ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

Implementación de herramientas de laboratorio para trabajar con los protocolos LoRaWAN y IEEE 802.15.4

PROTOTIPO PARA EVALUAR LA TASA DE TRAMAS PERDIDAS (FER) EN IEEE 802.15.4

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

BRYAN DAVID RODRIGUEZ VILLACRÉS
bryan.rodriguez01@epn.edu.ec

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

DMQ, abril 2024

CERTIFICACIONES

Yo, Bryan David Rodríguez Villacrés 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.

BRYAN DAVID RODRÍGUEZ VILLACRÉS

Certifico que el presente trabajo de integración curricular fue desarrollado por Bryan David Rodríguez Villacrés, bajo mi supervisión.

CARLOS ROBERTO EGAS ACOSTA
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante 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.

BRYAN DAVID RODRÍGUEZ VILLACRÉS

ING. CARLOS ROBERTO EGAS ACOSTA

DEDICATORIA

A mis queridos padres y hermanos, quienes han sido mi mayor fuente de amor, inspiración y apoyo incondicional a lo largo de toda mi vida. Su sacrificio y dedicación han sido la luz que me ha guiado en cada paso de este camino académico. Este trabajo está dedicado a ustedes, con todo mi amor y gratitud eterna.

A mis amigos y compañeros de universidad, Andrés Vera, Carlos Flores, Steven Peña, Paul Villagrán, David Mosquera y Alejandro Barros quienes han estado a mi lado a lo largo de toda la carrera siempre apoyándonos, compartiendo buenos y malos momentos, enfrentando dificultades y celebrando logros.

A Cristian Curipallo quien es mi mejor amigo y siempre ha estado para apoyarme y darme ánimos para poder llegar a ser quien soy, su amistad ha sido un regalo invaluable que ha enriquecido mi vida de formas inimaginables.

A Kimberly Cox quien además de ser mi pareja también ha sido mi mejor amiga y compañera en este viaje. Tu amor, comprensión y apoyo incondicional han sido un faro de luz en los momentos más oscuros.

Este trabajo está dedicado a ustedes, por ser parte indispensable de mi viaje y por su constante apoyo y ánimo.

Con cariño y gratitud,

Bryan David Rodríguez Villacrés

AGRADECIMIENTO

Quiero comenzar expresando mi sincero agradecimiento a mis padres y hermanos por su amor incondicional, constante apoyo y sacrificios que hicieron posible mi educación y la realización de este trabajo. Su aliento y motivación fueron mi mayor inspiración en cada paso de este viaje académico.

A mis compañeros de clase, les agradezco por su colaboración, discusiones estimulantes y amistad durante estos años. Sus perspectivas y críticas constructivas han enriquecido enormemente este trabajo y mi experiencia académica en general.

También quiero expresar mi profundo agradecimiento a mi director de TIC. Su guía experta, paciencia y dedicación fueron fundamentales para dar forma a este proyecto y llevarlo a buen término. Estoy enormemente agradecido por su apoyo y orientación a lo largo de este proceso de investigación.

A todas estas personas, mi más sincero agradecimiento. Sin su contribución, este logro no habría sido posible. Estoy eternamente agradecido por su generosidad, apoyo y confianza en mí.

ÍNDICE DE CONTENIDO

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

1 INTRODUCCIÓN 1
1.1 Objetivo general 2
1.2 Objetivos específicos 2
1.3 Alcance 2
1.4 Marco teórico 3
1.4.1 ESTÁNDAR IEEE 802.15.4 3
1.4.2 ESTRUCTURA DEL PROTOCOLO 802.15.4 4
1.4.3 ARQUITECTURA DEL PROTOCOLO 802.15.4 4
1.4.4 TOPOLOGÍAS DEL ESTÁNDAR 802.15.4 8
1.4.5 TRANSCEPTOR ATZB-256RFR2-XPRO 11

2 METODOLOGÍA 13
2.1 EXPOSICIÓN DEL PROBLEMA 13
2.2 DISEÑO DE LA HERRAMIENTA 13
2.3 DISEÑO DEL ALGORITMO 14
2.3.1 ALGORITMO DEL NODO TRANSMISOR 15
2.3.2 ALGORITMO DEL NODO RECEPTOR E INICIADOR 18
2.3.3 ALGORITMO DEL SOFTWARE PARA LA PC 19
2.3.4 ELEMENTOS NECESARIOS PARA LA IMPLEMENTACIÓN DEL ALGORITMO 21
2.3.5 CODIFICACIÓN DE LOS ALGORITMOS 22

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES 46
3.1 PRUEBAS 46
3.1.1 PRUEBA 1 46

3.1.2 PRUEBA 2 47
3.1.3 PRUEBA 3 48
3.1.4 PRUEBA 4 50
3.1.5 PRUEBA 5 51
3.1.6 PRUEBA 6 51
3.1.7 PRUEBA 7 53
3.2 Conclusiones 54
3.3 Recomendaciones 55

4 REFERENCIAS BIBLIOGRÁFICAS 56

5 ANEXOS I

RESUMEN

El proyecto se encamina hacia el desarrollo de una herramienta especializada para medir la tasa de tramas perdidas en transmisiones bajo el protocolo 802.15.4, con un enfoque técnico que involucra la implementación en lenguaje C. Se prevé que esta elección de lenguaje proporcionará eficiencia y control a nivel de sistema, asegurando una ejecución robusta y optimizada de la herramienta en diversos entornos de red.

Adicionalmente, se contempla la creación de una aplicación complementaria para PC utilizando el lenguaje Python. Esta aplicación permitirá una interfaz de usuario intuitiva y accesible, facilitando la configuración de la herramienta y la interpretación de los resultados. La combinación de estas dos implementaciones, una en lenguaje C para el núcleo de la herramienta y otra en Python para la interfaz de usuario, busca aprovechar las fortalezas específicas de cada lenguaje, ofreciendo así una solución integral y eficaz.

Este enfoque técnico dual refleja una estrategia integral para el desarrollo de la herramienta, optimizando tanto la eficiencia del código como la experiencia del usuario. La planificación futura incluirá la validación exhaustiva de ambas implementaciones, asegurando una herramienta funcional, versátil y lista para abordar las complejidades de la medición de tramas perdidas en entornos basados en el protocolo 802.15.4.

PALABRAS CLAVE: Protocolo, 802.15.4, Python, lenguaje C

ABSTRACT

The project is geared towards developing a specialized tool to measure the frame loss rate in transmissions under the 802.15.4 protocol, with a technical focus involving implementation in the C language. This language choice is expected to provide efficiency and system-level control, ensuring robust and optimized execution of the tool in various network environments.

Additionally, the creation of a complementary PC application using the Python language is envisaged. This application will offer an intuitive and user-friendly interface, simplifying the configuration of the tool and interpretation of results. The combination of these two implementations—C for the core of the tool and Python for the user interface—aims to leverage the specific strengths of each language, providing a comprehensive and effective solution.

This dual technical approach reflects a holistic strategy for tool development, optimizing both code efficiency and user experience. Future planning will include thorough validation of both implementations, ensuring a functional, versatile tool ready to address the complexities of frame loss measurement in environments based on the 802.15.4 protocol.

KEYWORDS: Protocol, 802.15.4, Python, languaje C.

1 INTRODUCCIÓN

El protocolo 802.15.4 ha emergido como un estándar fundamental en el panorama de las comunicaciones inalámbricas, especialmente en el contexto de las redes de área personal (WPAN). Su relevancia radica en su capacidad para facilitar la transmisión de datos en entornos donde se requiere una cobertura limitada, pero eficiente, abriendo paso a un mundo de aplicaciones en la era de la conectividad.

En la estructura del protocolo 802.15.4, la transmisión de tramas sigue una secuencia bien definida. Sin embargo, la precisión en la medición de la tasa de tramas perdidas dentro de este protocolo ha sido un desafío constante debido a la falta de herramientas específicas. Las metodologías convencionales de medición no siempre se adaptan a las particularidades y complejidades de este estándar, dificultando la evaluación exacta de la calidad de la transmisión. [1]

Este desafío se amplifica al considerar la falta de herramientas precisas para medir las tramas perdidas en el contexto del protocolo 802.15.4. Esta carencia representa una brecha significativa en la evaluación de la calidad y confiabilidad de las redes que operan bajo este estándar. La dificultad en la evaluación de la TER y las tramas perdidas subraya la necesidad urgente de desarrollar herramientas de medición adaptadas a las especificaciones únicas de este protocolo, permitiendo así evaluar y mantener la integridad de las redes de corto alcance basadas en el 802.15.4.

La relación del protocolo 802.15.4 con el crecimiento de la necesidad de redes de corto alcance es innegable. El aumento exponencial de dispositivos IoT y la demanda de comunicaciones eficientes en áreas con cobertura limitada han impulsado la relevancia de este estándar. Su capacidad para operar con bajos niveles de energía, establecer redes WPAN y adaptarse a entornos cambiantes lo posiciona como una solución idónea para aplicaciones en entornos domésticos, industriales o médicos. [2]

La eficiencia energética, flexibilidad escalable y costos reducidos del protocolo 802.15.4 han fomentado su adopción en entornos donde la interconexión de dispositivos en áreas restringidas es esencial. A pesar de su utilidad, la carencia de herramientas precisas para medir la TER y las tramas perdidas resalta la importancia de desarrollar dispositivos de medición específicos, fundamentales para mantener la calidad y fiabilidad de estas redes en evolución. [3]

En este Trabajo de Integración Curricular se tomará como enfoque principal el desarrollo de

una herramienta para la medición de las tasas de tramas perdidas en el protocolo 802.15.4.

1.1 OBJETIVO GENERAL

Desarrollar una herramienta para la evaluación del porcentaje de error en la transmisión de tramas punto a punto utilizando el protocolo 802.15.4.

1.2 OBJETIVOS ESPECÍFICOS

1. Realizar una revisión sobre el fundamento teórico del protocolo 802.15.4.
2. Revisar las características y el funcionamiento del microcontrolador ATMEGA RCB256RFR2.
3. Desarrollar un algoritmo para transmitir y recibir tramas con protocolo 802.15.4 de manera continua a partir de una señal de inicio
4. Realizar pruebas de transmisión y recepción variando parámetros como la distancia entre los nodos, la velocidad de transmisión, etc. Para verificar la tasa de error.

1.3 ALCANCE

El objetivo principal del componente es el desarrollo de una herramienta cuyo fin es evaluar la tasa de tramas perdidas en una transmisión punto a punto sobre el protocolo 802.15.4. Como punto de partida se realizará una revisión de la base teórica general del protocolo 802.15.4, de las herramientas de programación y de los microcontroladores ATmegaRCB256RFR2.

La herramienta se compone del software a implementar en los nodos y el software para implementar en la PC.

El software a implementar en los nodos realizará las siguientes tareas:

- ❒ Enviar los parámetros de transmisión
- ❒ Obtener la información de los nodos.
- ❒ Recibir los parámetros y armar las tramas que se enviará consecutivamente hacia el nodo receptor.
- ❒ El nodo receptor almacenará las tramas recibidas y enviará por puerto serial hacia la PC.
- ❒ Procesar la información de las tramas obtenidas.

El software se compone de tres algoritmos para los nodos que conformarán nuestra red, el primer algoritmo será para que el nodo 1 empiece a emitir 100 tramas de manera continua en la frecuencia deseada después de recibir una señal de inicio, el siguiente algoritmo será para que el nodo 2 en estado de recepción envíe las tramas recibidas por el nodo 1 de manera serial hacia una computadora, y el tercer algoritmo tiene como objetivo que el nodo 3 envíe la señal de inicio al nodo 1 después de presionar un botón.

Una vez desarrollados los algoritmos se realizarán pruebas para verificar las tramas perdidas, los nodos transmisor y receptor se colocarán a varias distancias con el objetivo de tener varios resultados para poder realizar la respectiva comparación.

Después de cada prueba se obtendrán los porcentajes de pérdidas con los cuales se realizará un breve análisis de la efectividad de transmisión de datos con el protocolo 802.15.4 para esto el software del PC realizará una comparación entre las tramas enviadas y las tramas recibidas, en base a esta división se obtiene el porcentaje de tramas perdidas.

1.4 MARCO TEÓRICO

1.4.1 ESTÁNDAR IEEE 802.15.4

El estándar IEEE 802.15.4 fue creado por el Grupo de trabajo de Área Personal Inalámbrica (WPAN) del IEEE (Institute of Electrical and Electronics Engineers) con el objetivo principal de proporcionar un marco para las comunicaciones inalámbricas de baja potencia y corto alcance.[4]

Se comenzó a desarrollar a finales de los años 90 y se publicó por primera vez en 2003. Su concepción surgió en respuesta a la necesidad de un estándar que permitiera la conectividad entre dispositivos con limitaciones de energía, como sensores, actuadores y otros dispositivos de baja complejidad, en entornos donde se requería una cobertura limitada pero eficiente.

El grupo de trabajo se propuso diseñar un protocolo que permitiera la comunicación fiable y eficiente entre estos dispositivos, priorizando la conservación de la energía como uno de los pilares fundamentales. Este enfoque era esencial para aplicaciones emergentes, como el Internet de las cosas (IoT), donde la conectividad entre dispositivos pequeños y de bajo consumo energético era crucial. [2]

El estándar IEEE 802.15.4 se distingue por su capacidad para ofrecer múltiples tasas de

datos, flexibilidad en la configuración de redes y su adaptabilidad a diferentes entornos y aplicaciones. A lo largo del tiempo, ha evolucionado con diversas revisiones y enmiendas para mejorar su eficiencia, capacidad de coexistencia con otras tecnologías inalámbricas y la robustez de las comunicaciones en entornos adversos.

La creación del estándar IEEE 802.15.4 representó un hito significativo en el desarrollo de tecnologías inalámbricas de baja potencia y corto alcance, sentando las bases para el crecimiento y la expansión de las redes de sensores, el IoT y otras aplicaciones que requieren conectividad eficiente en entornos con restricciones de energía y alcance limitado. [5]

1.4.2 ESTRUCTURA DEL PROTOCOLO 802.15.4

El protocolo IEEE 802.15.4 sigue una estructura definida en la transmisión de datos, que se compone de diferentes elementos organizados en una trama (frame) de comunicación. Esta trama consta de varios campos esenciales que permiten la correcta transmisión y recepción de datos entre los dispositivos dentro de una red. [1]

- ❒ **Preámbulo y Delimitador de Inicio de Trama (SFD):** El preámbulo se utiliza para sincronizar los relojes entre el transmisor y el receptor. El SFD indica el inicio de los datos y ayuda al receptor a identificar el comienzo de la trama.
- ❒ **Campo de Direcciones:** Incluye información sobre el destinatario (como la ID del PAN, dirección corta o larga) y la dirección del remitente.
- ❒ **Campo de Control:** Contiene información crítica sobre el tipo de trama (por ejemplo, datos, solicitud de ACK, etc.) y especificaciones sobre la secuencia de la trama.
- ❒ **Secuencia de Numeración de Trama** Este campo ayuda a identificar tramas únicas y evita la duplicación, asegurando una comunicación más confiable.
- ❒ **Carga Útil (Payload):** Este campo lleva los datos o información que se está transmitiendo.
- ❒ **Secuencia de Redundancia Cíclica (CRC):** Se utiliza para comprobar errores en la trama y garantizar la integridad de los datos durante la transmisión.

1.4.3 ARQUITECTURA DEL PROTOCOLO 802.15.4

El estándar IEEE 802.15.4 opera principalmente en dos capas del modelo OSI (Open Systems Interconnection) como se muestra en la Figura 1.1 [6]:

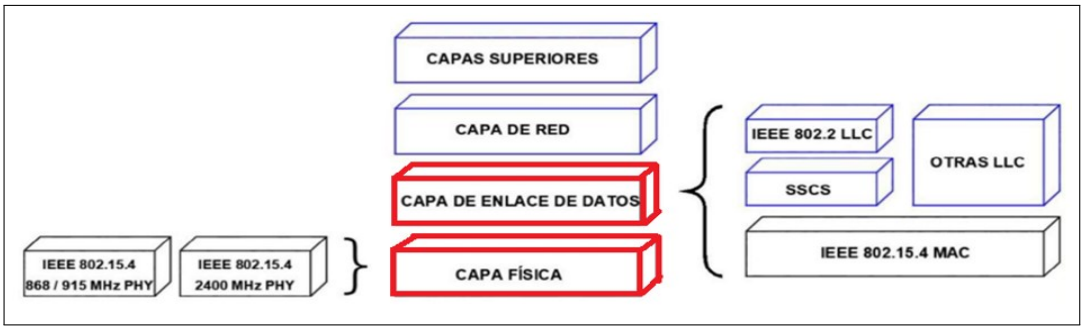

Figura 1.1: Capas del modelo OSI en las que trabaja el protocolo 802.15.4

1.4.3.1 CAPA FÍSICA (PHY - PHYSICAL LAYER)

La capa física (PHY) en el estándar IEEE 802.15.4 desempeña varias funciones esenciales para facilitar la comunicación inalámbrica de dispositivos en redes de área personal de baja potencia (WPAN). Algunas de las funciones principales son:[6]

Definición de Frecuencia y Canal

Frecuencia de Operación: El estándar 802.15.4 puede trabajar en diferentes bandas de frecuencia, como 2.4 GHz, 915 MHz o 868 MHz, dependiendo de la región y las regulaciones locales.

