ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

ALGORITMO PARA EL ACCESO AL CANAL EN TOPOLOGÍAS LINEALES QUE OPERAN CON EL PROTOCOLO IEEE 802.15.4

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

DANIEL ALEXANDER MATEUS CASTRO
daniel.mateus@epn.edu.ec

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

DMQ, abril 2023

CERTIFICACIONES

Yo, DANIEL ALEXANDER MATEUS CASTRO 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.

DANIEL ALEXANDER MATEUS CASTRO

Certifico que el presente trabajo de integración curricular fue desarrollado por DANIEL ALEXANDER MATEUS CASTRO, bajo mi supervisión.

ING. CARLOS ROBERTO EGAS ACOSTA, MSc
DIRECTOR

DECLARACIÓN DE AUTORÍA

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

DANIEL ALEXANDER MATEUS CASTRO
ING. CARLOS ROBERTO EGAS ACOSTA, MSc

DEDICATORIA

Dedicado con todo el cariño del mundo a mis padres que toda mi vida me han apoyado, me han dado las facilidades para cumplir esta meta y siempre han sabido brindarme todo su amor y confianza, a mi ñañito lindo, que más de una vez me motivo a no darme por vencido y a seguir dando lo mejor de mí pese a todas las adversidades y dificultades que hubo en el proceso.

Ellos son siempre mi motor y mi fuerza, y siempre lo serán.

AGRADECIMIENTO

Siempre agradecido con toda mi familia y con Diosito, cada uno de ellos me ha apoyado incondicionalmente y sin esperar nunca nada a cambio, lo han hecho siempre por amor.

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
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 REDES DE SENSORES INALÁMBRICOS 3
1.4.2 TOPOLOGÍAS LINEALES 4
1.4.3 ESTÁNDAR IEEE 802.15.4 4
1.4.3.1 Métodos de acceso al canal en IEEE 802.15.4 5
1.4.4 MICROCONTROLADOR ATMEGA256RFR2 7
1.4.4.1 Modos de operación. 7
1.4.4.2 Memorias. 9
1.4.4.3 Principales características de recepción y transmisión 9
2 METODOLOGÍA 11
2.1 INSTALACIÓN DEL SOFTWARE ATMEL STUDIO 7.0 11
2.1.1 CREACIÓN DE UN NUEVO PROYECTO 13
2.1.1.1 Archivos generados en la creación de un nuevo proyecto 15
2.2 INSTALACIÓN DEL SMARTRF PACKET SNIFFER 16
2.3 DISEÑO DEL ALGORITMO 17
2.3.1 PROBLEMÁTICA POR RESOLVER 17
2.3.2 DIAGRAMA DE FLUJO 18
2.3.3 CODIFICACIÓN DEL DIAGRAMA DE FLUJO 20
2.3.3.1 Código del primer nodo 21
2.3.3.2 Código del segundo nodo 22
2.3.3.3 Código del tercer nodo 23
2.3.3.4 Código del cuarto nodo 25
2.3.3.5 Código del quinto nodo 27
3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES 30
3.1 RESULTADOS 30
3.1.1 IMPLEMENTACIÓN DEL PROTOTIPO DE ENTORNO DE PRUEBAS 30
3.1.1.1 Proceso de quemado de los nodos 31
3.1.2 PRUEBAS REALIZADAS 32
3.1.2.1 Empleando únicamente el algoritmo diseñado 32
3.1.2.2 Empleando únicamente CSMA 36
3.1.2.3 Empleando el algoritmo diseñado y el protocolo CSMA juntos 37

3.1.2.4 Simulando un fallo en la transmisión 40
3.1.3 ANÁLISIS DE RESULTADOS 40
3.2 CONCLUSIONES 41
3.3 RECOMENDACIONES 42
4 REFERENCIAS BIBLIOGRÁFICAS 44
5 ANEXOS 45

RESUMEN

Este trabajo de titulación tiene como objetivo desarrollar e implementar un algoritmo de acceso al canal empleando el protocolo IEEE 802.15.4 en un prototipo con topología lineal, por lo que, este trabajo consta de tres secciones principales.

En el primer capítulo, se da una breve introducción al tema y se presentan las principales motivaciones de la realización de este trabajo, así como su alcance y objetivos a cumplir, posteriormente, en el marco teórico se repasa de la manera más resumida posible los temas más relevantes y necesarios para poder cumplir con el propósito planteado.

En el segundo capítulo, se presenta todo el proceso realizado hasta lograr la implementación funcional del prototipo, se detalla el proceso de instalación del software Atmel Studio 7.0 y Packet Sniffer, se presenta el diagrama de flujo a partir del cual se codifica el algoritmo de acceso al canal, así como el diagrama de tiempo y finalmente el código obtenido para cada nodo y que será usado en la parte de implementación.

En el tercer capítulo se presentan todos los resultados obtenidos de la implementación del algoritmo en el prototipo de la red, el desempeño obtenido y se realiza un contraste entre el método de acceso al canal usando el algoritmo diseñado y el método CSMA/CA, además de las conclusiones y recomendaciones pertinentes.

PALABRAS CLAVE: algoritmo de acceso al canal, IEEE 802.15.4, topología lineal, prototipo, acceso al canal.

ABSTRACT

The objective of this work is to develop and implement a channel access algorithm using the IEEE 802.15.4 protocol in a prototype with linear topology, so this work consists of three main sections.

In the first chapter, a brief introduction to the topic is given and the main motivations for this work are presented, as well as its scope and objectives to be met. Subsequently, in the theoretical framework, the most relevant and necessary topics are reviewed in the most summarized way possible in order to fulfill the proposed purpose.

In the second chapter, the entire process carried out to achieve the functional implementation of the prototype is presented, the installation process of the Atmel Studio 7.0 and Packet Sniffer software is detailed, the flowchart from which the channel access algorithm is coded is presented, as well as the timing diagram and finally the code obtained for each node and that will be used in the implementation part.

The third chapter presents all the results obtained from the implementation of the algorithm in the network prototype, the performance obtained, and a contrast is made between the channel access method using the designed algorithm and the CSMA/CA method, as well as the pertinent conclusions and recommendations.

KEYWORDS: channel access algorithm, IEEE 802.15.4, linear topology, prototype, channel access.

1 INTRODUCCIÓN

Hoy en día, existe una gran demanda del mercado de las comunicaciones inalámbricas aplicadas al campo de internet de las cosas (IoT). Las redes de sensores inalámbricos (WSN) se encuentran en un constante proceso de innovación y de mejora continua de la mano de una evolución continua en el campo de la microelectrónica, cada vez son más las situaciones donde se requiere el monitoreo de estructuras lineales con topologías lineales multicentro, como en oleoductos, carreteras, puentes, túneles, zonas protegidas e inclusive en escenarios más pequeños como en vigilancia de seguridad.

En una red de sensores inalámbricos, un conjunto de sensores se encargan de sondear el entorno que los rodea y compartir sus lecturas para evaluar la ocurrencia de un determinado evento o fenómeno que sea de interés para alguno de los casos mencionados anteriormente, especialmente en situaciones donde el acceso humano es sumamente limitado, ya sea por las condiciones climáticas del lugar, del terreno o de la infraestructura en general, permitiendo un monitoreo continuo de las estructuras del entorno con la mínima participación humana. La detección temprana de problemas puede ser de gran ayuda al momento de alertar a las autoridades y conseguir un tiempo de respuesta adecuado para solventar el problema con daños y consecuencias mínimas.

El protocolo 802.15.4 utiliza el mecanismo CSMA-CA para gestionar el control de acceso al medio cuando en la zona de cobertura existen cientos de nodos, en la práctica existen escenarios donde el número de nodos en la zona de cobertura es limitado y son pocos los nodos en cuestión que compiten por el acceso al canal en comparación con la cantidad total de nodos ajenos al proyecto que se desea implementar, situación para la cual el protocolo IEEE 802.15.4 fue diseñado.

