Inicializa el string
  delay(100);
  // Limpia el buffer de entrada
  while( Serial.available() > 0) Serial.read();
  Serial.println(ComandoAT); //Imprime el comando AT que se desea enviar
  respuesta=1;
  anterior = millis(); //Empieza a contar el tiempo (para verificar tiempo de espera)

  while((respuesta == 0) && ((millis() - anterior) < Tiempo_agotado));
  return respuesta;
}

void loop() {
  while( Serial.available() > 0) Serial.read();
  delay(100);
  Envio_ComandoAT("AT+CLASS", 2000);//Comando para verificar la clase del nodo
  delay(1000);
  Envio_ComandoAT("AT+ID=DevAddr", 2000);//Comando para solicitar la dirección del nodo
}
```

ANEXO IV

Video del funcionamiento del prototipo para asignación de direcciones

Se adjunta un link que redirige al video donde se realizan las pruebas de los escenarios planteados.

One Drive institucional:

https://epnecuador-my.sharepoint.com/:v:/g/personal/sebastian_bolanos_epn_edu_ec/ESRinLwjRb1Lvq5N5J0jH40BZahekGvxDJdBZlsQMoDUUw?e=nrvec1

Youtube:

https://youtu.be/NQDnCfJBSj0

*En ambos casos es el mismo video pero insertado en distintas plataformas.