Canalización: Divide el espectro de frecuencia disponible en canales discretos para la transmisión de datos. Define la cantidad de canales, su ancho de banda y su distribución dentro del espectro para evitar interferencias y maximizar la eficiencia espectral.

Modulación

Definen cómo los datos se convierten en señales electromagnéticas para la transmisión inalámbrica. Estos esquemas pueden incluir BPSK (Binary Phase Shift Keying), O-QPSK (Offset Quadrature Phase Shift Keying), o ASK (Amplitude Shift Keying), entre otros. Cada esquema tiene sus ventajas en términos de eficiencia espectral, resistencia al ruido y complejidad de implementación.

Tasas de datos

Especifica la máxima cantidad de datos que se pueden enviar por unidad de tiempo en un canal específico. La tasa de datos puede variar según el esquema de modulación y las condiciones del canal, y se elige según las necesidades de la aplicación y las limitaciones del entorno.

Gestión de potencia

Define niveles de potencia para la transmisión, lo que permite ajustar la intensidad de la señal transmitida por los dispositivos. Esto ayuda a conservar la energía y adaptarse a diferentes distancias de comunicación.

Sincronización

Proporciona mecanismos para la sincronización entre dispositivos dentro de la red. Esto es esencial para asegurar que los dispositivos puedan comunicarse en el mismo período de tiempo y en el mismo intervalo, evitando colisiones y mejorando la eficiencia de la comunicación.

Selección de canal dinámico

Permite la capacidad de cambiar entre diferentes canales para evitar interferencias o congestión en un canal específico. Esto mejora la calidad de la comunicación y la resistencia a las interferencias externas.

Tabla 1.1: Canales de frecuencia que maneja el protocolo 802.15.4 y sus características [5]

Bandas	No de canales	Técnica de ensanchamiento	Modulación	Velocidad del símbolo (kbaud)	Tasa de bits (kbps)
869MHz	1	Binary DSSS	BPSK	20	20
915MHz	10	Binary DSSS	BPSK	40	40
2.4GHz	1	16-array DSSS	O-QPSK	62.5	250

Estas funciones permiten establecer y mantener la comunicación inalámbrica de manera confiable, eficiente y adaptativa en entornos donde se implementan redes IEEE 802.15.4, como aplicaciones de Internet de las Cosas (IoT), sistemas de monitoreo y control, y redes de sensores inalámbricos.

1.4.3.2 CAPA DE CONTROL DE ACCESO AL MEDIO

La Capa de Control de Acceso al Medio (MAC) en el estándar 802.15.4 realiza varias tareas importantes para asegurar que los dispositivos puedan comunicarse de manera efectiva en una red inalámbrica, las cuales están descritas a continuación: [6]

Gestión de Asociación

Controla cómo los dispositivos se unen y se separan de la red. Cuando un dispositivo quiere unirse, la capa MAC maneja este proceso y lo registra en la red. Si un dispositivo quiere dejar la red, la capa MAC se encarga de retirarlo adecuadamente.

Control de Capacidad del Canal

Administra cuánta información puede ser transmitida y recibida en un período de tiempo determinado. Evita que el canal se sature, permitiendo que cada dispositivo tenga oportunidades justas para enviar datos.

Coordinación de Sincronización

Asegura que los dispositivos estén sincronizados en el tiempo para evitar interferencias y colisiones. Esto permite que los dispositivos se comuniquen en intervalos bien definidos, mejorando la eficiencia de la comunicación.

Manejo del Acceso al Canal

Regula cómo los dispositivos acceden al medio compartido para transmitir datos. Utiliza estrategias como “escuchar antes de hablar"para evitar colisiones, asegurando que un dispositivo no empiece a transmitir si detecta que otro está utilizando el canal.

Eficiencia Energética

Implementa técnicas para ahorrar energía, como el modo de bajo consumo, programando tiempos de actividad y de suspensión para los dispositivos. Esto es crucial para prolongar la vida útil de la batería en dispositivos alimentados por energía.

Control de Transmisión y Recepción

Supervisa el proceso de envío y recepción de datos, retransmitiendo en caso de errores y confirmando la entrega de datos. También gestiona situaciones donde varios dispositivos intentan transmitir simultáneamente, resolviendo conflictos.

Seguridad de Datos

Implementa medidas de seguridad como autenticación y cifrado para proteger la información transmitida, asegurando que los datos estén protegidos y no sean accesibles para usuarios no autorizados.

En conjunto, la capa MAC en el estándar 802.15.4 se encarga de orquestar la comunicación entre los dispositivos, garantizando que esta sea eficiente, sincronizada y segura dentro de

una red inalámbrica.

1.4.4 TOPOLOGÍAS DEL ESTÁNDAR 802.15.4

El estándar IEEE 802.15.4 admite varias topologías de red que pueden adaptarse a diferentes escenarios y aplicaciones. Las topologías comunes incluyen: [7]

Red tipo bus

Una topología tipo bus en el contexto de nodos 802.15.4 implica la conexión de múltiples nodos en un solo canal de comunicación compartido, similar a la estructura de un autobús. En este esquema, todos los nodos están conectados a la misma línea de comunicación, y los mensajes transmitidos por un nodo son accesibles para todos los demás nodos en el bus. Cada nodo puede enviar y recibir datos en el canal compartido.

Es importante tener en cuenta que, aunque la topología tipo bus tiene sus beneficios, también presenta desafíos como la posible colisión de tramas y la necesidad de un protocolo eficiente para gestionar el acceso al medio compartido. La elección de esta topología dependerá de los requisitos específicos de la aplicación y de la capacidad para abordar los desafíos asociados.

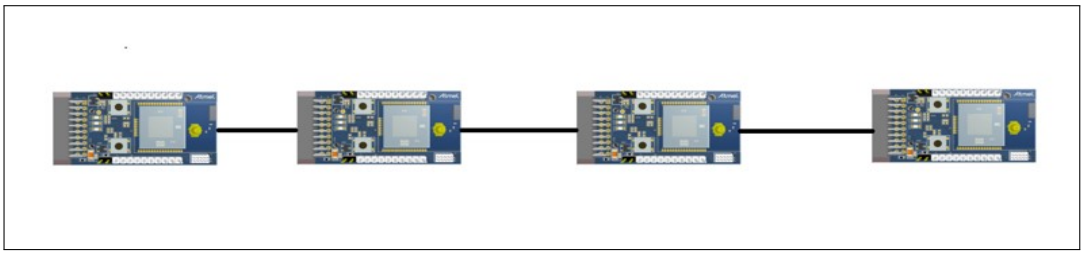

Figura 1.2: Topología tipo bus de nodos 802.15.4

Red en estrella

Una topología en estrella es un diseño de red donde todos los dispositivos están conectados a un nodo central, también conocido como "hub.º çoordinador". En esta configuración, cada dispositivo de la red se comunica directamente con el nodo central, pero no necesariamente con otros dispositivos.

Los componentes de una topología en estrella son:

- **Nodo Central (Coordinador):** Actúa como el punto focal de la red. Todos los dispositivos envían y reciben datos a través de este nodo central.
- **Dispositivos Periféricos:** Están conectados directamente al nodo central, pero generalmente no pueden comunicarse directamente entre sí. Cada dispositivo envía datos

al nodo central, que luego puede distribuir la información a otros dispositivos si es necesario.

Este tipo de topología es ideal para aplicaciones en las que un nodo central coordina la comunicación con múltiples dispositivos periféricos. Es útil en entornos donde se requiere una comunicación directa con un nodo coordinador central, como en sistemas de control de iluminación o monitoreo de sensores.

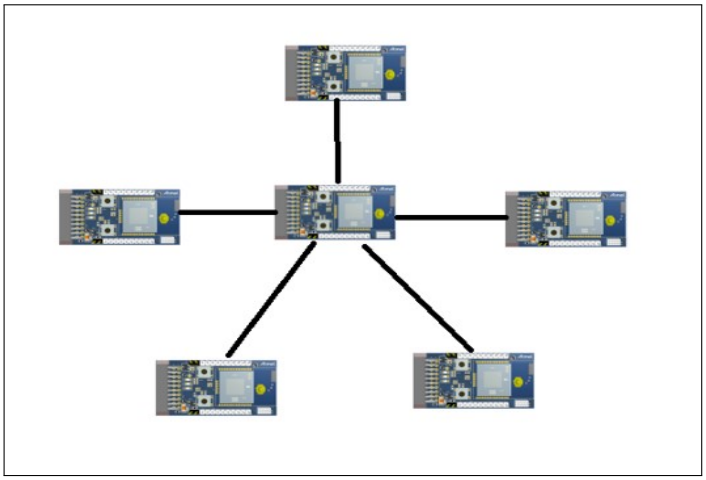

Figura 1.3: Topología tipo estrella de nodos 802.15.4

Red en malla (Mesh Network)

Una topología en malla es un diseño de red donde cada dispositivo está conectado directamente a múltiples otros dispositivos, creando una estructura de comunicación más flexible y redundante. En esta configuración, los dispositivos no solo se conectan al nodo central, sino que también pueden comunicarse entre sí, retransmitiendo datos a través de múltiples rutas posibles.

Esta topología es perfecta para entornos donde se necesita redundancia y flexibilidad en la comunicación. Los dispositivos pueden comunicarse directamente entre sí y retransmitir datos a través de múltiples rutas, lo que mejora la cobertura y la fiabilidad de la red. Esto es útil en aplicaciones de automatización del hogar, sistemas de seguridad y monitoreo industrial.

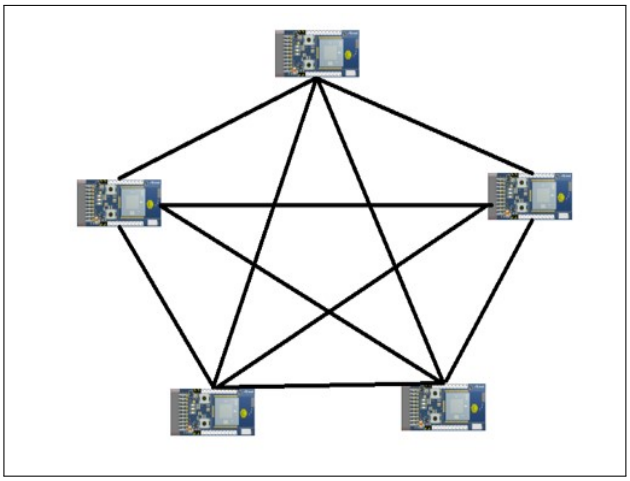

Figura 1.4: Topología tipo malla de nodos 802.15.4

Red en árbol

Una topología en árbol es un diseño de red que se organiza de manera jerárquica, similar a la estructura de un árbol. En esta configuración, los dispositivos se conectan en forma de árbol con un nodo raíz central que se comunica con nodos secundarios o terminales.

Características clave de una topología en árbol:

- **Nodo Raíz:** Actúa como el punto inicial de la red, también conocido como "nodo padre.º ”nodo coordinador". Este nodo central se conecta directamente a los nodos secundarios y no tiene padres.
- **Nodos Secundarios o Terminales:** Están conectados al nodo raíz y pueden ser otros nodos secundarios o finales de la red. Los nodos secundarios pueden tener sus propios nodos hijos conectados a ellos, formando una jerarquía descendente.
- **Comunicación Jerárquica:** Los datos fluyen desde el nodo raíz hacia los nodos secundarios, y estos a su vez pueden comunicarse con sus nodos hijos. La comunicación no ocurre directamente entre nodos a un nivel diferente de la jerarquía.
- **Estructura Ordenada:** La topología en árbol tiene una estructura de control centralizada desde el nodo raíz. Los nodos secundarios pueden comunicarse con el nodo raíz, pero no necesariamente entre sí directamente.
- **Aplicaciones de Control Jerárquico:** Se utiliza en aplicaciones donde se necesita una comunicación en cascada o en una dirección específica. Ejemplos comunes incluyen sistemas de control de edificios, redes de monitoreo ambiental o sistemas de distribución de datos en una estructura organizada.

Adecuada para aplicaciones donde se requiere una comunicación jerárquica, como siste-

mas de control en edificios, donde un nodo central se comunica con nodos secundarios y terminales. También es útil en aplicaciones de monitoreo ambiental o agrícola, donde se necesitan múltiples niveles de recolección y procesamiento de datos.

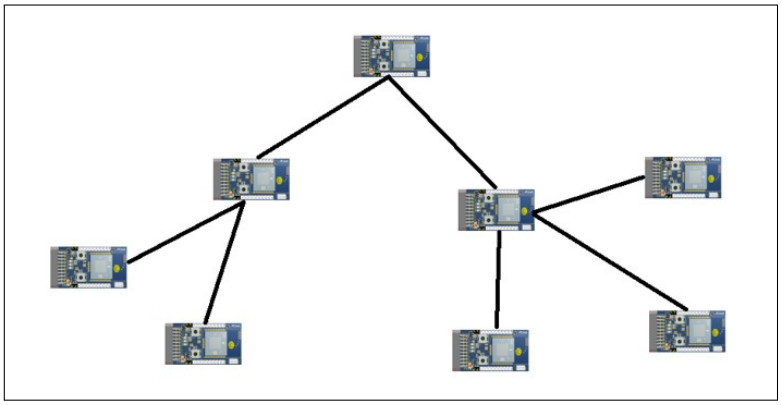

Figura 1.5: Topología tipo árbol de nodos 802.15.4

Cada topología tiene sus propias ventajas y se adapta a diferentes entornos y necesidades. La elección de la topología adecuada depende de factores como la distancia entre dispositivos, la cantidad de dispositivos en la red, la confiabilidad requerida y el nivel de control centralizado necesario.

1.4.5 TRANSCEPTOR ATZB-256RFR2-XPRO

El transceptor ATZB-256RFR2-XPRO es un módulo de comunicación inalámbrica basado en el estándar IEEE 802.15.4 y el protocolo Zigbee, diseñado para permitir la conectividad y la comunicación inalámbrica en aplicaciones IoT, redes de sensores, domótica y otros sistemas de control.

Este dispositivo incluye un microcontrolador integrado junto con un transceptor de radiofrecuencia (RF) que opera en la banda de frecuencia de 2.4 GHz. Algunas características principales del ATZB-256RFR2-XPRO son: [8]

Microcontrolador:

- ❒ Utiliza un microcontrolador AVR de 8 bits de baja potencia y alto rendimiento.
- ❒ Frecuencia de reloj de hasta 16 MHz.
- ❒ Memoria Flash de 256 KBytes para almacenamiento de código.
- ❒ RAM de 32 KB para almacenamiento temporal de datos.
- ❒ EPROM de 8 KB para almacenamiento no volátil

Transceptor RF:

- ❒ Opera en la banda de frecuencia de 2.4 GHz.
- ❒ Cumple con el estándar IEEE 802.15.4.
- ❒ Soporta el protocolo Zigbee, que proporciona un marco para la creación de redes inalámbricas de baja potencia y control de dispositivos.
- ❒ Ofrece opciones de modulación como BPSK y O-QPSK.

Interfaces y Conectividad:

- ❒ Interfaces UART, SPI y I2C para comunicación con otros dispositivos.
- ❒ Antena integrada.
- ❒ Conector para antena externa.

Alimentación:

- ❒ Requiere una fuente de alimentación de 2.1V a 3.6V.

Dimensiones y diseño:

- ❒ Diseño compacto y adecuado para integración en sistemas y prototipos.
- ❒ Compatible con kits de desarrollo Xplained Pro de Microchip.

Características Adicionales:

- ❒ Consumo de energía optimizado para aplicaciones de bajo consumo.
- ❒ Diseñado para facilitar el desarrollo de aplicaciones IoT y de control.

Estas características hacen del ATZB-256RFR2-XPRO una opción versátil para la implementación de comunicaciones inalámbricas basadas en IEEE 802.15.4 y Zigbee, ofreciendo conectividad confiable y eficiente para una variedad de aplicaciones en el ámbito de IoT, redes de sensores, automatización y control.

Figura 1.6: Nodo ATZB-256RFR2-XPRO

2 METODOLOGÍA

En el presente capítulo se describe el problema al momento de buscar herramientas de medición de parámetros de equipos inalámbricos WPAN, más específicamente la tasa de tramas perdidas del protocolo 802.15.4, se explica también el desarrollo de los algoritmos para los objetivos propuestos y su implementación mediante el lenguaje C explicando detalladamente los códigos desarrollados.