La principal ventaja de los nodos a trabajar radica en ser una alternativa económica, que trabaja con baterías y posee un tamaño reducido, lastimosamente, tanto dicho nodo como también el protocolo IEEE 802.15.4 presentan una serie de desventajas como una memoria y alcance limitados, además de bajas tasas de transmisión.

Con el afán de lograr una optimización de los recursos de cada nodo, del espectro radioeléctrico, minimizar los tiempos de retardo y aprovechar al máximo tanto la batería del nodo como el número de nodos disponibles en un entorno real, es necesario proponer un protocolo de acceso al canal sencillo que se adapte a las características de topologías lineales.

1.1 OBJETIVO GENERAL

Proponer un algoritmo de acceso al canal en topologías lineales que empleen el protocolo IEEE 802.15.4.

1.2 OBJETIVOS ESPECÍFICOS

1. Profundizar en el estudio del protocolo IEEE 802.15.4 y revisar el estado del arte de métodos de acceso al canal.
2. Proponer un algoritmo de acceso al canal en topologías lineales.
3. Implementar y probar el algoritmo en un prototipo.
4. Analizar los resultados obtenidos de las pruebas realizadas y constatar el funcionamiento óptimo del algoritmo implementado contrastando con el uso de protocolos de acceso al canal como CSMA.

1.3 ALCANCE

En este trabajo de titulación se pretende emplear toda la bibliografía relacionada al protocolo IEEE 802.15.4 y a los métodos de acceso al canal, con el fin de profundizar en el estudio del protocolo, hasta tener claro su funcionamiento y los tipos de métodos de acceso al canal que existen.

Empleando dicha bibliografía, se plantea estudiar implementaciones relevantes del protocolo y las características técnicas de las topologías lineales con el fin de obtener las condiciones necesarias para el número de nodos en la zona de cobertura hasta determinar qué factores influyen en el método de acceso al canal.

Después de haber analizado el funcionamiento del protocolo IEEE 802.15.4, topologías lineales y métodos de acceso al canal, se propondrá un algoritmo en base a la topología lineal que permita optimizar el acceso al canal de los nodos haciendo un uso eficiente de los recursos y procesos que deberán ser realizados, hasta obtener un diagrama de flujo que permita su posterior codificación e implementación.

A continuación, se implementará un prototipo con cinco nodos que trabajan con el protocolo IEEE 802.15.4 para lo cual se estudiará el funcionamiento de los nodos en cuestión, como se configuran, el kit de desarrollo que utiliza y como se transmiten los datos con el fin de obtener un prototipo listo para su implementación junto con el algoritmo, luego se codificará dicho algoritmo y se realizarán las pruebas necesarias hasta conocer como el algoritmo propuesto controla el acceso al canal de los nodos.

Finalmente, con los resultados obtenidos de las pruebas realizadas en escenarios definidos durante el desarrollo del trabajo de integración curricular, se realizará un análisis de dichos resultados y se verificará el funcionamiento óptimo del algoritmo implementado hasta lograr una contribución al estado del arte con respecto a implementaciones anteriores.

1.4 MARCO TEÓRICO

1.4.1 REDES DE SENSORES INALÁMBRICOS

Una red de sensores inalámbricos (WSN) es una infraestructura compuesta por elementos de detección o medición, computación y comunicación que brindan a un administrador la capacidad de instrumentar, observar y reaccionar ante eventos y fenómenos en un entorno específico [1].

Una red WSN típica consta de dos componentes principales, el nodo y la estación base, como se muestra en la Figura 1.1, en muchas ocasiones un nodo puede asumir el rol de la estación base sin ningún problema.

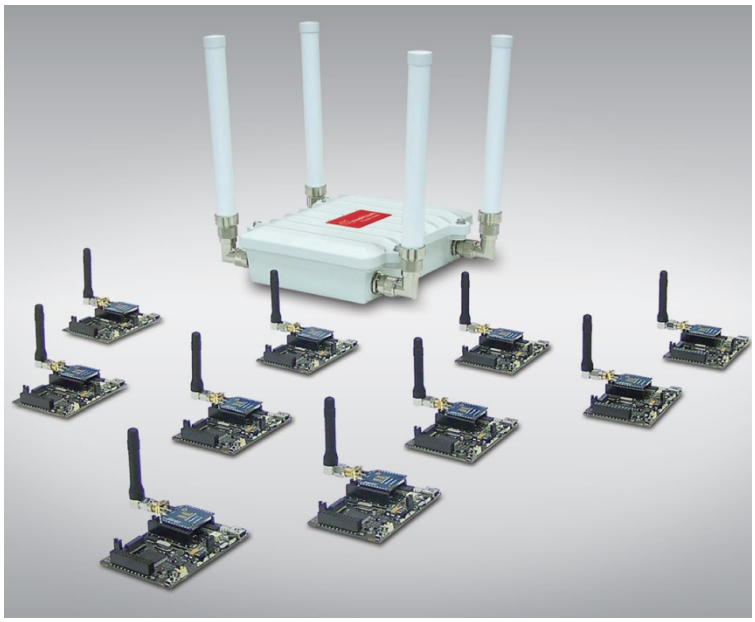


Figura 1.1 Red de sensores inalámbricos (WSN) [2]

Un nodo es un dispositivo que normalmente está equipado con capacidades de detección, procesamiento y comunicación, y es responsable de medir los parámetros asociados con una determinada solicitud. Una estación base es responsable de capturar y brindar acceso a todos los datos de medición de los nodos y, a veces, puede proporcionar servicios de puerta de enlace para permitir que los datos se administren de forma remota.

Las WSN normalmente emplean estándares de red de área personal inalámbrica (WPAN) o red de área amplia de baja potencia (LPWAN) para transmitir datos de medición a la estación base. Estos estándares incluyen IEEE 802.15.4, ZigBee y Bluetooth. No existe una única solución de conectividad que se considere adecuada para todas las WSN, y la elección del estándar depende completamente de los requisitos de comunicación y las limitaciones de recursos de una aplicación en particular [3].

1.4.2 TOPOLOGÍAS LINEALES

Se conoce como topología de red al estudio de la disposición geométrica de los distintos elementos de una red (nodos, enlaces). Las topologías físicas describen como se acoplan los diferentes elementos de una red entre sí, mientras que las topologías lógicas se refieren al flujo de información entre los distintos sitios en una red particular.

La topología lógica por trabajar es la topología llamada lineal, donde un nodo principal o fuente está conectado directamente a otro nodo destino, haciendo que cada dispositivo tenga dos vecinos para la comunicación.

En una topología lineal, cada vez que un nodo quiere comunicarse con otro nodo, envía un mensaje por el canal común, este mensaje es accesible por todos los nodos lógicos que se encuentran al alcance en la red, aunque el destinatario es en realidad el que acepta y procesa el mensaje [4][5].

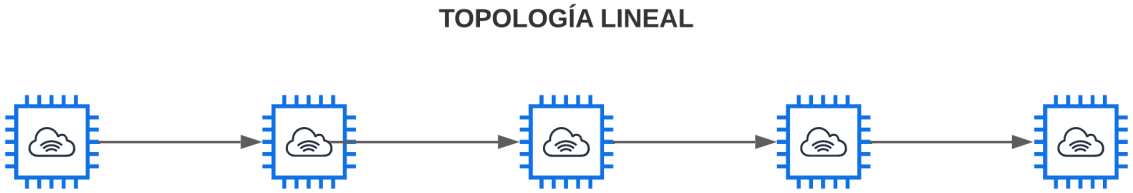


Figura 1.2 Topología lineal

1.4.3 ESTÁNDAR IEEE 802.15.4

El grupo de trabajo IEEE P802.15 tenía el objetivo de diseñar un protocolo de red de área personal inalámbrica de baja velocidad, bajo rango, baja complejidad y potencia mediante la estandarización del protocolo IEEE 802.15.4.

En comparación con otros estándares inalámbricos como IEEE 802.11 e IEEE 802.15.1, el menor consumo de energía y producción en unidades de tamaño físico pequeño con bajo costo por unidad son características destacadas del protocolo IEEE 802.15.4.

La facilidad de instalación, la transferencia de datos confiable y la duración razonable de la batería son otras de las atracciones de IEEE 802.15.4 que lo convierten en un protocolo de subcapa estándar para algunos protocolos.

IEEE 802.15.4 define las especificaciones de la subcapa física (PHY) y de control de acceso al medio (MAC) para redes de área personal inalámbricas de baja potencia y velocidad de datos baja (LR-WPAN). El estándar es responsable de la detección de energía de los canales, la indicación de la calidad del enlace (LQI) para recibir paquetes, la evaluación del canal limpio (CCA) y la selección de la frecuencia del canal. Opera en tres bandas sin licencia que cubren 2,4 GHz, 915 MHz y 868 MHz con una velocidad máxima de datos de hasta 250 Kbps, 30 Kbps y 20 Kbps, respectivamente. Hay 16 canales a 2,4 GHz, 10 canales a 902 a 928 MHz y 1 canal a 868 MHz. Sin embargo, 2,4 GHz es la banda de frecuencia más popular en las aplicaciones. Esta banda ofrece un ancho de banda de 5 MHz para cada canal y una capacidad de transmisión de código de 2 Mchip/s mediante el uso de espectro ensanchado de secuencia directa (DSSS), donde cada símbolo de datos (4 bits) se codifica en un código de pseudo-ruido (PN) de 32 bits. Utiliza modulación de desplazamiento de fase en cuadratura compensada (O-QPSK) en su capa PHY. La potencia de salida máxima de la radio IEEE 802.15.4 suele ser de 0 dBm; las sensibilidades del receptor son -85 dBm que admiten un rango de transmisión de hasta 100 m.

Un paquete IEEE 802.15.4 comienza con una secuencia de preámbulo (PRE), continúa con el encabezado de sincronización (SYN), luego el encabezado PHY, como se puede ver en la Figura 1.3 El tamaño máximo de la unidad de transmisión (MTU) del protocolo es de 127 octetos. También es posible transportar paquetes IPv6 dentro del protocolo IEEE 802.15.4 a través de una capa de adaptación implementada que fragmenta octetos IPv6, que es un mínimo de 1280 en paquetes IEEE 802.15.4 de 127 octetos [6].

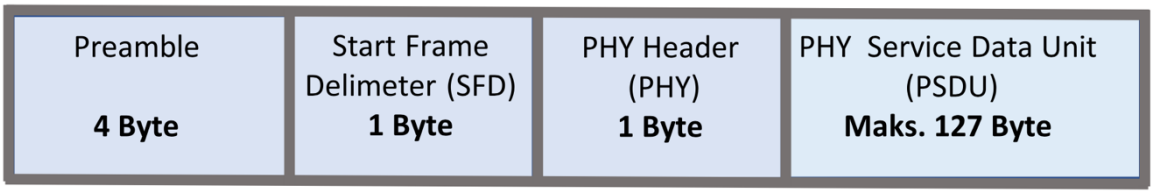


Figura 1.3 Estructura de un paquete IEEE 802.15.4 [6]

1.4.3.1 Métodos de acceso al canal en IEEE 802.15.4

En IEEE 802.15.4 la subcapa MAC es responsable del acceso al canal de radio, la sincronización de los intervalos de tiempo, la conexión y la desvinculación de la red de área personal inalámbrica, la asignación de intervalos de tiempo garantizados (GTS) y la operación CSMA/CA para el control de acceso al canal. La capa MAC del protocolo

transmite tramas después de realizar un proceso de evaluación de canal limpio. Para reducir las posibilidades de transmisiones simultáneas entre múltiples nodos, IEEE 802.15.4 MAC trabaja con dos algoritmos CSMA/CA, CSMA/CA con ranuras y sin ranuras. En el modo ranurado, un coordinador de red es responsable de la sincronización de tiempo entre los dispositivos que se comunican. El canal se comparte entre nodos individuales en intervalos de tiempo predeterminados, luego el coordinador determina la secuencia de intervalos de tiempo dividiendo el tiempo en super tramas a través de tramas de baliza. Cada marco de baliza incluye información sobre la configuración de la red y proporciona sincronización de tiempo para el uso del canal, que contiene una parte activa y una parte inactiva donde los nodos pueden dormir y conservar energía. El encabezado MAC IEEE 802.15.4 se ilustra en la Figura. 1.4.

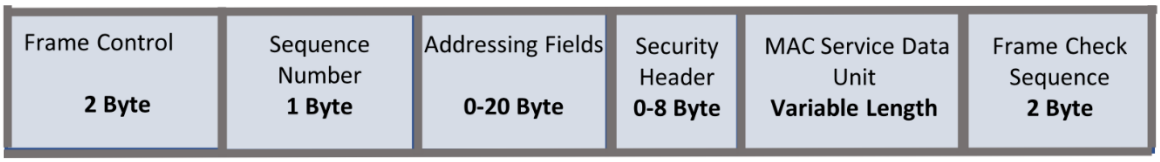


Figura 1.4 Subcapa MAC en IEEE 802.15.4 [6]

El modo sin ranura es más apropiado para aplicaciones basadas en eventos. En este modo, los nodos que utilizan el canal pueden realizar solicitudes de transmisión en cualquier momento. En la primera fase de transmisión de datos, el remitente retrocede durante un tiempo aleatorio para evitar posibles colisiones con otros nodos. El tiempo aleatorio elegido no depende del período de interrupción del otro nodo en esa red de área personal (PAN) en particular. A continuación, el emisor espera otro período de 8 símbolos para la detección de energía. En el caso de detectar que el canal está ocupado, se repite el mismo proceso hasta alcanzar el valor de back off máximo permitido. En este modo, las transmisiones, las evaluaciones de canales despejados y los retrocesos se alinean con unidades de tiempo denominadas “períodos de back off”, que son de 320 μs.

La segunda fase es opcional, ya sea que la opción de solicitud de acuse de recibo (ACK) esté habilitada o no. Si ACK está habilitado, la comunicación se logra si el remitente recibe una solicitud de ACK en un período de espera de MAC ACK permitido. En el caso de que el paquete ACK no se tome en este intervalo de tiempo, el proceso se repetirá hasta que se alcance el número máximo permitido de retransmisión de tramas [6].

1.4.4 MICROCONTROLADOR ATMEGA256RFR2

Microcontrolador de la marca ATMEL, tiene 8 bits y su estructura está basada en AVR y arquitectura RISC, actúa en conjunto con un transceptor de alta tasa de transferencia de datos para la banda 2,4 GHz ISM.

Debido a que ejecuta potentes instrucciones en un solo ciclo de reloj, el nodo es capaz de obtener rendimientos que rondan los 1 MIPS por MHz, lo cual permite que el usuario pueda aprovechar al máximo el bajo consumo energético en contraste con la velocidad de procesamiento de los datos.

Posee un transceptor de radio que brinda altas de tasas de transferencia de datos que rondan desde los 250 Kbyte/s hasta los 2 Mbyte/s, el manejo de tramas junto con la sensibilidad del receptor son excelentes y además cuenta con una alta potencia de salida de transmisión capaz de habilitar comunicaciones inalámbricas fiables y robustas cumpliendo con los estándares IEEE 802.15.4-2003/2006/2011 [7].