2.1 EXPOSICIÓN DEL PROBLEMA

El crecimiento exponencial en la adopción de redes inalámbricas de corto alcance ha generado un incremento en la demanda de herramientas de medición de manera específicas para evaluar los parámetros críticos de estos sistemas dando así con una respuesta de insatisfacción en los mismo ya que este crecimiento exponencial ha desencadenado la proliferación de sensores inalámbricos, que suelen ser utilizados en una variedad de aplicaciones como IoT hasta sistemas de monitoreo industrial. Por ende, la falta de herramientas estandarizadas y accesibles para medir con precisión parámetros vitales, como la tasa de tramas perdidas, la calidad de la señal, la duración de batería, entre otros. Existen muchas tecnologías de redes inalámbricas de alcance personal WPAN ya sea, Bluetooth, Zigbee, 802.15.4, Lora, etc [4]. Cada una de estas tecnologías tiene sus características propias, pero al no conocer los parámetros mencionados anteriormente, no podemos hacer un análisis acerca de qué tecnología es más fiable en el desarrollo de un proyecto u otro. Es por eso que se necesitan herramientas para la medición de estos parámetros de las distintas tecnologías para encontrar los límites del correcto funcionamiento de cada uno.

Ante esta problemática, se plantea el desarrollo de una herramienta de medición especializada que permita evaluar la tasa de tramas perdidas en transmisiones bajo el protocolo 802.15.4. El propósito central de esta iniciativa radica en realizar un análisis detallado de los resultados obtenidos, con el fin de proporcionar una comprensión más profunda de las limitaciones y desafíos inherentes a estos sensores. Se aspira que este análisis contribuya a la mejora de futuros proyectos que involucren sensores inalámbricos, permitiendo así un desarrollo más eficiente y confiable de sistemas más amplios basados en esta tecnología.[9]

2.2 DISEÑO DE LA HERRAMIENTA

El sistema que desarrollaremos se compone de los elementos mostrados a continuación:

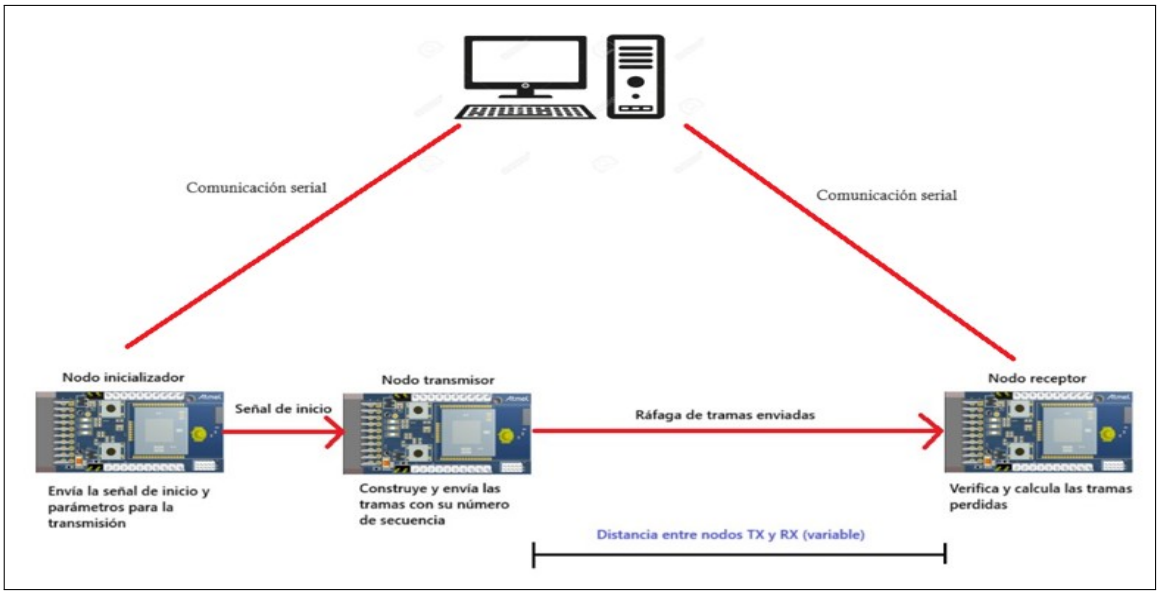

Figura 2.1: Diseño del sistema para detectar FER

Como se mencionó anteriormente, la herramienta que se desarrollará esta pensado para un sistema de tres nodos de los cuales dos de ellos estan conectados hacia la PC por puerto serial para poder enviar los parámetros desde la PC hacia el nodo iniciador y para que el nodo receptor pueda enviar las tramas guardadas de igual manera por puerto serial.

2.3 DISEÑO DEL ALGORITMO

Para este proyecto se propone una red de comunicación punto a punto basada en el estándar 802.15.4, se establece una estructura con tres nodos principales: un transmisor, un receptor y un nodo inicializador. Esta configuración permite una comunicación eficiente y controlada entre los dispositivos. El nodo inicializador desempeña un papel fundamental al enviar una señal de inicio al transmisor para activar el proceso de transmisión de tramas hacia el receptor.

El proceso comienza con el nodo inicializador, encargado de iniciar la secuencia de transmisión. Al enviar un mensaje de inicio al nodo transmisor, activa el proceso de envío de tramas 802.15.4 hacia el receptor. Este mensaje de inicio puede ser una señal específica o un comando que notifica al transmisor que está listo para recibir datos.

El nodo transmisor, al recibir la señal de inicio, comienza a generar tramas de datos con información relevante, como el nivel de batería, potencia de transmisión y otros datos pertinentes. Cada trama se etiqueta con un identificador único o número de secuencia para permitir la verificación de la integridad de la transmisión.

Estas tramas son enviadas a una frecuencia determinada desde el transmisor hacia el re-

ceptor a través del canal de comunicación establecido. El receptor, al recibir las tramas, procede a verificar la secuencia de cada una de ellas. Utiliza el identificador único incluido en las tramas para detectar posibles pérdidas durante la transmisión.

Una vez que el receptor ha recibido todas las tramas programadas, inicia un proceso de verificación. Compara los identificadores de secuencia de las tramas recibidas con la secuencia esperada para determinar si hay tramas faltantes. Este análisis permite calcular la tasa de tramas perdidas y precisar qué trama específica no fue recibida.

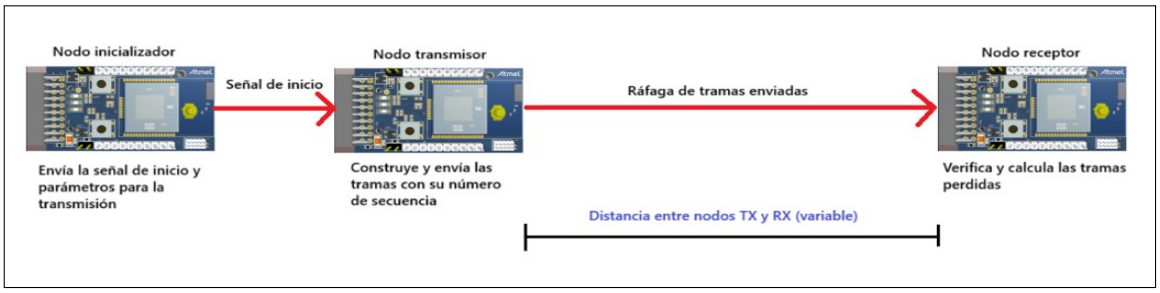

Figura 2.2: Proceso del algoritmo para detección de tramas perdidas

2.3.1 ALGORITMO DEL NODO TRANSMISOR

El diseño para este algoritmo tiene como objetivo el envío de tramas que lleven información la cual podamos leer desde el nodo receptor, para estos datos lo que haremos es tomar datos del nodo transmisor como el nivel de batería actual, nivel de potencia de transmisión de la trama y un numero de secuencia, además de agregar datos aleatorios para llenar el campo del payload, todos estos datos se guardaran en este campo en un formato de vector de bytes para poder realizar el envío.

Para este algoritmo se debe definir la cantidad de tramas que se van a enviar, debemos definir el tipo de trama, ya sea trama larga (vector de 100 posiciones) o trama corta (vector de 23 posiciones) además del tiempo entre tramas.

2.3.1.1 OBTENCIÓN DE INFORMACIÓN DEL NODO TRANSMISOR

Nivel de batería

1. **Lectura del sensor de batería:** Los nodos 802.15.4 que utilizaremos cuentan con un sensor de batería integrado, del cual podemos obtener su información utilizando la función *get_bat_sensor_data()*.

2. **Conversión de datos:** Los datos obtenidos se guardan en una cadena float los cuales

tenemos que cambiar a formato de cadena de caracteres hexadecimales *uint_t*, los cuales se guardarán en el vector de datos del payload en las posiciones *[7], [8], [9] y [10]*.

Nivel de potencia de transmisión de la trama

1. **Seteo del nivel de potencia:** Para poder realizar la transmisión debemos setear el valor de la potencia de transmisión de la trama el cual se guarda en formato entero en la variable *int PotenciaTransmision*.

2. **Conversión de datos:** Como se mencionó en el apartado anterior el formato de la potencia se guarda en una variable tipo ”int”, estos valores para poder ser transmitidos tienen que ser transformados a formato de vector de caracteres hexadecimales *uint_t* los cuales se guardarán en el vector de datos del payload en las posiciones *[11], [12], [13]*.

Número de secuencia y tipo de trama

1. **Incremento del número de secuencia:** Para este valor lo que realizaremos es inicializar el contador *numeroSecuencia* en 0 para de esta manera iniciar el bucle y después de cada trama enviada este valor se irá aumentando en 1.

2. **Conversión de datos:** Estos datos al ser enteros se tienen que transformar al formato *uint_t* para guardarlo en el vector de datos del payload en este caso se guardarán en las posiciones *[1]-[6]*.

3. **Tipo de trama:** Para elegir el tipo de trama que se enviará tenemos que setear la variable *TipoTrama* en valor 1 o 2, el cual entrará en un comparador y si el valor seteado es 1 será la trama corta la cual define la longitud de trama en 23 bytes y guardaremos el caracter ”C” en formato *uint_t* en la posición *[0]* del payload. Si el valor seteado es 2 será la trama larga la cual define la longitud de trama en 100 bytes y guardaremos el caracter ”L” en formato *uint_t* en la posición *[0]* del payload..

El resto de las posiciones del payload se guardarán con valores aleatorios seteados al principio del código.

2.3.1.2 CREACIÓN Y ENVÍO DE TRAMAS

Una vez obtenidos los datos que se enviarán se debe construir la trama, esto lo haremos con la función *transmit_sample_frame(datosPayload, longitudTrama)* el cual setea el vector de datos del payload y la longitud de la trama, con esta función además de la construcción de la trama también envía la misma.

Para el control de la frecuencia de transmisión de las tramas tenemos que definir un tiempo de delay el cual pausará el bucle de envío de tramas el tiempo que se haya definido.

2.3.1.3 ENVÍO CONTÍNUO DE TRAMAS

Todo el proceso mencionado anteriormente funciona dentro de un bucle el cual va comparando el valor de la variable *iniciarTransmision* que tiene un valor por defecto de 0, una vez que esta variable cambia a 1 inician las funciones para obtener la información y el envío de las tramas, cuando se envía cada trama va aumentando +1 el valor de la variable *numeroSecuencia* y va reduciendo -1 el valor de la variable *contTX* la cual inicialmente tiene el número de tramas totales que vamos a enviar.

Este proceso se repetirá las veces requeridas hasta que consecutivamente la variable *contTX* llegue a 0, una vez que esta variable llega a este valor se envía una última trama indicando el final de la transmisión.

A continuación, se muestra el diagrama de flujo del algoritmo descrito anteriormente para la transmisión de datos.

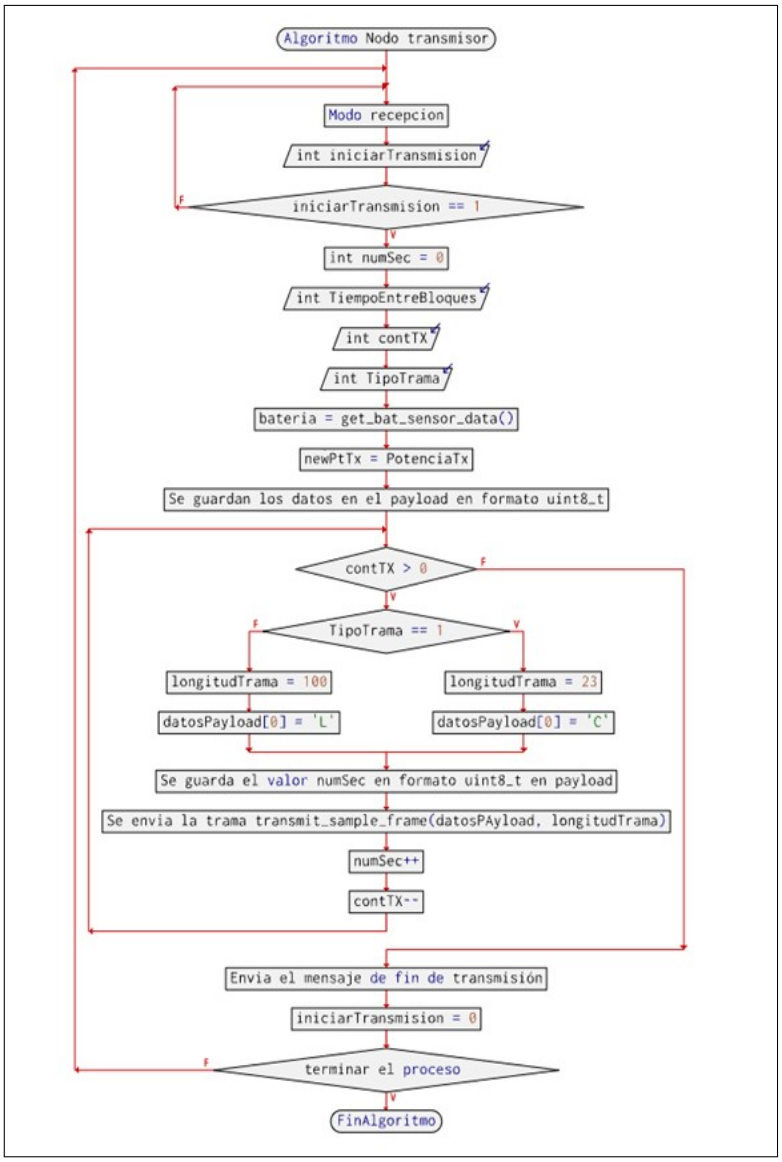

Figura 2.3: Diagrama de flujo del nodo transmisor

2.3.2 ALGORITMO DEL NODO RECEPTOR E INICIADOR

En principio se plantea realizar el proyecto con 3 nodos, transmisor, receptor e iniciador. Se encontró una solución para optimizar el diseño y unificar las funciones del nodo receptor y el nodo iniciador en uno solo, ya que el iniciador solo tiene que enviar tramas definidas para que luego se inicie la transmisión de las tramas.

Este algoritmo une estas dos funciones en un solo código.

El nodo tiene que estar conectado a la PC de manera que se tenga comunicación serial, en primera instancia el nodo funciona como iniciador, ya que, por medio de la PC se ingresan los parámetros que tendrá la transmisión (número de tramas, delay entre tramas, etc), una vez ingresados estos datos se transmiten por el puerto serial hacia el nodo el cual después

de recibir esta información, envía las tramas con dicha información hacia el nodo transmisor, el cual, al recibir los parámetros inicia la transmisión de las tramas. Las tramas serán enviadas hacia el mismo nodo iniciador, el cual, ahora toma la función de receptor, cada trama que se recibe, se va almacenando en un vector de bytes en el orden que le llegan las distintas tramas.

Una vez finalizada la transmisión, todos estos datos almacenados en el nodo, son enviados por puerto serial hacia la PC.