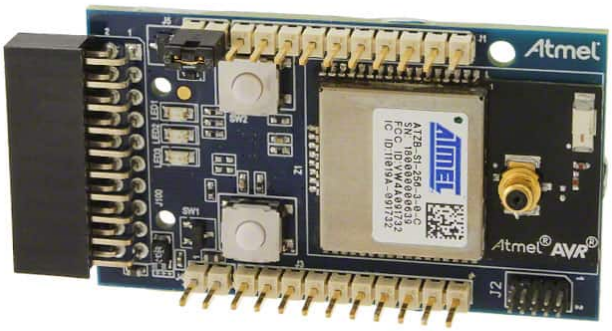


Figura 1.5 Atmel ATZB – 256RFR2 [7]

1.4.4.1 Modos de operación.

Para poder entender de manera adecuada el funcionamiento del transceptor y poder lograr una codificación adecuada de este, se deben entender los principales modos de operación en los que puede trabajar.

Un detalle importante por considerar es que los nodos trabajan en un modo de comunicación half dúplex, es decir, que no posee una comunicación bidireccional.

El transceptor posee tres modos de operación, modo de operación básico o elemental, modo de operación extendido y modo promiscuo.

En este trabajo solo ahondaremos en el modo elemental, pues este es el compila todos los estados que son necesarios para cumplir con el objetivo del algoritmo, dichos estados son:

- **Modo de espera:** se lo conoce también con el nombre de Stand By y es el estado en el cual se encuentra el transceptor tras ser encendido o reiniciado, en dicho estado el transceptor se mantiene a la espera de una instrucción y en general es muy similar al modo apagado, con la diferencia de que el oscilador está ejecutándose, por lo que, no se transmiten ni capturan ningún tipo de señal.
- **Modo sleep:** modo utilizado cuando el transceptor no se encuentra en funcionamiento y su objetivo es realizar un ahorro de la batería. El transceptor y los circuitos dejan de ser alimentados pero los registros continúan funcionando, además, el contenido del buffer es eliminado.
- **Modo de recepción:** aquí el transceptor se encuentra habilitado para recibir tramas entrantes, las procesa y pasa a un sub-estado llamado “ocupado receptando” y permanece en este hasta que termine la recepción de la trama, el cambio de estado en este sub-estado es imposible debido a que se debe garantizar que la trama llegue de una manera correcta, los bytes son almacenados en el buffer hasta que termine el proceso de recepción.
- **Modo de transmisión:** en este modo el transceptor sintoniza un canal y frecuencia para realizar la transmisión, posteriormente, pasa a un sub-estado llamado “ocupado transmitiendo”, en el cual permanece hasta que finalice la transmisión [7] [8].

Como el nodo maneja los estados mencionados se presenta en la figura 1.6.

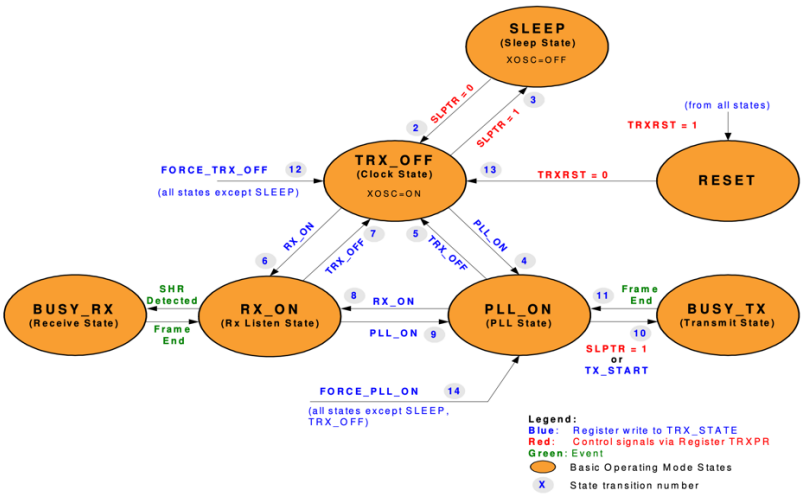


Figura 1.6 Manejo de estados en Atmel ATZB – 256RFR2 [7]

1.4.4.2 Memorias.

El microcontrolador Atmega256RFR2 posee una arquitectura AVR, la misma que tiene tres espacios de memoria fundamentales, el primero es la memoria EEPROM utilizada para el almacenamiento de los datos, el segundo es la memoria flash y por último la memoria SRAM o memoria del programa.

- **Memoria EEPROM:** posee un tamaño de 8 Kbps, en la figura 1.7 se pueden observar los espacios de memoria.

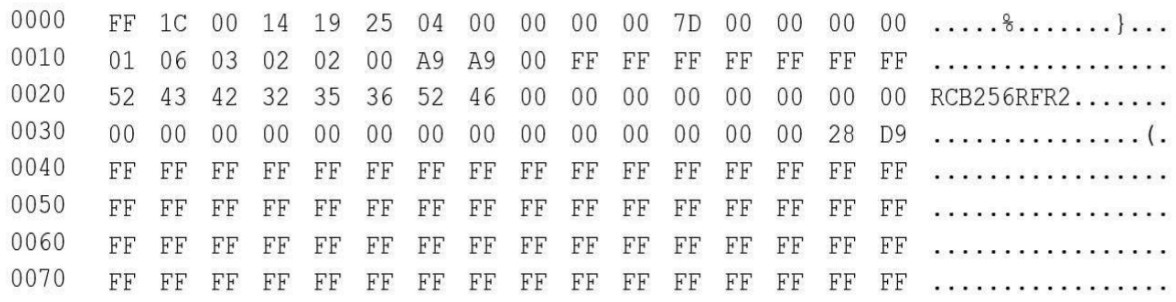


Figura 1.7 Espacios de memoria EEPROM [9]

- **Memoria Flash:** memoria que puede ser reprogramada en el sistema, posee un tamaño de 256 Kbps y por seguridad es dividida en dos partes, el programa de aplicación y de arranque.
- **Memoria SRAM:** posee un tamaño de 32 Kbps [10].

1.4.4.3 Principales características de recepción y transmisión

Las principales características del nodo transceptor son las siguientes:

- Velocidades de transmisión excelentes que rondan los 250 Kbps, 500 Kbps, 1Mbps y hasta 2 Mbps.
- Potencia de transmisión de salida aproximada de 3.5 dBm
- Sensibilidades en la recepción cercanas a -100 dBm
- Posee un sintetizador “Phase Locked Loop” (PLL) que permite espaciado entre canales de 5 MHz y 500 KHz para trabajar en bandas de 2.4 GHz
- Bajo consumo de energía que ronda valores de 1.8V a 3.6V y 10.1 mA a 18.6 mA, específicamente 6 mA en recepción y 14.5 mA a 3.5 dBm en transmisión.
- Hardware diseñado en base a las principales características del estándar IEEE 802.15.4

    - Evaluación del canal libre (CCA)
    - Detección de energía (ED)
    - Detección de portadora (CS)
    - Control y transmisión de tramas
- Posee también un buffer de 128 Bytes para poder almacenar la trama que ingresa o sale del nodo transceptor.
- Amplificador con la capacidad de eliminar del espectro los lóbulos laterales de alta potencia para la transmisión.
- Interfaz amigable con el usuario que permite.
    - Acceder al buffer que se encarga del almacenado de las tramas, en especial la de acceso a los registros
    - Realizas interrupciones concretas para poder controlar de mejor manera la transmisión y recepción de las tramas
- Soporta aplicaciones como: IEEE 802.15.4 2011/2006/2003, ZigBee, WirelessHART, ISM, RF4CE, SP100 y 6LoWPAN [10][11].

2 METODOLOGÍA

Este trabajo de integración curricular ha sido desarrollado con un enfoque del tipo de investigación aplicada empleando un tipo de trabajo experimental para la recolección de información y la debida obtención de resultados, por lo que, en esta sección se presentan todas las acciones necesarias para cumplir con el objetivo de la implementación del algoritmo de acceso al canal en topologías lineales que trabajan con el protocolo IEEE 802.15.4.

2.1 INSTALACIÓN DEL SOFTWARE ATMEL STUDIO 7.0

Atmel Studio, o también conocido como Microchip Studio, es un entorno de desarrollo integrado (IDE) empleado para el desarrollo y el depurado de aplicaciones para microcontroladores SAM y AVR. Combina una gran cantidad de herramientas de desarrollo para facilitar el entorno de programación el lenguaje C/C++ o en código ensamblador.

Para la instalación de este software primero debemos descargar el ejecutable de Microchip Studio versión 7, para lo cual, debemos ingresar a la página de Microchip Studio, aquí escoger la opción Tools and resources > Develop > Microchip Studio for AVR and SAM devices, en esta página, podemos descargar la versión más reciente del IDE, siendo esta la 7.0.2594, misma con la cual se realizó este trabajo de titulación.

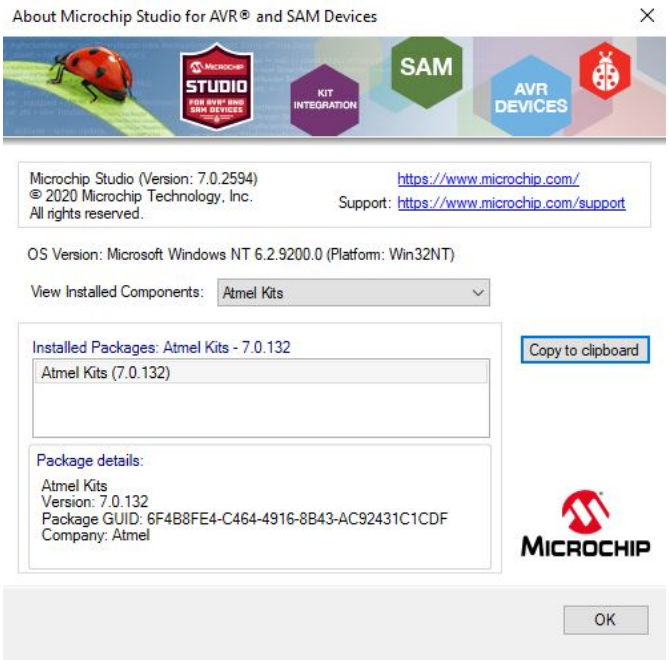


Figura 2.1 Versión de Microchip Studio instalada en el equipo de trabajo.

El siguiente paso será ejecutar el archivo descargado y proceder con la instalación, este proceso es bastante intuitivo y no se ahondará en el mismo, una vez instalado se debe tener la interfaz presentada en la figura 2.2.

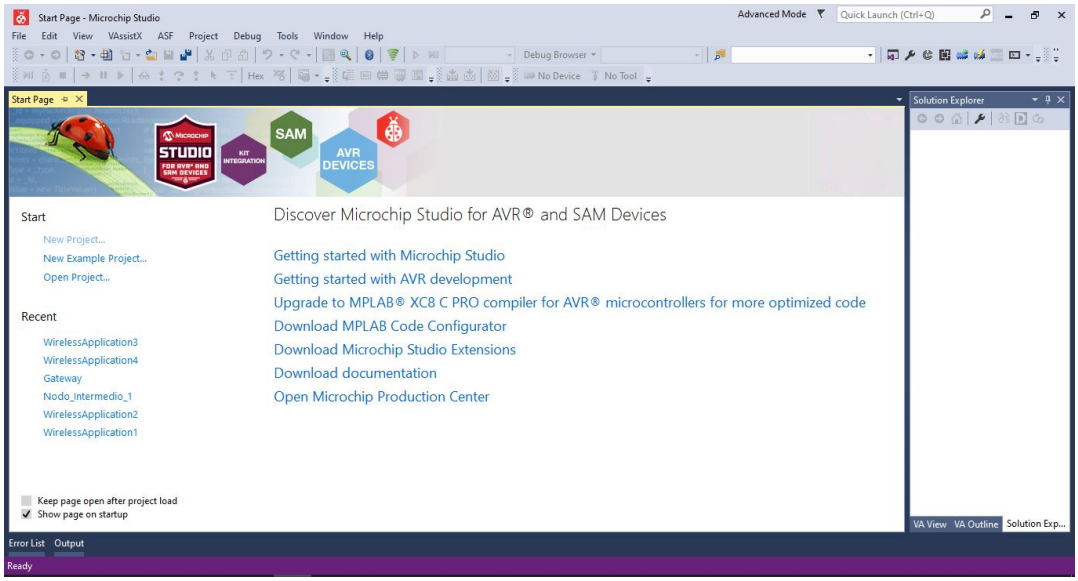


Figura 2.2 Interfaz de usuario de Microchip Studio

Ahora, se deben instalar las extensiones necesarias para que el hardware sea reconocido por el equipo y podamos trabajar con el mismo, debemos dirigirnos a la pestaña Tools > Extensions and Updates, y debemos instalar la extensión “Atmel Software Framework” como se puede observar en la figura 2.3.

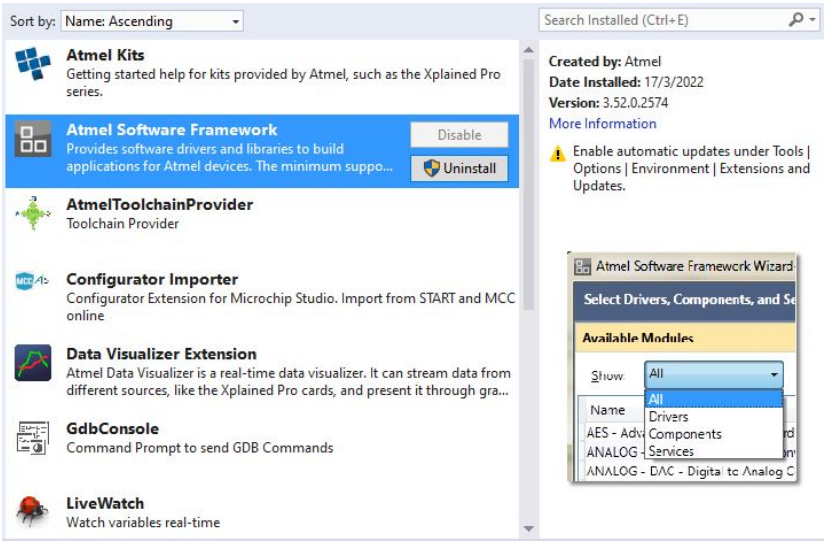


Figura 2.3 Extensión Atmel Software Framework

Otra extensión necesaria se encuentra en la pestaña Online, y se trata del “Wireless Composer 7.0”, como se presenta en la figura 2.4, esta proporciona un analizador de

rendimiento inalámbrico y el asistente de proyectos MAC IEEE 802.15.4. Con el analizador de rendimiento inalámbricos podemos explotar al máximo las diversas características y capacidades de los nodos transceptores Atmel IEEE 802.15.4. El asistente de proyectos proporciona una interfaz más amigable para crear proyectos con los microcontroladores de Atmel.

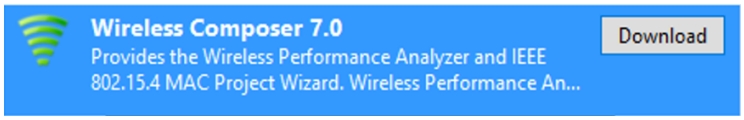


Figura 2.4 Extensión Wireless Composer 7.0

Con esas extensiones instaladas se concluye el proceso de instalación del software Atmel Studio 7.0.

2.1.1 CREACIÓN DE UN NUEVO PROYECTO

Para la creación de un nuevo proyecto es necesario seguir los pasos que se presentan a continuación, primero se debe ir a la pestaña File > New > Project.

Aquí el asistente nos pedirá escoger que clase de proyecto queremos crear, como se puede ver en la figura 2.5, la opción a escoger será “GCC C IEEE 802.15.4 MAC Wireless Executable Project” cuyo lenguaje de programación es C/C++.