Se muestra el diagrama de flujo del algoritmo a continuación:

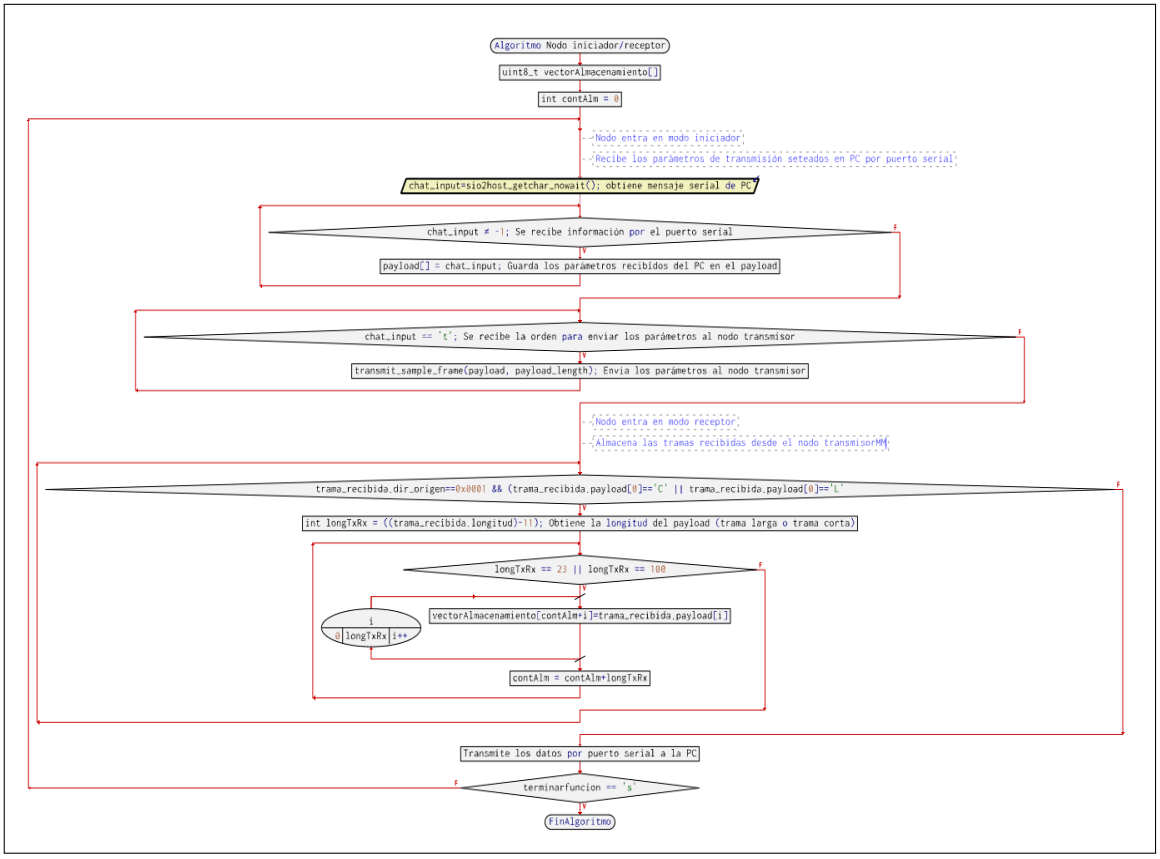

Figura 2.4: Diagrama de flujo del nodo receptor e iniciador

2.3.3 ALGORITMO DEL SOFTWARE PARA LA PC

El algoritmo que se desarrollará para el programa de la PC puede ser descrito en los siguientes pasos:

1. **Configuración de parámetros de transmisión:** El programa en la PC recibe los parámetros necesarios para la transmisión de tramas. Estos parámetros pueden incluir información como la dirección del nodo destino, el tipo de trama a enviar, el contenido de la trama, entre otros. Esta configuración se utiliza para preparar los datos que se

enviarán a través del puerto serial hacia el nodo 802.15.4.

2. **Transmisión de tramas hacia los nodos 802.15.4:** Una vez que se han configurado los parámetros, el programa envía los datos a través del puerto serial hacia el nodo 802.15.4. Estos datos pueden incluir las tramas que se desean enviar a uno o varios nodos dentro de la red 802.15.4. El nodo receptor procesará estas tramas según lo programado y realizará las acciones correspondientes.

3. **Recepción de tramas desde el nodo receptor:** Después de que el nodo receptor ha recibido y procesado las tramas, se envía una orden desde la PC para que el nodo que recibió las tramas envíe de vuelta todas las tramas recibidas por puerto serial hacia la PC. Esto puede implicar que el nodo recolecte las tramas recibidas y las envíe a través de su propio puerto serial hacia la PC.

4. **Generación de archivo Excel:** Una vez que la PC ha recibido todas las tramas de vuelta desde el nodo, el programa las procesa y genera un archivo Excel. Cada trama recibida se puede representar como una fila en la hoja de cálculo, con columnas que muestran los diferentes campos de la trama, como el tipo de trama, número de secuencia, nivel de batería, potencia de recepción y el resto de la carga útil de la trama. Cuando se genere el archivo con las tramas se genera un contador el cual va aumentando por cada trama almacenada, este contador representa las tramas recibidas, con este valor una vez guardadas todas las tramas en el archivo se agrega una fila el cual detalle las tramas enviadas, las tramas recibidas y el porcentaje de tramas perdidas.

En resumen, este algoritmo establece una comunicación bidireccional entre una PC y nodos 802.15.4, permitiendo la transmisión de tramas desde la PC a los nodos y la recepción de tramas desde los nodos de vuelta a la PC, seguido de la generación de un archivo Excel para análisis posterior.

Se muestra el diagrama de flujo del algoritmo a continuación:

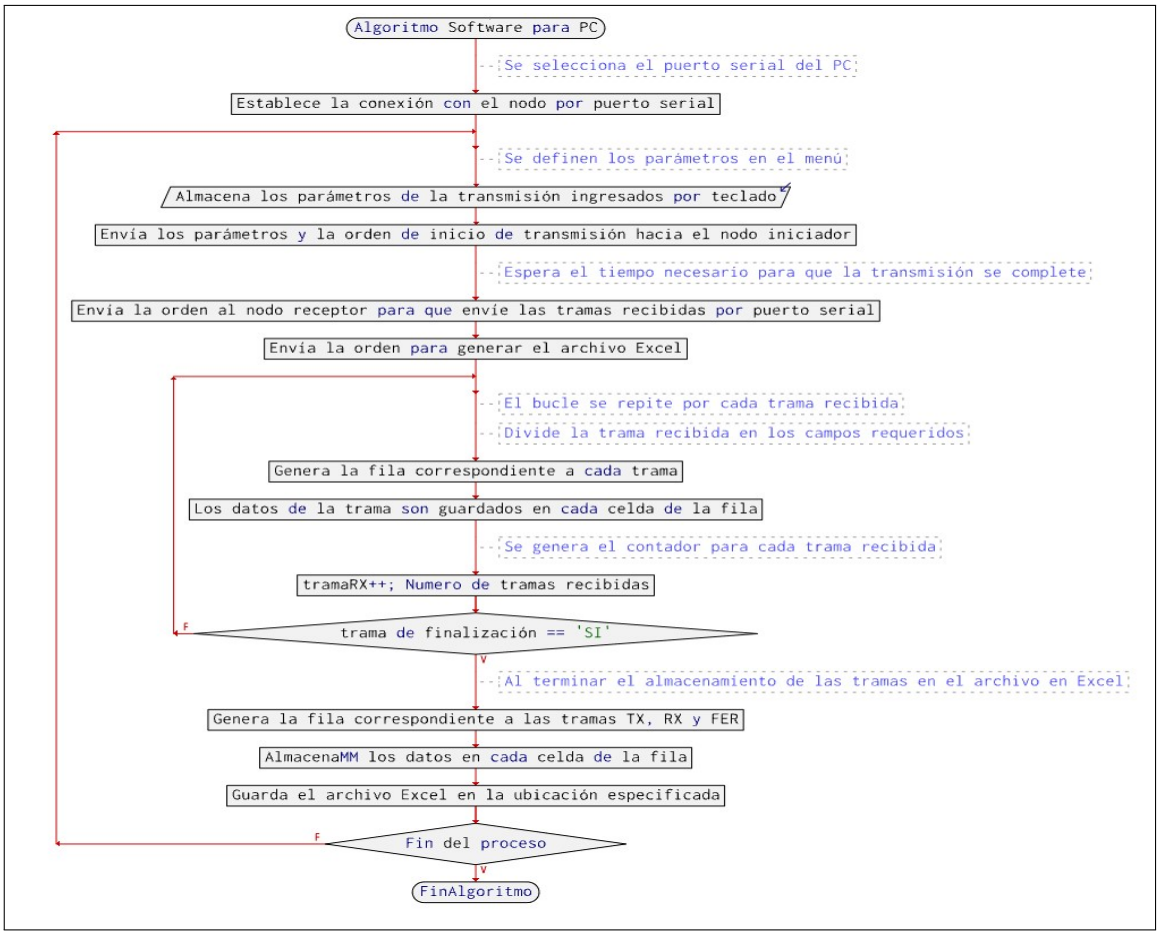

Figura 2.5: Diagrama de flujo del programa para la PC

2.3.4 ELEMENTOS NECESARIOS PARA LA IMPLEMENTACIÓN DEL ALGORITMO

Para la implementación de este algoritmo utilizaremos los nodos ATZB-256RFR2-XPRO el cual es compatible con el estándar 802.15.4 con su fuente de alimentación, pero también para el desarrollo de los códigos utilizaremos el software Atmel Studio 7.0 en el que se programa en lenguaje C.

Tabla 2.1: Materiales necesarios para el desarrollo del proyecto

HARDWARE		
Herramienta	**Qty**	**Imagen**
Computador	1	
Nodo ATZB-256RFR2-XPRO	2	
Interfaz serial USB RS-232	1	
SOFTWARE		
Windows 11	1	
Atmel Studio 7.0	1	
PyCharm Community Edition	1	

2.3.5 CODIFICACIÓN DE LOS ALGORITMOS

En esta sección se describe el proceso para inicializar un proyecto para los nodos 802.15.4 en el software ATMEL STUDIO 7 para continuar con la explicación de los códigos desarrollados.

2.3.5.1 Inicialización del proyecto

Debemos abrir el programa mencionado en nuestro computador, al realizar esta acción se despliega la ventana de inicio del programa en el cual debemos escoger la opción **New Project**.

Figura 2.6: Ventana de inicio de programa ATMEL STUDIO 7

Una vez se despliega la siguiente ventana después de haber realizado el paso anterior en la pestaña ”Installed” seleccionamos la opción ”C/C++” que nos sirve para crear un proyecto de aplicación AVR 8-bit XC8, es decir, un proyecto con lenguaje C.

Luego de esto se nos despliegan los tipos de proyecto que podemos crear, en el que seleccionaremos la opción ”GCC C IEEE 802.15.4 MAC Wireless Executable Project” el cual especifica que el proyecto será desarrollado con el compilador GCC para lenguaje C, para dispositivos inalámbricos que implementan el estándar 802.15.4. Se agrega el nombre del proyecto y ubicación para al final seleccionar el botón ”OK”.

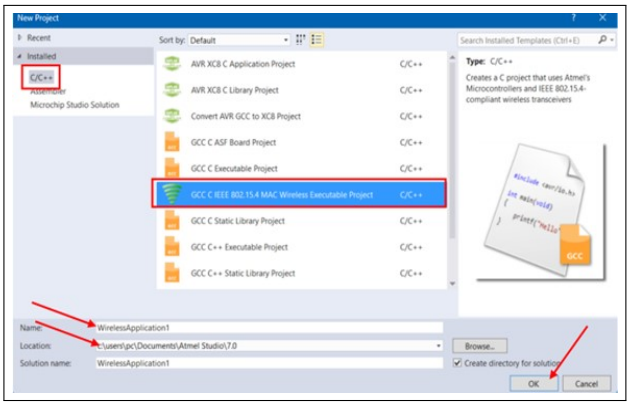

Figura 2.7: Opciones a escoger en ventana New Project

Se despliega la siguiente imagen en la cual debemos seleccionar la opción”Generic Project” con el que cargamos un código por defecto y poder elegir el dispositivo exacto sobre el que correremos el código.

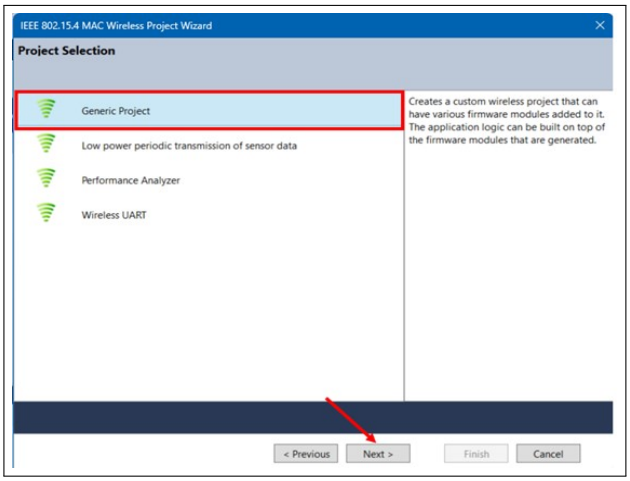

Figura 2.8: Ventana para escoger opciones de un proyecto de comunicación inalámbrica

En la siguiente ventana tenemos que elegir el microcontrolador que contiene nuestro nodo, en este caso, los nodos ATZB-256RFR2-XPRO contienen el microcontrolador Atmega256RFR2.

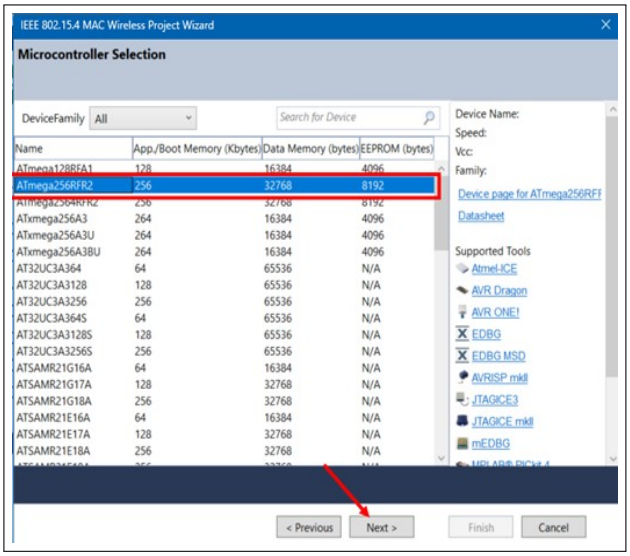

Figura 2.9: Opciones para escoger el microcontrolador del nodo a programar

Luego de esto se presenta una ventana en la que podemos escoger las características del nodo, aquí podemos definir valores como la frecuencia, la velocidad de transmisión, el ID del nodo y el ID de red.

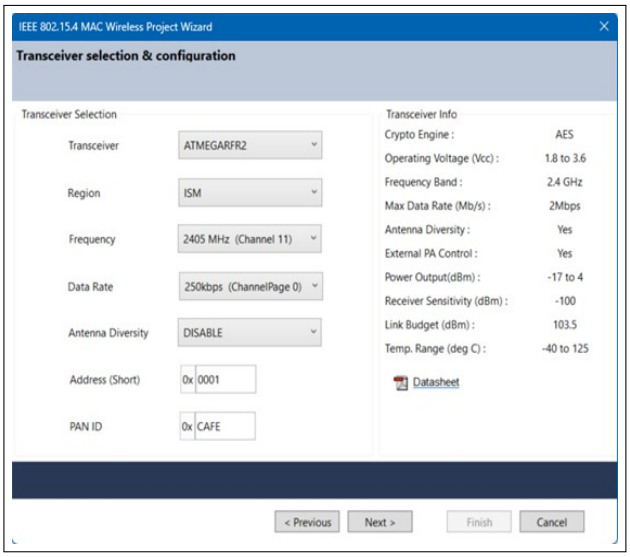

Figura 2.10: Selección de características de la comunicación

En la siguiente ventana se nos presenta las configuraciones de transmisión y recepción. Para este proyecto al tener una comunicación punto a punto en la opción de ”Transmit data” elegimos ”Unicast” ya que solo transmitiremos hacia el nodo receptor, en la dirección del receptor se coloca el ID del nodo que recibirá las tramas.

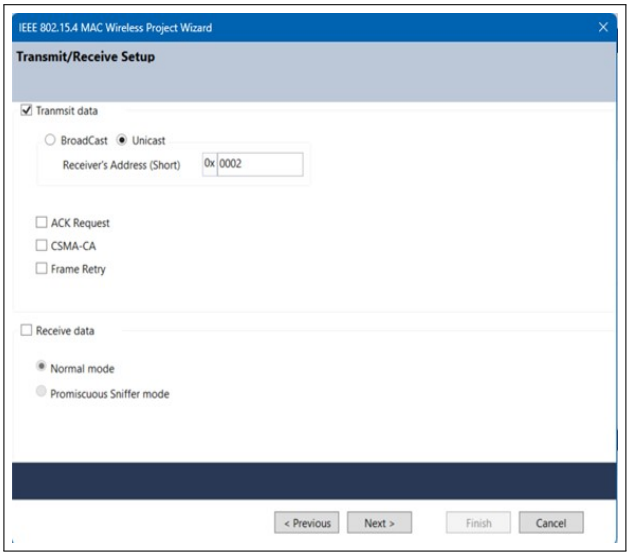

Figura 2.11: Definición de parámetros de los nodos

2.3.5.2 Programación del proyecto en el software

Una vez realizados todos estos pasos se nos carga un proyecto genérico con los archivos necesarios para una transmisión inalámbrica bajo el estándar 802.15.4 que son descritos a continuación:

- ❒ *main.c* Archivo donde se convocan las funciones programadas para que actúen bajo

un bucle el cual se ejecuta de manera continua para la transmisión y recepción del nodo.