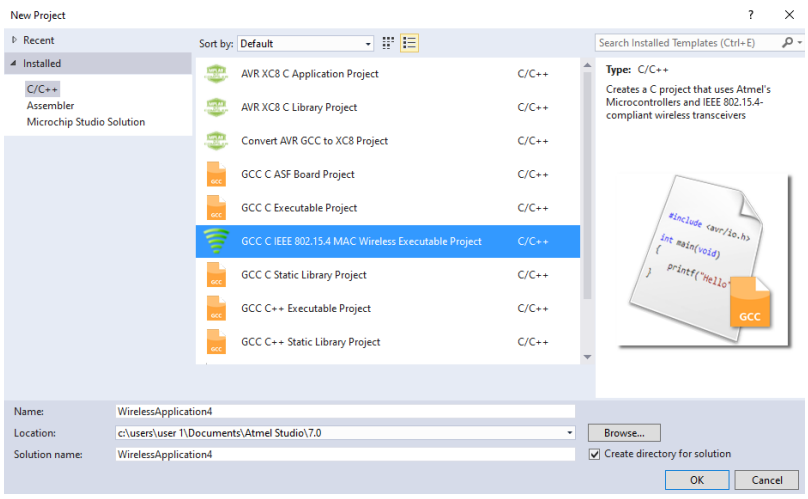


Figura 2.5 Proyecto IEEE 802.15.4

Después debemos escoger la opción de proyecto genérico, y es aquí donde escogeremos el microcontrolador con el que trabajaremos, como ya se mencionó en la sección 1.4.4 del marco teórico de este trabajo, se usará el Atmega256RFR2, esto se puede observar en la figura 2.6.

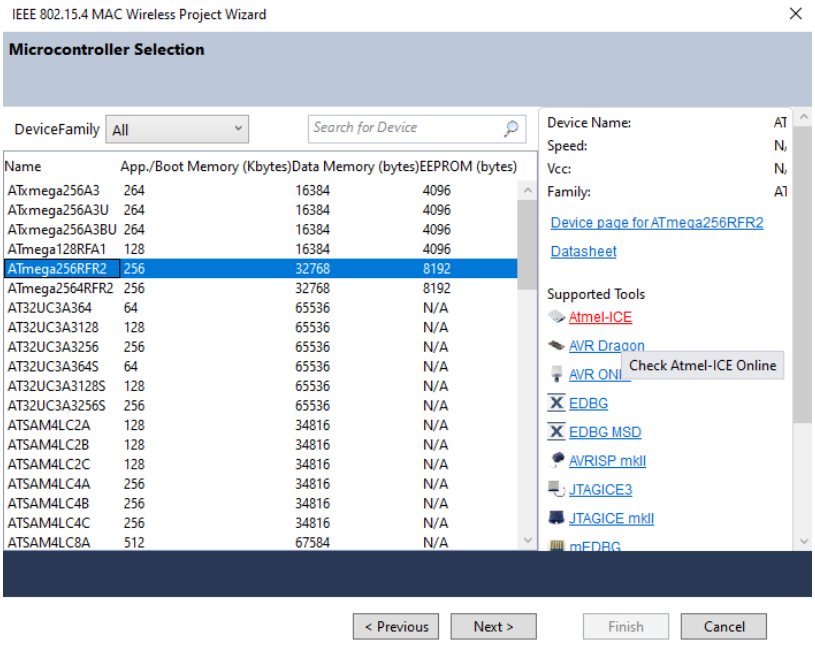


Figura 2.6 Ventana de selección de microcontrolador

Después se nos presentará una ventana con las configuraciones elementales del proyecto, como se puede observar en la figura 2.7, estas opciones van desde el transceptor, hasta la ID de la red de área personal (PAN ID).

Es importante mencionar que las opciones como la frecuencia, tasa de transferencia, diversidad de antenas, direcciones y PAN pueden ser cambiadas posteriormente en los archivos de configuración.

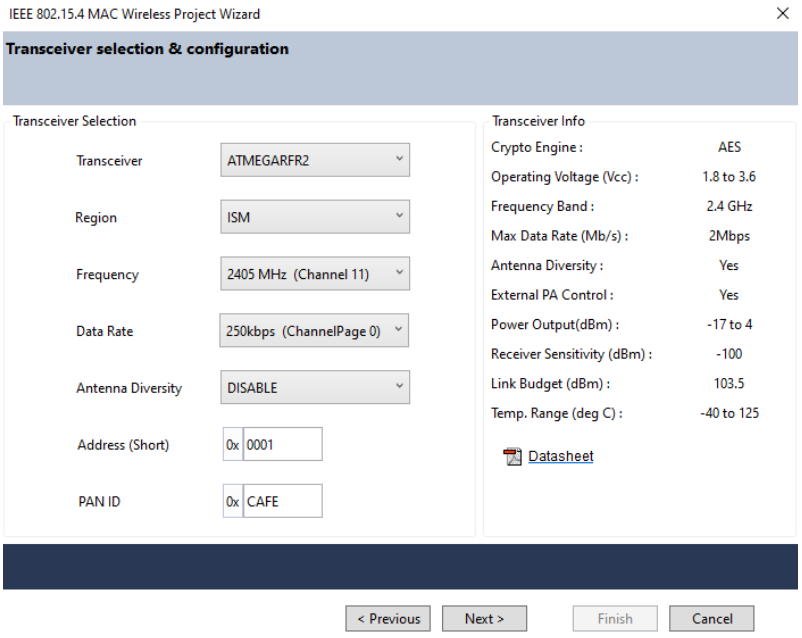


Figura 2.7 Configuraciones elementales del proyecto

Finalmente, se presenta un resumen de las configuraciones seleccionadas como se puede observar en la figura 2.8 y se concluye con la creación del nuevo proyecto.

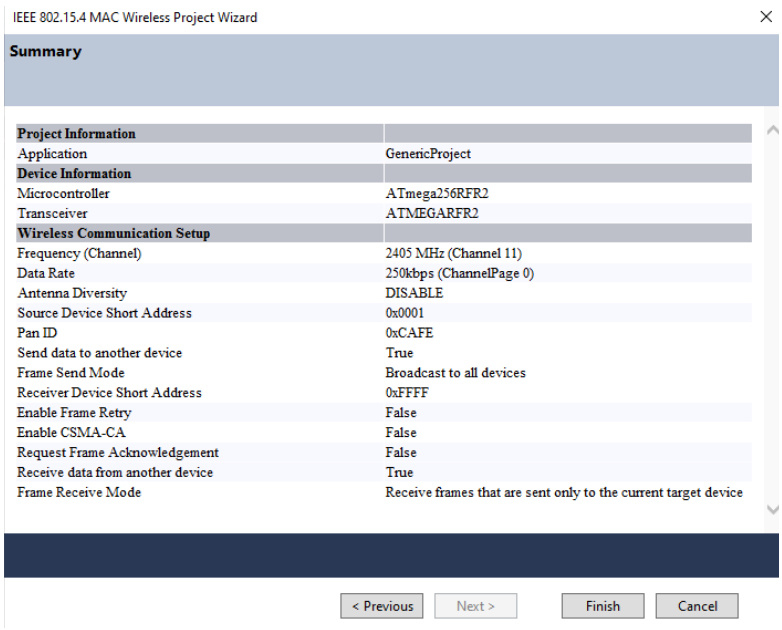


Figura 2.8 Resumen de configuraciones seleccionadas

2.1.1.1 Archivos generados en la creación de un nuevo proyecto

Existen tres principales tipos de archivos que se general al crear un nuevo proyecto, estos varían dependiendo de las opciones que se escojan al momento de crear un nuevo proyecto, en caso de escoger que el nodo transmita y reciba información como se puede observar en la figura 2.9 se tienen los siguientes archivos.

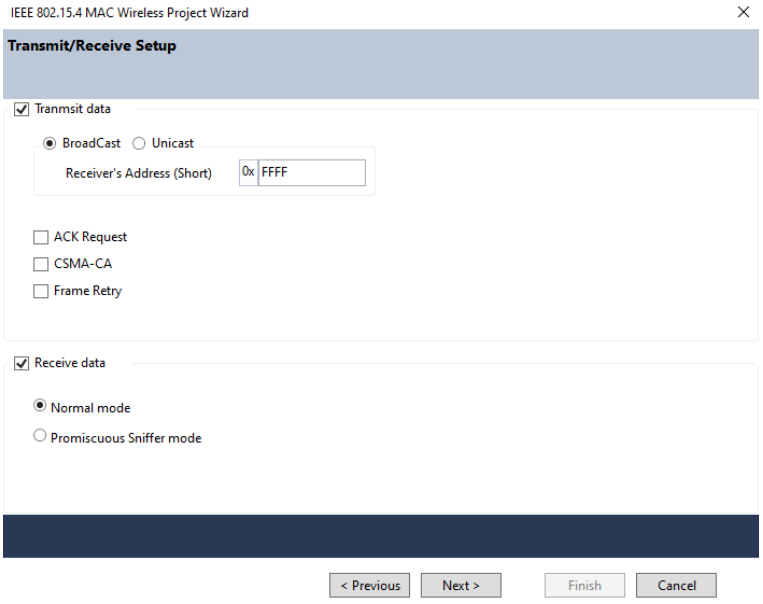


Figura 2.9 Configuración de transmisión y recepción

- **Archivo main.c:** en este se inicializan todas las funciones básicas del nodo transceptor.
- **Archivos de usuario:** el principal es usr_wireless.c donde se coloca todo el código de los nodos y las tareas que va a ejecutar.
- **Wireless:** carpeta en la cual se encuentran todos los archivos "source" (fuente) de los nodos transceptores, incluyendo aquellos archivos que fueron creados con el proceso de creación de nuevo proyecto y su asistente. Aquí se encuentran los archivos wireless_api.c, wireless_api.h y wireless_config.h que son los encargados de proporcionar las funciones para modelar el nodo, generar las instrucciones para la transmisión y recepción, y seleccionar una red inalámbrica, el identificador de los transceptores, el identificador de la red o el canal de transmisión (frecuencia).

Estos archivos se pueden apreciar en la figura 2.10

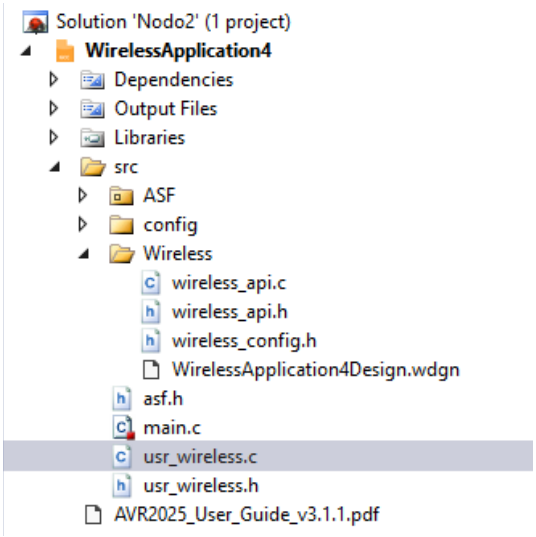


Figura 2.10 Archivos de configuración del proyecto

2.2 INSTALACIÓN DEL SMARTRF PACKET SNIFFER

Un sniffer es una herramienta que permite la detección de redes, esta es una forma de análisis de paquetes de datos que permite la supervisión de redes en tiempo real. La detección de red se utiliza para diagnosticar problemas de red y analizar la actividad general de la red y las aplicaciones. Con información a nivel de paquete, los administradores pueden identificar ralentizaciones, categorizar y evaluar el tráfico e identificar riesgos de seguridad.

Para la instalación de este software primero debemos descargar el ejecutable de Packet Sniffer de Texas Instruments, para lo cual se debe entrar a la página de Texas Instruments y en el buscador de la página escribir “Packet Sniffer Calculation Tool” y seleccionar la primera opción, aquí se debe descargar el software “PACKET-SNIFFER” puesto que este es compatible con el la versión del dongle que se empleará para este trabajo de titulación (CC2531), el proceso de instalación es bastante intuitivo y no se ahondará en el mismo.

Posteriormente, se debe conectar el sniffer al puerto USB de nuestro computador y verificar que este sea reconocido por el equipo, esto se debe hacer con la herramienta “Administrador de dispositivos” propia de Windows, basta con actualizar los drivers y el sniffer queda instalado correctamente, al abrir el software se debe tener la interfaz presentada en la figura 2.11.

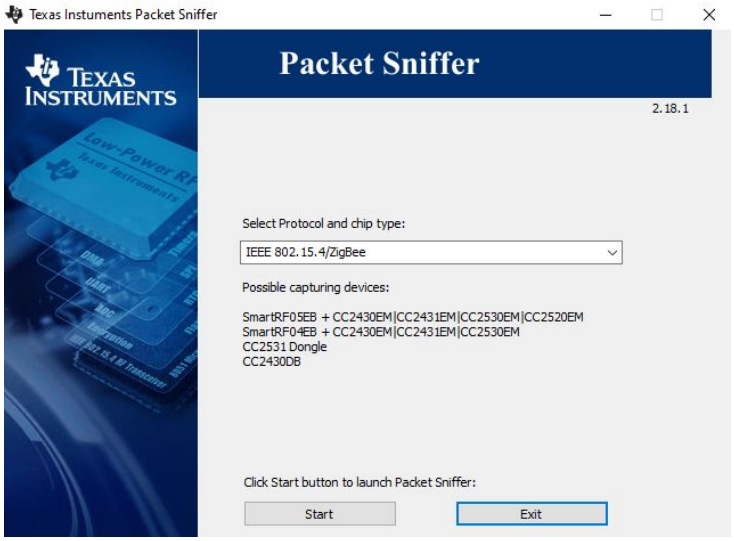


Figura 2.11 Software “Texas Instruments Packet Sniffer”

2.3 DISEÑO DEL ALGORITMO

2.3.1 PROBLEMÁTICA POR RESOLVER

La gran mayoría de algoritmos de acceso al canal en la actualidad están diseñados para trabajar con cientos o miles de nodos en grandes zonas de cobertura, por lo que al momento de querer trabajar con una topología lineal en una zona reducida donde el número de nodos sea limitado, el resultado es una implementación que requiere de una mayor complejidad, pues en dichos algoritmos se debe competir por el acceso al canal, resultando en que estos no sean capaces de aprovechar de manera eficiente las características de los nodos al transmitir información en una red pequeña.

El propósito del algoritmo presentado en este trabajo de titulación busca dar una solución a esa problemática presentando un algoritmo que permita un acceso óptimo al canal en una topología lineal empleando el protocolo IEEE 802.15.4 para lo cual se plantea realizar la transferencia de un “token” que habilite la transmisión entre los nodos, así se aprovecha al máximo los recursos de cada uno de los nodos que participen en la transmisión, además, a través de un código eficiente se plantea disminuir tiempos de retardo, maximizar la vida útil de la batería de cada nodo transmisor y lograr una transmisión y recepción de la información ordenada.