- ❒ Archivos del usuario: Los archivos usr_ en donde se codifican las funciones que serán convocadas en el main.
- ❒ Archivos ”Wireless”: Son los archivos que contienen las características del modelo del transceptor, aquí se cargan los datos generados por la configuración inicial, pero pueden ser cambiados en este archivo, datos como ID del nodo, ID de la red, frecuencias.

Estos archivos se muestran en la siguiente imagen:

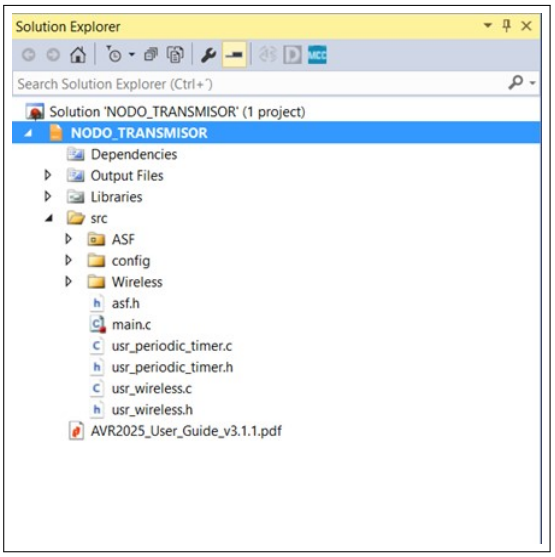

Figura 2.12: Archivos necesarios para la programación de la comunicación inalámbrica

Librerías utilizadas

El uso de librerías en programación es sumamente importante ya que cada librería contiene funciones y rutinas predefinidas las cuales nos facilitan el proceso de desarrollo, en este caso utilizamos las funciones Wireless para manipular la transmisión inalámbrica, y las funciones math.h y stdio.h para la manipulación de los archivos requeridos, estas librerías serán utilizadas en los códigos del nodo transmisor y receptor/iniciador.

Código 2.1: Librerías utilizadas para la transmisión y recepción

```
1     //Librerias utilizadas
-  #include "usr_wireless.h"  //Función para inicializar el nodo y sus funciones
-  #include "wireless_config.h" //Brinda el acceso a los archivos que contienen las
       características de la transmisión
-  #include "math.h" //Permite realizar operaciones matemáticas básicas
5  #include "stdio.h" //Permite la manipulación de archivos de entrada y salida
```

```
#include "sio2host.h" //Contiene las características necesarias para una
    comunicación serial
```

Función main

Como se mencionó anteriormente esta es la función principal del código, aquí se genera el bucle el cual funcionará continuamente para que el nodo transmita y reciba datos. La función main será la misma para los códigos de los diferentes nodos que programaremos.

Código 2.2: Función main para la comunicación inalámbrica

```
 * \Función principal de la aplicación
 */
int main(void)
{
    /* Inicializa los módulos inalámbricos */
    wireless_init();

    /*Se inicializan los módulos seleccionados en el asistente */
    modules_init();

    while (1)
    {
        WirelessTask(); //Realiza las tareas inalámbricas del nodo
    }
}
```

Función usr_wireless.h

En este archivo se crea la estructura en donde se va a almacenar las tramas del estándar 802.15.4, la estructura se explica a continuación:

Código 2.3: Función usr_wireless.h para la comunicación inalámbrica

```
 #ifndef USR_WIRELESS_H_
#define USR_WIRELESS_H_

#include "wireless_api.h"
#include "tfa.h"
#define max_dato 101
typedef struct
{
    uint8_t longitud;       // Longitud de Trama
```

```
    uint16_t fcf;       // fcf (campo de control de trama)
    uint8_t num_sec;        // Numero de secuencia
    uint16_t dir_PAN;       // Direccion PAN
    uint16_t dir_destino;   // Direccion ORIGEN
    uint16_t dir_origen;        // Direccion DESTINO
    uint8_t cargaUtil_802_15_4[max_dato];           // Carga útil
    uint16_t fcs;             // fcs (campo de control de secuencia)
} trama_ieee_802_15_42;

trama_ieee_802_15_42 trama_recibida;
```

Esta estructura se utilizará en la codificación de los 3 nodos, por lo cual el archivo será en mismo en los 3 proyectos.

Función usr_wireless.c

En este archivo es donde codificamos las funciones que va a realizar nuestro nodo en el proceso de transmisión y recepción, para que, al implementarlo, estas funciones sean llamadas en la función main del proyecto.

Este archivo es el que variará según el nodo que se implemente ya que cada uno tendrá su función específica para que la herramienta funcione.

Nodo transmisor

Para desarrollar el código primero se definen las librerías que utilizaremos, en este caso se usarán las mencionadas en un apartado anterior. Una vez definidas las librerías tenemos que definir las variables que vamos a utilizar en nuestro algoritmo.

Las variables se describen a continuación:

Código 2.4: Definición de variables del programa para el nodo transmisor

```
//Variables utilizadas
int obtenerNB=1;//Variable para acceder al nivel de batería del nodo transmisor
uint8_t voltajeBateria[7]; //Vector para guardar el voltaje de la batería
uint8_t datosPayload[100]={-2, '1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
    '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1
    ', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
     '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '
    1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1'
    , '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
    '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1'}; //Información a
```

```
    enviar en el payload
uint16_t numeroSecuencia=0; //Numero de secuencia de las tramas

int NumTransmiciones=0; //Número de veces que se envían las tramas
int contTX=0; //Contador de control de envío de tramas
int TiempoEntreTramas=200; //Tiempo de delay entre tramas

int NumeroBloques=0; //Numero de bloques de tramas que se envían
int contBL=0;//Contador de control de envío de bloques
int indicadorBL=0;
int TiempoEntreBloques=200;//Tiempo de delay entre bloques

int PotenciaTransmision=0;//Valor de potencia con el que se van a transmitir las
    tramas

int TipoTrama=0;//1 --> trama corta; 2 --> trama larga
int iniciarTransmision=0; ento
int longitudTrama=0; //Define la longitud de la trama
int escucharParametros=0; //Variable para que inicie a recibir los parámetros del
     nodo iniciador
```

Una vez declaradas las variables el nodo entra en el modo recepción, el cual consiste en que el nodo recibirá tramas y comparará la información que se recibe, si el nodo recibe la trama desde el ID del nodo inicializador empieza a leer toda la trama, de la cual, obtendrá los parámetros para la transmisión de las tramas consecutivas, los parámetros serán:

- ❒ Número de tramas que se enviarán.
- ❒ El tiempo de delay entre tramas.
- ❒ Cuantos bloques de tramas se enviarán.
- ❒ El tiempo de delay entre bloques.
- ❒ La potencia de transmisión.
- ❒ El tipo de tramas que se enviarán

Luego de definirse los parámetros para la transmisión, la variable *Iniciartransmision* toma el valor de 1 para iniciar a armar las tramas.

Código 2.5: Algoritmo que define el estado de recepción del nodo a la espera de los parámetros de la transmisión

```
void usr_frame_received_cb(frame_info_t *frame)
//Funcion de recepción del nodo
{
    memset(&trama_recibida,0,sizeof(trama_recibida));
    // Elimina informacion previa de la estructura de recepcion
    memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));
    //Almacena en una estructura la trama 802.15.4 recibida
    bmm_buffer_free(frame->buffer_header);
    //limpia el buffer de recepción

    if (trama_recibida.dir_origen==0x0002)
    // distingue si la trama es de origen del nodo controlador
    {
        uint8_t iniciarRx[3];
        //permite almacenar las ordenes de la aplicación
        iniciarRx[0]=(uint8_t)trama_recibida.cargaUtil_802_15_4[0];
        iniciarRx[1]=(uint8_t)trama_recibida.cargaUtil_802_15_4[1];
        iniciarRx[2]=(uint8_t)trama_recibida.cargaUtil_802_15_4[2];
        // almacena los tres primeros valores de la carga util
        //////////////////////////debe cumplir esta condición para escuchar las
            ordenes de la siguiente trama
        if ( iniciarRx[0]=='i' && iniciarRx[1]=='n' && iniciarRx[2]=='i' )
        //Compara si los 3 primeros valores de la trama recibida es "ini" para
            recibir las órdenes
        {
            escucharParametros=1;
            //Inicia a recibir los parámetros de la transmisión

        }
        if (escucharParametros==1)
        //Si el nodo empieza a recibir los parámetros
        {
            int longitudTramaRx=((int)trama_recibida.longitud)-11;
            //Define la longitud de la trama
            char ord=trama_recibida.cargaUtil_802_15_4[longitudTramaRx-1];
            //Parámetro que será definido

            if (ord=='a')
            //Si la orden es "a" se define el número de transmisiones
```

```
            {
                //LED_Toggle(LED0);
                NumTransmiciones = atoi(trama_recibida.cargaUtil_802_15_4);
                //Guarda el valor de tramas que se enviarán
                //contParametros++;
            }
            else if(ord=='b')
            //Si la orden es "b" se define el tiempo entre tramas
            {
                //LED_Toggle(LED1);
                TiempoEntreTramas = atoi(trama_recibida.cargaUtil_802_15_4);
                //Guarda el valor del tiempo entre tramas
            }
            else if(ord=='c')
            //Si la orden es "c" se define el número de bloques de tramas que se
                enviarán
            {
                NumeroBloques = atoi(trama_recibida.cargaUtil_802_15_4);
                //Guarda el valor del número de bloques a transmitir
            }
            else if(ord=='d')
            //Si la orden es "d" se define el tiempo de delay entre bloques
            {
                TiempoEntreBloques = atoi(trama_recibida.cargaUtil_802_15_4);
                TiempoEntreBloques =TiempoEntreBloques*1000;
                // Guarda el valor del tiempo entre bloques
            }
            else if(ord=='e')
            //Si la orden es "e" se define el valor de la potencia de transmisión
            {
                //LED_Toggle(LED2);
                PotenciaTransmision = atoi(trama_recibida.cargaUtil_802_15_4);
                //Guarda el valor de la potencia de transmisión
            }
            else if(ord=='f')
            //Si la orden es "f" se define el tipo de trama que se enviarán
            {
                TipoTrama = atoi(trama_recibida.cargaUtil_802_15_4);
                //Guarda el identificador del tipo de trama
            }
            ////////////////////////////////////Deja de escuchar órdenes
            else if(ord=='n')
```

```
            //Si la orden es "n" se finaliza la función de receptor del nodo
            {
                //LED_Toggle(LED0);
                uint8_t noEscuchar[3];
                noEscuchar[0]=(uint8_t)trama_recibida.cargaUtil_802_15_4[0];
                noEscuchar[1]=(uint8_t)trama_recibida.cargaUtil_802_15_4[1];
                noEscuchar[2]=(uint8_t)trama_recibida.cargaUtil_802_15_4[2];
                // almacena los cuatro primeros valores de la cadena en el vector
                    N
                if (noEscuchar[0]=='f' && noEscuchar[1]=='i' && noEscuchar[2]=='n
                    ')
                //Compara si los 3 valores de la trama recibida es "fin" para
                    dejar de recibir órdenes
                {
                    LED_On(LED1);
                    LED_On(LED2);
                    escucharParametros=0;
                    //Deja de recibir órdenes
                    //contParametros==0;
                    iniciarTransmision=1;
                    //Inicia la transmisión de las tramas
                    contTX=NumTransmiciones;
                    //Guarda la variable
                    contBL=NumeroBloques;
                    //Guarda la variable
                    //max=0;
                }
            }
        }
    }
}
```

Para iniciar a armar las tramas que se transmitirán, como se definió en el diagrama de flujo, se obtienen los valores de batería y potencia de transmisión del nodo:

Código 2.6: Obtención de datos de nivel de batería y potencia TX del nodo transmisor

```
obtenerNB=0;
        // Obtener nivel de Bateria
            ////////////////////////////////////////////////////////////
        char charBateria[20]; // Almacena el nivel de la batería convertido en
            una cadena de caracteres
```

```
        float floatBateria=get_bat_sensor_data();// Obtiene en nivel de batería
        ftoa(floatBateria, charBateria, 4);// Convierte el nivel de batería de
            float a una cadena de caracteres hexadecimales
        datosPayload[7]=(uint8_t)charBateria[0];// almacena los cuatro primeros
            valores de la cadena en el vector NB
        datosPayload[8]=(uint8_t)charBateria[1];
        datosPayload[9]=(uint8_t)charBateria[2];
        datosPayload[10]=(uint8_t)charBateria[3];
        // Obtener nivel de Potencia
        char Prx[3]="000"; // Ajusta el tamaño según tus necesidades
        // Utiliza sprintf para convertir el entero a cadena
        sprintf(Prx,"%03d",PotenciaTransmision);
        datosPayload[11]=(uint8_t)Prx[0];// almacena los tres primeros valores de
             la cadena en el vector NB
        datosPayload[12]=(uint8_t)Prx[1];
        datosPayload[13]=(uint8_t)Prx[2];
```

Con la información obtenida, se inicia el bucle con el número de transmisiones que se van a realizar, en cada iteración se compara si la trama es larga o corta y guarda las características de cada una, se define el valor de la secuencia y todo esto se almacena en el campo del payload de la trama, una vez cargada la información se transmite la trama, cambian los contadores de número de transmisiones y el número de secuencia y termina la iteración para volver a cargar la información. Este bucle se repetirá hasta que el valor del número de transmisiones llegue a 0, el algoritmo compara el valor de número de bloques a transmitir y repite el bucle de transmisión de tramas las veces que se definieron en el número de bloques, cada vez que se termine de transmitir cada bloque este valor se reduce, el proceso se repite hasta que el valor de número de bloques llegue a 0 y se transmite la trama de finalización de transmisión.

Código 2.7: Armado y envío en bucle de las tramas hacia el nodo receptor

```
if (contTX > 0)//Cantidad de veces que se transmitirán las tramas
        {//Trasmisión del mensaje y su tamaño
            char NumSec[6]; //Vector para enviar el número de secuencia
```

```
            if (TipoTrama==1) //Se transmitirá la trama corta

            {
                LED_Toggle(LED0);//Encender el led0 del nodo
                longitudTrama=23;//longitud de la trama en bytes
                datosPayload[0]='C';//Bandera de trama corta enviada en el
                    payload
                sprintf(NumSec,"0x%04X",numeroSecuencia);//transformación del
                    formato para enviar
                // almacena los valores del número de secuencia
                datosPayload[1]=(uint8_t)NumSec[0];
                datosPayload[2]=(uint8_t)NumSec[1];
                datosPayload[3]=(uint8_t)NumSec[2];
                datosPayload[4]=(uint8_t)NumSec[3];
                datosPayload[5]=(uint8_t)NumSec[4];
                datosPayload[6]=(uint8_t)NumSec[5];
            }
            else if (TipoTrama==2) //Se transmitirá la trama larga
            {
                LED_Toggle(LED0);
                longitudTrama=100;//longitud de la trama en bytes

                datosPayload[0]='L';//Bandera de trama larga enviada en el
                    payload
                sprintf(NumSec,"0x%04X",numeroSecuencia);//transformación del
                    formato para enviar
                // almacena los valores del número de secuencia
                datosPayload[1]=(uint8_t)NumSec[0];
                datosPayload[2]=(uint8_t)NumSec[1];
                datosPayload[3]=(uint8_t)NumSec[2];
                datosPayload[4]=(uint8_t)NumSec[3];
                datosPayload[5]=(uint8_t)NumSec[4];
                datosPayload[6]=(uint8_t)NumSec[5];
            }
            transmit_sample_frame(datosPayload, longitudTrama);//Función para
                transmitir la trama
            delay_ms(TiempoEntreTramas);//Tiempo de espera entre tramas enviadas
            numeroSecuencia++; //Aumenta el número de secuencia para la siguiente
                 trama
            contTX--; //Reduce el número de tramas faltantes
        }
```

```
40
        else if(contTX <= 0) //Indica el fin de la transmisión de un bloque
        {//Reinicio de variables
            uint8_t msgBL[11]="Bloque_0___";
            contTX=0;
45          //LED_On(LED0);
            if (contBL > 1) //Si hay más bloques de transmisión se reinicia el
                envío de las tramas
            {
                //LED_On(LED1);
                iniciarTransmision=1;
50              contTX=NumTransmiciones;

                indicadorBL++;
                msgBL[7]=indicadorBL+'0';
                transmit_sample_frame(msgBL,8);
55              delay_ms(TiempoEntreBloques);//Tiempo de espera entre tramas
                    enviadas
                contBL--;
            }
            else if (contBL<= 1) //Si el contador de bloques es menor que 1 se
                envía la última trama y se detiene la transmisión
            {
60              indicadorBL++;
                msgBL[7]=indicadorBL+'0';
                transmit_sample_frame(msgBL,11);
                iniciarTransmision=0;
                delay_ms(TiempoEntreTramas);//Tiempo de espera entre tramas
                    enviadas
65              indicadorBL=0;
            }
        }
    }
```

Una vez finalizada la función de transmisión del nodo, éste, vuelve al modo recepción a la espera de otra orden del nodo iniciador.

El código completo del nodo transmisor se presenta en el apartado de los anexos.

Nodo receptor/iniciador

Para el código, luego de definir las librerías antes mostradas, se definen las variables que

utilizaremos para los procesos dentro de las funciones:

Código 2.8: Definición de variables del nodo receptor

```
//variables

uint8_t vectorAlmacenamiento[20000]; //Vector que almacena todas las tramas recibidas

uint8_t ed_val; uint8_t potencia_trama; //Variables para obtener la potencia de la trama recibida

int Npayload=0; int desplas=0; //Variables de control para el almacenamiento de tramas

int  contAlm=0; int limAlm=0; //Contadores de Recepción
```

El nodo entra en el modo iniciador, en el cual, recibe los datos que se envían desde la PC por puerto serial, estos datos son los parámetros que tendrá la transmisión de tramas explicadas anteriormente, cuando se reciben datos por puerto serial el nodo tiene que enviar un mensaje de confirmación, al confirmar la recepción de los datos seriales, se van almacenando en el vector ”payload” para ser transmitidos posteriormente. Después de recibir estos datos, se envía desde la PC la orden para que estos parámetros sean enviados hacia el nodo transmisor, en este caso la orden será el carácter ”\t", el nodo envía el mensaje de confirmación hacia la PC y luego transmite la trama de los parámetros de transmisión.

Código 2.9: Codificación del modo iniciador del nodo para recibir los parámetros desde la PC y enviarlos al nodo transmisor

```
uint8_t mensajedeConfirmacion[6]="Listo";//Mensaje de confirmación de conexión
    mensajedeConfirmacion[5]=0xFF;

    int8_t chat_input = sio2host_getchar_nowait();//obtención del mensaje serial
    static uint8_t payload_length,payload[aMaxPHYPacketSize-FRAME_OVERHEAD]; //recibe datos desde la interfaz serial

    if (chat_input != -1) //Cuando recibe un valor diferente a -1 comienza a guardar los datos recibidos del PC en el payload
    {
            payload[payload_length++] = chat_input; //Guarda los datos recibidos del PC en el vector payload

```

```
    }
    if((chat_input == '\t')||(payload_length == (aMaxPHYPacketSize-FRAME_OVERHEAD
        ))){//El tipo de dato enviado de forma serial

        LED_Toggle(LED2);      // activa las siguientes funciones:

        sio2host_tx(mensajedeConfirmacion,6);//transmisión del mensaje de
            confirmación de conexión

        transmit_sample_frame(payload,payload_length-1);//transmite al nodo
            externo lo recibido por la conexión serial.

        payload_length = 0;//reseteo de variable

    }
```

Después de que se realice la transmisión de todas las tramas y el nodo haya almacenado la información en *vectorAlmacenamiento*, por puerto serial se envía la orden desde la PC para que se exporten estos datos hacia la computadora, para realizar esta orden se envía el caracter "\r", el cual inicia la función que divide el *vectorAlmacenamiento* el cual tiene todas las tramas 802.15.4 guardadas en vectores de 256 bytes los cuales se envían de manera serial hacia la PC.

Código 2.10: Codificación para el envío de tramas recibidas hacia la PC por puerto serial

```
if((chat_input == '\r')||(payload_length == (aMaxPHYPacketSize-FRAME_OVERHEAD)))
    //El tipo de dato enviado de forma serial
    {
        LED_Toggle(LED1);//encender el led
        desplas=0; //Inicializa la variable de lectura del vector de
            almacenamiento
//Guardado y envío del almacenamiento de forma serial al pc cada 256 bytes,
        for (int i = 0; i < 42; i++)
        {
            uint8_t datosPC[255]; //Vector para transmitir los datos a la PC por
                puerto serial

            for (int j = 0; j <255 ; j++)
            {
                datosPC[j]=vectorAlmacenamiento[desplas+j]; //Almacena los datos
                    recibidos para transmitir a la PC
```

```
            }
            desplas=desplas+255;
            sio2host_tx(datosPC,255);//Envio de datos al pc
        }
    }
```

Por último, el nodo para poder almacenar las tramas enviadas por el nodo transmisor entra en modo recepción. El nodo al empezar a recibir tramas lo que hace es comparar la dirección de origen de las tramas en este caso del nodo transmisor es 0x0001 y compara el primer caracter de la trama para definir si es ”C"trama corta o ”L"trama larga. Teniendo estos datos confirmados se extrae el valor de potencia de recepción del nodo receptor y se almacena en el payload de la trama recibida. Se obtiene el valor de la longitud del payload de la trama 802.15.4 y se compara si cumple con los datos establecidos de 23 o 100 dependiendo el tipo de trama, si cumple la condición se guarda cada byte de la trama recibida en *vectorAlmacenamiento* y se agrega un contador para que cada trama guardada, la siguiente trama toma las posiciones posteriores del vector para que no se sobreescriba la información.

Código 2.11: Codificación para guardar las tramas recibidas por el nodo transmisor

```
void usr_frame_received_cb(frame_info_t *frame)
{
    memset(&trama_recibida,0,sizeof(trama_recibida));// Elimina información
        previa de la estructura de recepción
    memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));// copia la
        información de la memoria fuente(buffer) al detino

    if (trama_recibida.dir_origen==0x0001 && (trama_recibida.cargaUtil_802_15_4
        [0]=='C'||trama_recibida.cargaUtil_802_15_4[0]=='L')) // Si se cumple
        esta condición la trama es del nodo transmisor
    {
        LED_Toggle(LED0);
        // Extracción del nivel de energía
        uint8_t *payload_ptr=frame->mpdu; //Trama recibida almacena en el vector
        uint8_t mpdu_len =payload_ptr[0]+2; //Busca la ubicación del valor de
            potencia de la trama recibida en los datos recibidos
        uint8_t potencia_trama=payload_ptr[mpdu_len];//Guarda el valor de la
            potencia de la trama
        trama_recibida.cargaUtil_802_15_4[14]=potencia_trama; //Agrega el valor
            de la potencia de la trama en el payload
```

```
        int longTrRx=((int)trama_recibida.longitud)-11; //Obtiene la longitud del
             payload
        if (longTrRx==23||longTrRx==100) //Si la longitud del payload es 23 (
            trama corta) o 100 (trama larga) puede guardar los datos
        {
            for (int i = 0; i < longTrRx; i++) //Recorre cada byte del payload
            {
                vectorAlmacenamiento[contAlm+i]=trama_recibida.cargaUtil_802_15_4
                    [i]; //Almacena los datos del payload en el vector de
                    almacenamiento
            }
                    contAlm=contAlm+longTrRx; //Variable de control para
                        almacenar el payload de las tramas recibidas en el vector
                         de almacenamiento
        }
    }
}
```

Una vez realizado todo este proceso se resetean las variables y el nodo vuelve al estado de iniciador.

El código completo se encuentra en el apartado de los anexos.

Codificación de la aplicación para la PC

Todo lo que es la programación en los nodos esta completo, pero también tenemos que desarrollar un programa para la computadora, específicamente para definir los parámetros de la transmisión y enviarlos por puerto serial hacia el nodo iniciador y para descargar las tramas del nodo receptor de igual manera por el puerto serial, para de esta manera obtener toda la información y calcular la tasa de tramas recibidas.

Para este programa se utiliza el programa PyCharm en el cual se debe utilizar el lenguaje de programación Python para de esta manera genera una interfaz gráfica que sea amigable con el usuario.

Este código se divide en secciones para poder realizar las distintas operaciones que realizará el programa, primero, se despliega la ventana de la aplicación la cual tiene las divisiones mencionadas, en la primera sección se establece la conexión con el puerto serial del PC, esto podemos verlo en el ”administrador de dispositivos” de Windows, una vez elegido el puerto selecciona el botón que da inicio a la comunicación serial, el siguiente paso es de-

finir los parámetros que recibirá el nodo iniciador para que sea enviado al nodo transmisor, estos parámetros se guardan en variables las cuales son enviadas de manera serial y se envía la orden al nodo para emitir las tramas al nodo transmisor.

Enseguida se despliega una barra de avance el cual indica el estatus de la transmisión de las tramas, una vez esta barra llega al 100 % significa que la transmisión se finalizó.

En la siguiente sección se envía la orden al nodo receptor de importar las tramas recibidas hacia la PC por el puerto serial, todas estas tramas se almacenan en un vector, una vez almacenadas las tramas, se procesa este vector de manera que se divide todas las tramas y se guardan en una matriz la cual cada fila corresponde a cada trama recibida y cada columna corresponde al parámetro que se enviaron en cada trama (tipo de trama, No de secuencia, nivel de batería, PTx, PRx, carga útil), almacenados los datos en la matriz en la ventana de la aplicación se escoge la orden para guardar esta matriz en un archivo Excel, dentro del archivo también se muestra las tramas enviadas, las tramas recibidas y la tasa de tramas perdidas.

A continuación, se presenta la sección del código donde se envían los parámetros hacia el nodo iniciador por el puerto serial:

Código 2.12: Envío de parámetros por puerto serial hacia el nodo iniciador

```
1         def EnviarOrdenes(self):
-
-         self.conectar.open()
-         self.conectar.flush()
5         sparador = b'\t'
-         identificador = b'\n'
-         msj = 'ini'  # Inicio de Ordenes
-         mensajeCod = msj.encode()
-         self.conectar.write(mensajeCod + sparador)
10        conf = self.conectar.read_until(b'\xFF')
-         print('Confirmacion: ' + conf[:5].decode())
-         # Orden 'a' Número de Transmisiones
-         msj = self.primeraventana.spinBoxNumeroTransmisiones.text() + 'a'  #
              Orden 'a' Numero de Transmiciones
-         NumeroTX = int(self.primeraventana.spinBoxNumeroTransmisiones.text())
15        mensajeCod = msj.encode()
-         self.conectar.write(mensajeCod + sparador)
-         conf = self.conectar.read_until(b'\xFF')
-         print('Confirmacion: ' + conf[:5].decode())
```

```
        # Orden 'b' Tiempo entre tramas
        msj = self.primeraventana.spinBoxTiempoEntreTramas.text() + 'b'
        TiempoEntreTr = int(self.primeraventana.spinBoxTiempoEntreTramas.text())
        mensajeCod = msj.encode()
        self.conectar.write(mensajeCod + sparador)
        conf = self.conectar.read_until(b'\xFF')
        print('Confirmacion: ' + conf[:5].decode())
        # Orden 'c' Número de Bloques
        msj = self.primeraventana.spinBoxNumeroBloques.text() + 'c'
        NumeroBl = int(self.primeraventana.spinBoxNumeroBloques.text())
        mensajeCod = msj.encode()
        self.conectar.write(mensajeCod + sparador)
        conf = self.conectar.read_until(b'\xFF')
        print('Confirmacion: ' + conf[:5].decode())
        # Orden 'd' Tiempo entre Bloques
        msj = self.primeraventana.spinBoxTiempoEntreBloques.text() + 'd'
        TiempoEntreBl = int(self.primeraventana.spinBoxTiempoEntreBloques.text())
        mensajeCod = msj.encode()
        self.conectar.write(mensajeCod + sparador)
        conf = self.conectar.read_until(b'\xFF')
        print('Confirmacion: ' + conf[:5].decode())
        # Orden 'e' Potencia de transmisión
        msj = self.primeraventana.spinBoxPotenciaTransmision.text() + 'e'
        mensajeCod = msj.encode()
        self.conectar.write(mensajeCod + sparador)
        conf = self.conectar.read_until(b'\xFF')
        print('Confirmacion: ' + conf[:5].decode())
        # Orden 'f' Potencia de transmisión
        tamañoTrama = self.primeraventana.comboxTamanioTrama.itemText(
            self.primeraventana.comboxTamanioTrama.currentIndex())
        ordenTamañoTrama = '0'
        if tamañoTrama == 'Trama Corta':
            ordenTamañoTrama = '1'
        elif tamañoTrama == 'Trama Larga':
            ordenTamañoTrama = '2'
        msj = ordenTamañoTrama + 'f'
        mensajeCod = msj.encode()
        self.conectar.write(mensajeCod + sparador)
        conf = self.conectar.read_until(b'\xFF')
        print('Confirmacion: ' + conf[:5].decode())
        # Finalización de órdenes
        msj = 'fin'
```

```
        mensajeCod = msj.encode()
        self.conectar.write(mensajeCod + sparador)
        conf = self.conectar.read_until(b'\xFF')
        print('Confirmacion: ' + conf[:5].decode())
        self.conectar.close()  # Cerrar conexión
        print('Ordenes Completas ')
        ##########################################################################
        #Configuración de tiempos
        tiempoAdicionalBl = (TiempoEntreBl) * 0.55
        tiempoAdicionalTr = (TiempoEntreTr) * 0.55
        self.tiempoTotal=0.0
        if NumeroBl==1:
            self.tiempoTotal = (TiempoEntreTr*NumeroTX) / 1000
        elif NumeroBl>1:
            self.tiempoTotal = (((TiempoEntreTr+tiempoAdicionalTr) * (NumeroTX)*
                (NumeroBl)) / 1000) + ((TiempoEntreBl+ tiempoAdicionalBl) * (
                NumeroBl-1))
        print(f"tiempoTotal: {self.tiempoTotal}")
        self.iniciarTemporizadorRecepcion()
```

Una vez el nodo receptor recibe todas las tramas se envía la orden para enviar los datos por puerto serial hacia la PC y ésta las almacena.

Código 2.13: Obtener datos del nodo receptor hacia la PC

```
    def ObtenerDatos(self):
        self.conectar.open() # Abrir el puerto serial
        self.conectar.flush()
        sparador = b'\r'
        tramaHexString = ''
        codificado = bytes.fromhex(tramaHexString)  # tramaHexString.encode()
        self.conectar.write(codificado + sparador)  # Señal para obtener datos
        print(f"MSG enviado: {codificado + sparador}")
        self.contador=0
        self.iniciarTemporizadorDescarga()
```

Por último se muestra el código el cual arma el documento en Excel:

Código 2.14: Obtener datos del nodo receptor hacia la PC

```
    def GenerarArchivoEXEL(self):
        cadena_bytes = self.DatosCompletosGlobales
        # Inicializar la lista para almacenar los resultados
```

```
        resultados = []
        # Iterar sobre la cadena de bytes
        i = 0
        while i < len(cadena_bytes):
            # Buscar 'C0x' o 'L0x'
            if cadena_bytes[i:i + 3] == b'C0x':
                num_bytes_siguientes = 20
                resultado = [
                    cadena_bytes[i:i + 1].decode(),  # L
                    cadena_bytes[i + 1:i + 7].decode(),  # 0x0000
                    float(cadena_bytes[i + 7:i + 11].decode()),  # 2.55
                    int(cadena_bytes[i + 11:i + 14].decode()),  # 003
                    ord(cadena_bytes[i + 14:i + 15]) - 90,  # Restar (-90)
                    cadena_bytes[i + 15:i + 23].decode()  # '1111111111'
                        decodificado
                ]
                resultados.append(resultado)
                i += 3 + num_bytes_siguientes
            elif cadena_bytes[i:i + 3] == b'L0x':
                num_bytes_siguientes = 97
                resultado = [
                    cadena_bytes[i:i + 1].decode(),  # L
                    cadena_bytes[i + 1:i + 7].decode(),  # 0x0000
                    float(cadena_bytes[i + 7:i + 11].decode()),  # 2.55
                    int(cadena_bytes[i + 11:i + 14].decode()),  # 003
                    ord(cadena_bytes[i + 14:i + 15]) - 90,  # Restar (-90)
                    cadena_bytes[i + 15:i + 100].decode()  # '1111111111'
                        decodificado
                ]
                resultados.append(resultado)
                i += 3 + num_bytes_siguientes
            else:
                i += 1
        print('Datos encontrados:')
        for resultado in resultados:
            print(resultado)
        # Crear un DataFrame con los resultados
        df = pd.DataFrame(resultados,
                          columns=['Tipo', 'No. Secuencia', 'Nivel Bateria[V]', '
                              Ptx[dBm]', 'Prx[dBm]', 'Carga Util'])
        print(df)
        # Crear una ventana de Tkinter (no se mostrará físicamente)
```

```
        # Crear una ventana de diálogo para seleccionar la ubicación y el nombre del archivo Excel
        root = Tk()
        root.withdraw()  # Ocultar la ventana principal de la aplicación

        file_path = filedialog.asksaveasfilename(defaultextension=".xlsx", filetypes=[("Excel files", "*.xlsx")])

        # Verificar si se proporcionó un nombre de archivo antes de guardar
        if file_path:
            # Guardar el DataFrame en el archivo Excel seleccionado
            with pd.ExcelWriter(file_path, engine='openpyxl') as writer:
                df.to_excel(writer, index=False, sheet_name='Hoja1')

                # Ajustar automáticamente el tamaño de las celdas
                worksheet = writer.sheets['Hoja1']
                for column_cells in worksheet.columns:
                    max_length = 0
                    column = [cell for cell in column_cells]
                    for cell in column:
                        try:
                            if len(str(cell.value)) > max_length:
                                max_length = len(cell.value)
                        except:
                            pass
                    adjusted_width = (max_length + 2)
                    worksheet.column_dimensions[column[0].column_letter].width = adjusted_width

                # Centrar el contenido en todas las celdas
                for row in worksheet.iter_rows(min_row=2, max_row=worksheet.max_row, min_col=1,
                                               max_col=worksheet.max_column):
                    for cell in row:
                        cell.alignment = cell.alignment.copy(horizontal='center', vertical='center')
            print(f"DataFrame guardado exitosamente en {file_path}")
        else:
            print("Operación de guardado cancelada.")
```

El código completo de la aplicación para la PC se encuentra en los anexos.

En la Figura 2.13 se muestra la ventana desplegada de la aplicación desarrollada para la PC.

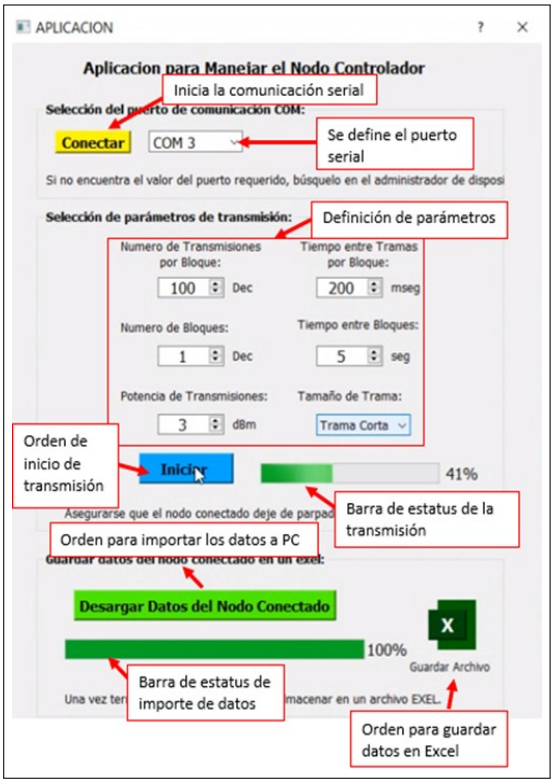

Figura 2.13: Secciones de la ventana desplegada al abrir la aplicación para la PC

El archivo se guarda en la ubicación y con el nombre especificado, al abrir el archivo se puede observar el listado de las tramas recibidas con su información.

Tipo	No. Secuencia	Nivel Bateria[V]	Ptx[dBm]	Prx[dBm]	Carga Util	Tramas enviadas	Tramas recibidas	% tramas perdidas
C	0x0000	2.69	3	-27	11111111	100	100	0
C	0x0001	2.69	3	-27	11111111			
C	0x0002	2.69	3	-27	11111111			
C	0x0003	2.69	3	-27	11111111			
C	0x0004	2.69	3	-27	11111111			
C	0x0005	2.69	3	-27	11111111			
C	0x0006	2.69	3	-27	11111111			
C	0x0007	2.69	3	-28	11111111			
C	0x0008	2.69	3	-27	11111111			
C	0x0009	2.69	3	-27	11111111			
C	0x000A	2.69	3	-27	11111111			
C	0x000B	2.69	3	-27	11111111			
C	0x000C	2.69	3	-27	11111111			
C	0x000D	2.69	3	-27	11111111			
C	0x000E	2.69	3	-27	11111111			
C	0x000F	2.69	3	-27	11111111			
C	0x0010	2.69	3	-27	11111111			
C	0x0011	2.69	3	-27	11111111			
C	0x0012	2.69	3	-27	11111111			
C	0x0013	2.69	3	-27	11111111			
C	0x0014	2.69	3	-27	11111111			
C	0x0015	2.69	3	-27	11111111			
C	0x0016	2.69	3	-27	11111111			
C	0x0017	2.69	3	-27	11111111			
C	0x0018	2.69	3	-27	11111111			
C	0x0019	2.69	3	-27	11111111			

Figura 2.14: Ejemplo de archivo generado a partir de las tramas recibidas

De esta manera se puede ver toda la información requerida de las tramas y lo más importante podemos calcular la tasa de tramas perdidas, en la siguiente sección se muestran las pruebas realizadas a los nodos ATZB-256RFR2-XPRO.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 PRUEBAS

En este capítulo se muestran los resultados de las diferentes pruebas que se hicieron a los nodos 802.15.4, estas pruebas se realizan enviando distintos tipos de tramas, variando la distancia entre nodos y cambiando el tiempo de delay, la potencia de transmisión será 3dBm por defecto.

Las pruebas se llevarán a cabo a distancias de 10cm, 5m y 10m entre los nodos con el fin de observar la pérdida de las tramas, en caso de ser necesario se agregarán obstáculos a la línea de vista de los nodos, en todas las pruebas se transmitirán 100 tramas consecutivas variando el tipo de la trama para observar los resultados y se tomará el tiempo entre cada trama de 200ms para todas las pruebas.

3.1.1 PRUEBA 1

Para esta prueba se colocaron a los nodos a una distancia de 10cm entre sí con un delay entre tramas de 200ms y enviando tramas cortas enviando en total 100 tramas consecutivas, lo que se espera de esta prueba es que no se tenga pérdida alguna ya que estamos dentro del alcance de estos nodos, se muestra la pantalla de la aplicación con los parámetros a continuación.

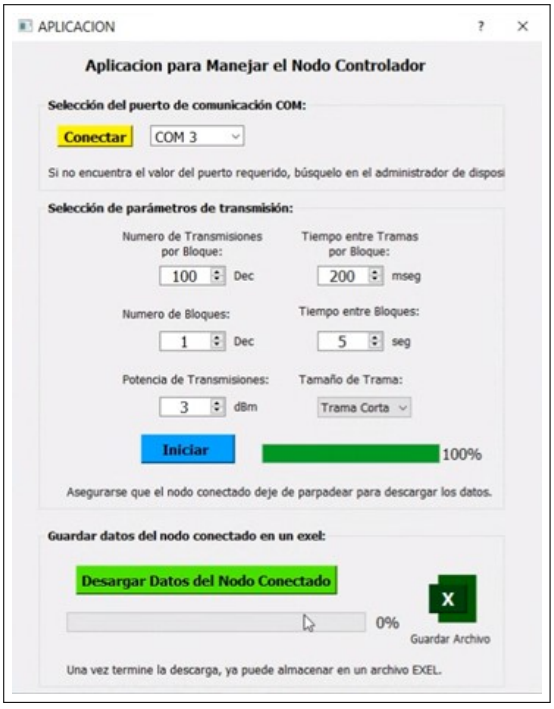

Figura 3.1: Parámetros de transmisión de 200ms entre tramas y 10cm entre nodos y tramas cortas

Resultados:

Figura 3.2: Archivo generado de la prueba 1

	A	B	C	D	E	F	G	H	I	J
1	Tipo	No. Secuencia	Nivel Bateria[V]	Ptx[dBm]	Prx[dBm]	Carga Util		Tramas enviadas	Tramas recibidas	% tramas perdidas
2	C	0x0000	2.69	3	-27	11111111		100	100	0
3	C	0x0001	2.69	3	-27	11111111				
4	C	0x0002	2.69	3	-27	11111111				
5	C	0x0003	2.69	3	-27	11111111				
6	C	0x0004	2.69	3	-27	11111111				
7	C	0x0005	2.69	3	-27	11111111				
8	C	0x0006	2.69	3	-27	11111111				
9	C	0x0007	2.69	3	-28	11111111				
10	C	0x0008	2.69	3	-27	11111111				
11	C	0x0009	2.69	3	-27	11111111				
12	C	0x000A	2.69	3	-27	11111111				
13	C	0x000B	2.69	3	-27	11111111				
14	C	0x000C	2.69	3	-27	11111111				
15	C	0x000D	2.69	3	-27	11111111				
16	C	0x000E	2.69	3	-27	11111111				
17	C	0x000F	2.69	3	-27	11111111				
18	C	0x0010	2.69	3	-27	11111111				
19	C	0x0011	2.69	3	-27	11111111				
20	C	0x0012	2.69	3	-27	11111111				
21	C	0x0013	2.69	3	-27	11111111				
22	C	0x0014	2.69	3	-27	11111111				

Figura 3.3: Resultado del archivo generado con las tramas de la prueba 1

Como se puede ver en los resultados tenemos la recepción de las 100 tramas enviadas, lo que nos da una tasa de pérdida del 0 %, también podemos darnos cuenta de cuanta es la potencia de recepción que estamos teniendo, en este caso estamos recibiendo un campo de -27dBm en promedio, lo cual es óptimo para una comunicación inalámbrica.

3.1.2 PRUEBA 2

Para esta prueba, a fin de comparar los resultados cuando se envían tramas largas utilizaremos los mismos parámetros que la transmisión anterior variando el parámetro del tipo de trama.

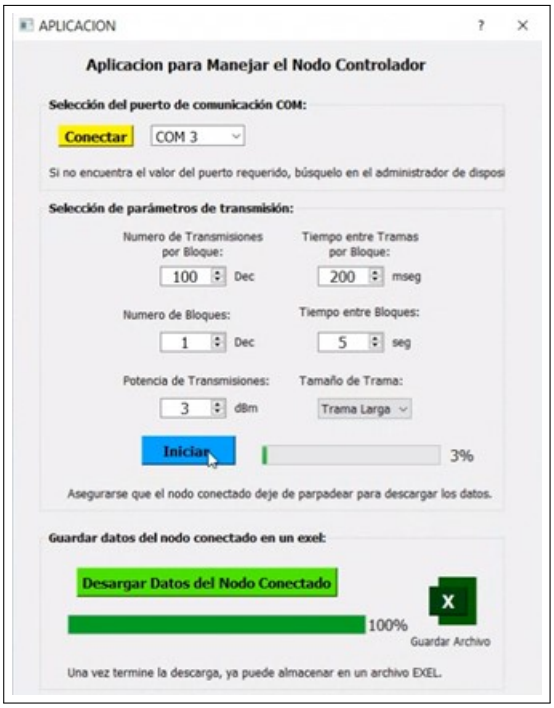

Figura 3.4: Parámetros de transmisión de 200ms entre tramas y 10cm entre nodos y tramas largas

Resultados:

Figura 3.5: Archivo generado de la prueba 2

Tipo	No. Secuencia	Nivel Bateria[V]	Ptx[dBm]	Prx[dBm]	Carga Util	Tramas enviadas	Tramas recibidas	% tramas perdidas
L	0x0000	2.69	3	-31	1111111111111111111111111111111111111111111111111111111111111111	100	100	0
L	0x0001	2.69	3	-31	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0002	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0003	2.69	3	-31	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0004	2.69	3	-31	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0005	2.69	3	-31	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0006	2.69	3	-31	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0007	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0008	2.69	3	-31	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0009	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x000A	2.69	3	-31	1111111111111111111111111111111111111111111111111111111111111111			
L	0x000B	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x000C	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x000D	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x000E	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x000F	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0010	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0011	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0012	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0013	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			
L	0x0014	2.69	3	-30	1111111111111111111111111111111111111111111111111111111111111111			

Figura 3.6: Resultado del archivo generado con las tramas de la prueba 2

En este caso también tenemos lo esperado, tenemos el 100 % de la transmisión exitosa.

3.1.3 PRUEBA 3

En este caso vamos a variar la distancia entre nodos, los colocamos a una distancia de 5m entre sí, sin obstáculos, con 200ms de delay entre tramas y 100 tramas cortas.

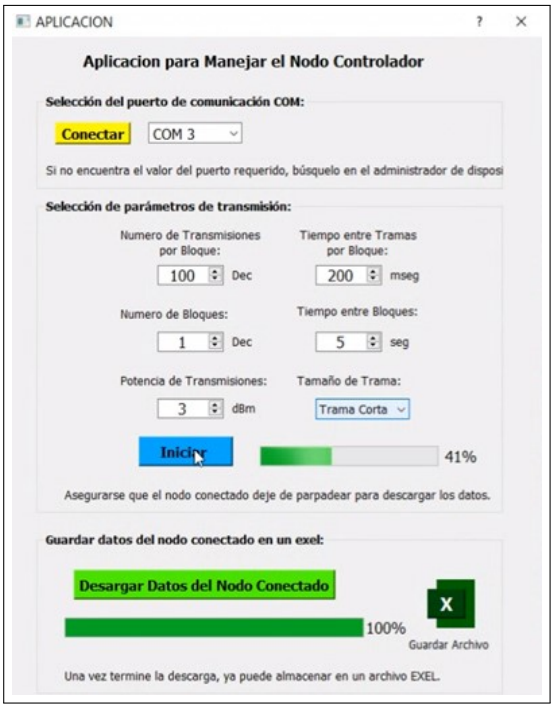

Figura 3.7: Parámetros de transmisión de 200ms entre tramas y 5 entre nodos y tramas cortas

Resultados:

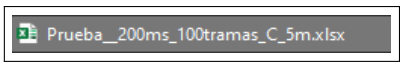

Figura 3.8: Archivo generado de la prueba 3

	A	B	C	D	E	F	G	H	I	J
1	Tipo	No. Secuencia	Nivel Bateria[V]	Ptx[dBm]	Prx[dBm]	Carga Util		Tramas enviadas	Tramas recibidas	% tramas perdidas
2	C	0x0000	2.69	3	-64	11111111		100	100	0
3	C	0x0001	2.69	3	-66	11111111				
4	C	0x0002	2.69	3	-66	11111111				
5	C	0x0003	2.69	3	-68	11111111				
6	C	0x0004	2.69	3	-67	11111111				
7	C	0x0005	2.69	3	-65	11111111				
8	C	0x0006	2.69	3	-66	11111111				
9	C	0x0007	2.69	3	-67	11111111				
10	C	0x0008	2.69	3	-66	11111111				
11	C	0x0009	2.69	3	-69	11111111				
12	C	0x000A	2.69	3	-67	11111111				
13	C	0x000B	2.69	3	-69	11111111				
14	C	0x000C	2.69	3	-66	11111111				
15	C	0x000D	2.69	3	-69	11111111				
16	C	0x000E	2.69	3	-66	11111111				
17	C	0x000F	2.69	3	-69	11111111				
18	C	0x0010	2.69	3	-66	11111111				
19	C	0x0011	2.69	3	-66	11111111				
20	C	0x0012	2.69	3	-66	11111111				
21	C	0x0013	2.69	3	-66	11111111				
22	C	0x0014	2.69	3	-66	11111111				

Figura 3.9: Resultado del archivo generado con las tramas de la prueba 3

En este caso también tenemos una transmisión del 100 % ya que se reciben todas las tramas transmitidas, pero ahora, cuando variamos la distancia podemos ver que la potencia de recepción se reduce sustancialmente a comparación de la transmisión a 10cm, tenemos un campo de -66dBm.

3.1.4 PRUEBA 4

Para esta prueba utilizaremos los mismos parámetros que la transmisión anterior pero esta vez enviaremos tramas largas para observar si se tiene cambios en la recepción.

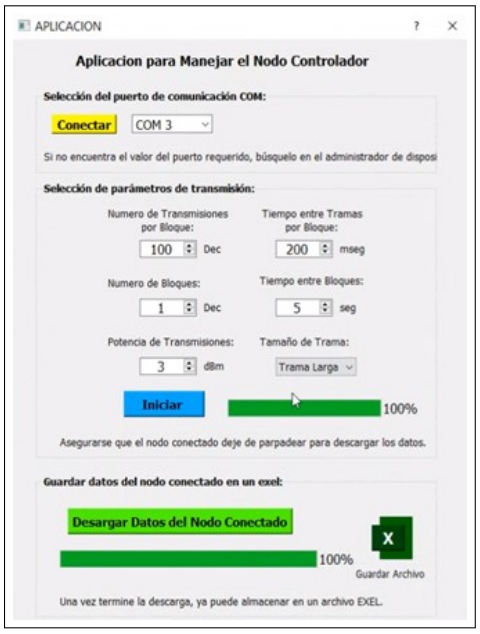

Figura 3.10: Parámetros de transmisión de 200ms entre tramas y 5m entre nodos y tramas largas

Resultados:

Figura 3.11: Archivo generado de la prueba 4

Tipo	No. Secuencia	Nivel Bateria[V]	Ptx[dBm]	Prx[dBm]	Carga Util	Tramas enviadas	Tramas recibidas	% tramas perdidas
L	0x0064	2.69	3	-68	1111111111111111111111111111111111111111111111111111111111111111111111111111	100	100	0
L	0x0065	2.69	3	-67	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0066	2.69	3	-66	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0067	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0068	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0069	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x006A	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x006B	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x006C	2.69	3	-64	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x006D	2.69	3	-64	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x006E	2.69	3	-64	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x006F	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0070	2.69	3	-64	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0071	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0072	2.69	3	-64	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0073	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0074	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0075	2.69	3	-65	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0076	2.69	3	-64	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0077	2.69	3	-64	1111111111111111111111111111111111111111111111111111111111111111111111111111			
L	0x0078	2.69	3	-64	1111111111111111111111111111111111111111111111111111111111111111111111111111			

Figura 3.12: Resultado del archivo generado con las tramas de la prueba 4

Como podemos observar tenemos campos de recepción parecidos con la transmisión de tramas cortas, tenemos una potencia de -65dBm en promedio, pero aun así tenemos 0 % de pérdidas.

3.1.5 PRUEBA 5

En esta prueba se agrega una pared entre los nodos a una distancia de 5m para verificar el comportamiento de la transmisión, igualmente se transmite 100 tramas cortas con 200ms de delay entre tramas, los parámetros de la transmisión son los mismos que la Figura 3.7.

Resultados:

Figura 3.13: Archivo generado de la prueba 5

	A	B	C	D	E	F	G	H	I	J
1	Tipo	No. Secuencia	Nivel Bateria[V]	Ptx[dBm]	Prx[dBm]	Carga Util		Tramas enviadas	Tramas recibidas	% tramas perdidas
2	C	0x0000	2.69	3	-72	11111111		100	100	0
3	C	0x0001	2.69	3	-72	11111111				
4	C	0x0002	2.69	3	-72	11111111				
5	C	0x0003	2.69	3	-71	11111111				
6	C	0x0004	2.69	3	-69	11111111				
7	C	0x0005	2.62	3	-69	11111111				
8	C	0x0006	2.62	3	-69	11111111				
9	C	0x0007	2.62	3	-69	11111111				
10	C	0x0008	2.69	3	-69	11111111				
11	C	0x0009	2.62	3	-69	11111111				
12	C	0x000A	2.62	3	-69	11111111				
13	C	0x000B	2.62	3	-69	11111111				
14	C	0x000C	2.62	3	-69	11111111				
15	C	0x000D	2.69	3	-69	11111111				
16	C	0x000E	2.62	3	-69	11111111				
17	C	0x000F	2.62	3	-69	11111111				
18	C	0x0010	2.62	3	-69	11111111				
19	C	0x0011	2.69	3	-69	11111111				
20	C	0x0012	2.69	3	-69	11111111				
21	C	0x0013	2.69	3	-69	11111111				
22	C	0x0014	2.69	3	-69	11111111				

Figura 3.14: Resultado del archivo generado con las tramas de la prueba 5

Como era de esperarse al tener un obstáculo entre los nodos la transmisión se vuelve más difícil ya que parte de la potencia que se transmite no puede atravesar la pared, en este caso tenemos una reducción de campo de 4dBm, en promedio la potencia de recepción es de -70.08dBm, pero aun así tenemos el 100 % de efectividad en la transmisión.

3.1.6 PRUEBA 6

En esta prueba agregaremos más dificultad para la comunicación inalámbrica, la distancia entre los nodos será de 10m con 2 paredes entre ellos, se enviarán 100 tramas cortas con delay de 200ms entre tramas, con el fin de obtener pérdidas en la transmisión.

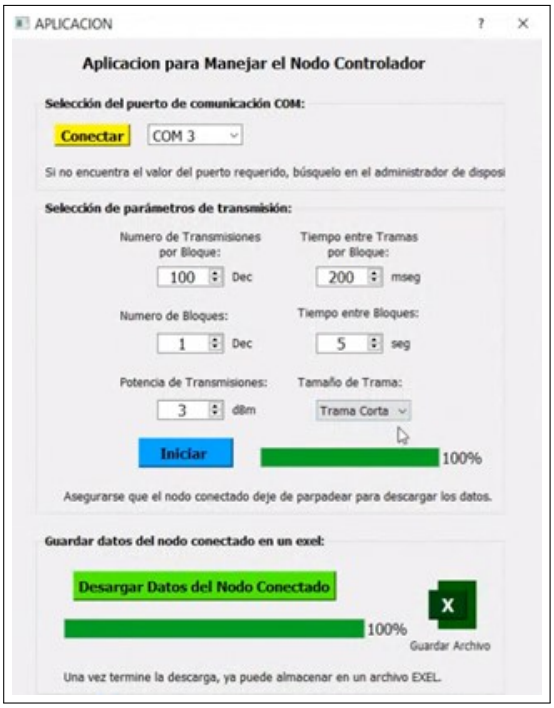

Figura 3.15: Parámetros de transmisión de 200ms entre tramas y 10m entre nodos y tramas cortas

Resultados:

Figura 3.16: Archivo generado de la prueba 6

	A	B	C	D	E	F	G	H	I	J
1	Tipo	No. Secuencia	Nivel Bateria[V]	Ptx[dBm]	Prx[dBm]	Carga Util		Tramas enviadas	Tramas recibidas	% tramas perdidas
2	C	0x0002	2.62	3	-88	11111111		100	34	66
3	C	0x0003	2.62	3	-87	11111111				
4	C	0x0004	2.69	3	-88	11111111				
5	C	0x0006	2.69	3	-89	11111111				
6	C	0x0009	2.69	3	-89	11111111				
7	C	0x000A	2.69	3	-89	11111111				
8	C	0x000B	2.62	3	-89	11111111				
9	C	0x000C	2.69	3	-90	11111111				
10	C	0x000D	2.69	3	-89	11111111				
11	C	0x000E	2.62	3	-89	11111111				
12	C	0x0011	2.69	3	-89	11111111				
13	C	0x0012	2.62	3	-89	11111111				
14	C	0x0013	2.62	3	-89	11111111				
15	C	0x0015	2.62	3	-89	11111111				
16	C	0x0016	2.69	3	-89	11111111				
17	C	0x001A	2.62	3	-89	11111111				
18	C	0x001D	2.69	3	-89	11111111				
19	C	0x001E	2.69	3	-89	11111111				
20	C	0x0024	2.69	3	-88	11111111				
21	C	0x0025	2.62	3	-90	11111111				
22	C	0x002F	2.69	3	-85	11111111				

Figura 3.17: Resultado del archivo generado con las tramas de la prueba 6

En esta prueba ya podemos observar que tenemos pérdidas del 66 % de tramas, esto significa que los obstáculos y la distancia hicieron que la potencia que se recibe de algunas tramas sea menor que el umbral de los nodos, en este caso el umbral de los nodos es de -90dBm, quiere decir que las tramas que llegan con menos potencia que ese valor umbral no pueden ser detectadas.

3.1.7 PRUEBA 7

Con el fin de comparar el efecto que tiene el delay entre tramas, se realizará la misma prueba 6 pero aumentando el delay a 500ms, idealmente como tenemos una menor frecuencia de datos las pérdidas deben ser menores.

Resultados:

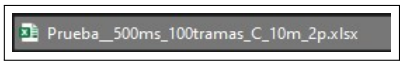

Figura 3.18: Archivo generado de la prueba 7

	A	B	C	D	E	F	G	H	I	J
1	Tipo	No. Secuencia	Nivel Bateria[V]	Ptx[dBm]	Prx[dBm]	Carga Util		Tramas enviadas	Tramas recibidas	% tramas perdidas
2	C	0x0066	2.62	3	-88	11111111		100	63	37
3	C	0x0067	2.62	3	-88	11111111				
4	C	0x006B	2.62	3	-87	11111111				
5	C	0x006C	2.62	3	-88	11111111				
6	C	0x006D	2.62	3	-87	11111111				
7	C	0x0071	2.62	3	-81	11111111				
8	C	0x0072	2.62	3	-83	11111111				
9	C	0x0073	2.62	3	-85	11111111				
10	C	0x0074	2.62	3	-87	11111111				
11	C	0x0075	2.62	3	-85	11111111				
12	C	0x0076	2.62	3	-84	11111111				
13	C	0x0077	2.62	3	-84	11111111				
14	C	0x0079	2.62	3	-85	11111111				
15	C	0x007A	2.62	3	-85	11111111				
16	C	0x007B	2.62	3	-82	11111111				
17	C	0x007C	2.62	3	-82	11111111				
18	C	0x007E	2.62	3	-84	11111111				
19	C	0x007F	2.62	3	-84	11111111				
20	C	0x0080	2.62	3	-82	11111111				
21	C	0x0083	2.62	3	-82	11111111				
22	C	0x0084	2.62	3	-83	11111111				

Figura 3.19: Resultado del archivo generado con las tramas de la prueba 7

Como podemos ver en estos resultados tenemos una pérdida de tramas del 37 %, eso quiere decir que si se reduce la frecuencia de transmisión de las tramas el sistema de comunicación se vuelve más robusto, a fines prácticos si la distancia de los nodos es muy grande o hay obstáculos entre ellos una manera de volver más fuerte a la comunicación es reducir la frecuencia de transmisión.

A continuación, se presenta la tabla con los resultados obtenidos de las pruebas:

Tabla 3.1: Resultados de las pruebas de transmisión de tramas 802.15.4

No de tramas	Delay entre tramas (ms)	Tipo de trama	Distancia entre nodos	Obstáculos	Potencia recepción promedio	Tasa de tramas perdidas %
100	200	Corta	10cm	No	-27.03dBm	0
100	200	Larga	10cm	No	-30.4dBm	0
100	200	Corta	5m	No	66.8dBm	0
100	200	Larga	5m	No	-65.55dBm	0
100	200	Corta	5m	1 pared	-70.08dBm	0
100	200	Corta	10m	2 paredes	-88.23dBm	66
100	500	Corta	10m	2 paredes	-83.12dBm	37

En la Tabla 3.1 podemos darnos cuenta de mejor manera del comportamiento que tienen los nodos en las transmisiones, a medida que se separan entre sí, la potencia que se recibe de la comunicación es menor pero las tramas que logren estar dentro del umbral de recepción de los nodos son recibidos sin mayor problema, de los archivos generados también podemos darnos cuenta que el umbral de estos transceptores rodea los -90dBm ya que ninguna trama es recibida con la potencia menor a ese valor.

3.2 CONCLUSIONES

- En conclusión, el desarrollo de una herramienta específica para calcular la tasa de tramas perdidas en transmisiones mediante el protocolo 802.15.4 representa una respuesta valiosa a la carencia evidente de herramientas especializadas en el mercado actual. La ausencia de soluciones estandarizadas y accesibles para medir de manera precisa y consistente este parámetro crítico ha generado obstáculos significativos para la evaluación efectiva de las redes inalámbricas que siguen estas directrices. La iniciativa de construir una herramienta dedicada no solo busca abordar esta limitación, sino que también pretende ofrecer a desarrolladores y administradores de redes una herramienta adaptada y confiable que facilite la identificación de problemas de transmisión y la mejora continua de la calidad de las comunicaciones inalámbricas. Este proyecto no solo suple una necesidad inmediata en el ámbito de las mediciones de redes, sino que también sienta las bases para futuras investigaciones y desarrollos, contribuyendo así a la evolución y optimización de las tecnologías de comunicación inalámbrica. En última instancia, esta herramienta representa un paso adelante significativo en el fortalecimiento de la infraestructura de medición para redes basadas en el protocolo 802.15.4.

- En conclusión, el proyecto desarrollado para medir la tasa de tramas perdidas en el protocolo 802.15.4 emerge como un recurso funcional y esencial. Los resultados obtenidos demuestran su eficacia en proporcionar mediciones precisas y detalladas, destacando su utilidad para identificar y abordar problemas de transmisión.

- La herramienta desarrollada proporcionó evidencia concluyente de que los nodos 802.15.4 no son adecuados para transmisiones a largas distancias. La medición de la tasa de tramas perdidas demostró su eficiencia en entornos de corto alcance, reforzando la comprensión de su limitado alcance. Un ejemplo ilustrativo de su aplicación efectiva se encuentra en el ámbito del Internet de las Cosas (IoT), donde estos nodos son idóneos para conectar y controlar dispositivos en un hogar inteligente, ofreciendo una comunicación fiable y de bajo consumo de energía en distancias cortas, pero limitando su uso en aplicaciones que requieren cobertura extensa.

- Durante el dessarrollo del proyecto pudimos darnos cuenta de que se puede optimizar el modelo inicial, al únicamente utilizar solo 2 nodos en lugar de 3 vuelve más accesible la herramienta para un uso cotidiano en caso de que la persona que quiera usar la herramienta cuente con nodos limitados.

3.3 RECOMENDACIONES

- Se recomienda definir y documentar claramente los parámetros específicos que la herramienta medirá, como la tasa de tramas perdidas y los intervalos de tiempo. Considere la conformidad con estándares reconocidos para garantizar la coherencia y la comparabilidad de los resultados en distintos entornos y aplicaciones del protocolo 802.15.4.

- Se recomienda el uso de esta herramienta para tener un mejor escenario en caso de querer realizar un proyecto de IoT con nodos 802.15.4 ya que de esta manera se puede optimizar el performance de los nodos y tener mejores resultados ya teniendo claros los límites de los equipos.

- Según los resultados obtenidos en la subsección anterior se recomienda que si se realiza una red con nodos 802.15.4 de manera doméstica se recomienda colocar los nodos a una distancia de no más de 10m entre sí, en caso de tener obstáculos como paredes o columnas ya que la señal no llega de la manera más eficaz.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] G. Lu, B. Krishnamachari y C. Raghavendra, "Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks," en *IEEE International Conference on Performance, Computing, and Communications, 2004*, abr. de 2004, págs. 701-706. DOI: 10.1109/PCCC.2004.1395158. dirección: https://ieeexplore.ieee.org/abstract/document/1395158 (visitado 05-02-2024).

[2] A. J. G. Robayo, "ALGORITMO PARA LA TRANSMISIÓN CONFIABLE DE PAQUETES DE DATOS EN ESTRUCTURAS LINEALES A NIVEL DE LA CAPA DE ENLACE DE DATOS SIN NECESIDAD DE USAR ACUSES DE RECIBO," es,

[3] L. De Nardis y M.-G. Di Benedetto, "Overview of the IEEE 802.15.4/4a standards for low data rate Wireless Personal Data Networks," en *Navigation and Communication 2007 4th Workshop on Positioning*, mar. de 2007, págs. 285-289. DOI: 10.1109/WPNC.2007.353647. dirección: https://ieeexplore.ieee.org/abstract/document/4167853 (visitado 05-02-2024).

[4] I. M. N. Méndez y C. R. E. Acosta, "REDES DE SENSORES INALÁMBRICOS PARA IOT ALGORITMO PARA LA ELIMINACIÓN DE TRAMAS REPETIDAS EN AMBIENTES CON TOPOLOGÍA LINEAL QUE OPERAN CON EL PROTOCOLO IEEE 802.15.4.," es,

[5] F. D. C. Reyes, "IMPLEMENTACIÓN DEL ALGORITMO DE PROTOCOLO DE DIRECCIONAMIENTO PARA REDES DE SENSORES INALÁMBRICOS CON EL ESTANDAR IEEE 802.15.4.," es,

[6] A. Faridi, M. R. Palattella, A. Lozano et al., "Comprehensive Evaluation of the IEEE 802.15.4 MAC Layer Performance With Retransmissions," *IEEE Transactions on Vehicular Technology*, vol. 59, n.º 8, págs. 3917-3932, 2010. DOI: 10.1109/TVT.2010.2063720.

[7] F. Cuomo, S. Della Luna, E. Cipollone, P. Todorova y T. Suihko, "Topology Formation in IEEE 802.15.4: Cluster-Tree Characterization," en *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, mar. de 2008, págs. 276-281. DOI: 10.1109/PERCOM.2008.26. dirección: https://ieeexplore.ieee.org/abstract/document/4517407 (visitado 05-02-2024).

[8] "ZigBit Extension User Guide (USER GUIDE)," en,

[9] Z. Tao, S. Panwar, D. Gu y J. Zhang, “Performance analysis and a proposed improvement for the IEEE 802.15.4 contention access period,” en *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.*, vol. 4, 2006, págs. 1811-1818. DOI: 10.1109/WCNC.2006.1696571.

5 ANEXOS

Los códigos mencionados en el proyecto tanto para los nodos como para la aplicación de la PC, además de, los archivos generados de las pruebas realizadas se adjuntan en el siguiente repositorio:

https://github.com/BDavidr/Archivos-TIC_David-Rodriguez