2.3.2 DIAGRAMA DE FLUJO

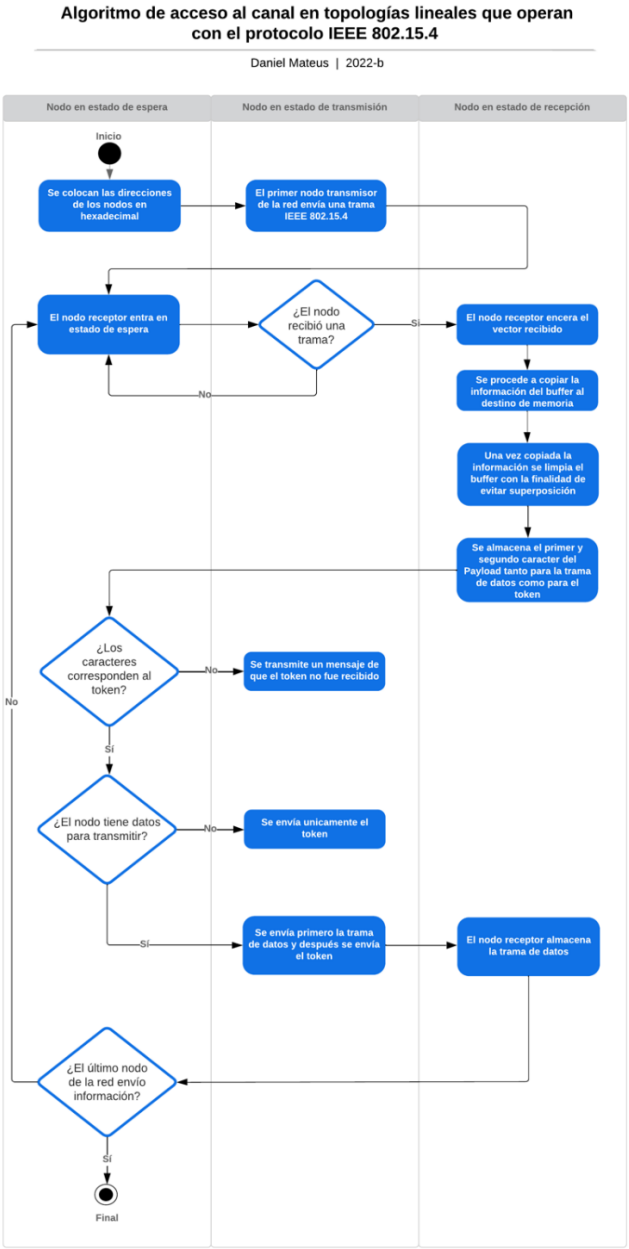


Figura 2.12 Diagrama de flujo a implementar

Se presenta en la figura 2.12 el diagrama de flujo correspondiente, en donde se contemplan los tres estados principales de los nodos transceptores, el detalle de funcionamiento de este se presenta a continuación.

Primero, es necesario colocar las direcciones en cada uno de los nodos en hexadecimal, posteriormente, el primer nodo de la red envía una trama mientras que el resto de los nodos entran en estado de espera, el nodo receptor verifica si efectivamente se envió una trama IEEE 802.15.4 y comienza el procesamiento de la trama, primero se encera el vector, después, se almacena la información del buffer en memoria, una vez copiada la información se limpia el buffer para evitar superposición, se almacenan el primer y segundo carácter de la trama, tanto si es de datos como el token, ahora se realiza una comparación, en caso de no tratarse del token la transmisión finaliza con un mensaje en broadcast de "TKN no recibido", en caso de que se tratase del token la transmisión continua y se comprueba que el nodo tenga datos para transmitir, en caso de no tener datos simplemente se transmite el token para que el siguiente nodo pueda continuar con el proceso, en caso de poseer datos se envía primero la trama de datos y después el token, el nodo receptor almacena la trama de datos para enviarla al siguiente nodo después de haber recibido el token, finalmente se verifica si el último nodo de la red ha transmitido información y se da por concluido el proceso de transmisión en la red, caso contrario se repite el proceso.

Es de suma importancia tener en cuenta que los nodos transceptores Atmel 256RFR2, deben ser programados con un lenguaje C/C++ y que manejan una comunicación half dúplex.

En la figura 2.13 se presenta un diagrama de flujo de tiempo en el cual se muestra cómo se enviarán las tramas entre los nodos considerando los tiempos necesarios para el procesamiento de estas y el almacenado que realiza cada nodo, así como el token, el propósito de este gráfico es lograr una mejor comprensión del funcionamiento de la red y del algoritmo.

En este gráfico se consideran los tiempos de procesamiento (Tp) que realiza el nodo después de recibir cada trama ya sea de datos o el token.

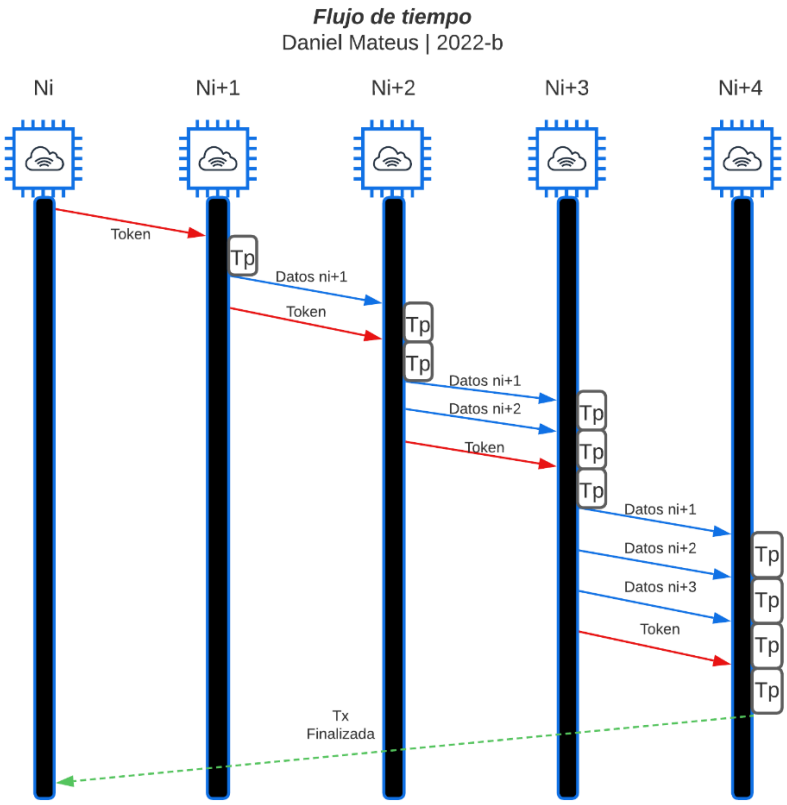


Figura 2.13 Diagrama de flujo de tiempo

2.3.3 CODIFICACIÓN DEL DIAGRAMA DE FLUJO

El código implementado en los nodos se realizó en base a los diagramas presentados en las figuras 2.12 y 2.13 de la sección anterior y teniendo en cuenta un entorno de prueba de 5 nodos.

Es necesario crear una estructura para una trama IEEE en lenguaje C, una estructura es un grupo de variables relacionadas bajo el mismo nombre, de esta estructura se trabaja con el payload en donde se enviará el token encargado de habilitar la transmisión, la estructura codificada se presenta a continuación.

```
/*----------------------ESTRUCTURA DE LA TRAMA IEEE 802.15.4----------------------*/
typedef struct
{
	uint8_t longitud;                                   // Longitud de la Trama
	uint16_t fcf;                                       // Frame Control Field
	uint8_t num_sec;                                    // Número de secuencia
	uint16_t d_PAN;                                     // Dirección PAN
	uint16_t d_dstn;                                    // Dirección de destino
	uint16_t d_orgn;                                    // Dirección de origen
	uint8_t payload[max_dato];                          // Carga útil
	uint16_t fcs;                                       // Frame Control
Sequence
}trama_ieee_802_15_42;

trama_ieee_802_15_42 trama_recibida;
```

Ahora, se presenta la función principal de cada nodo, aquella en donde se encuentra la codificación en sí, pues los archivos de configuración son sumamente extensos y no aportan con nada relevante a la codificación.

2.3.3.1 Código del primer nodo

El primer nodo de la red es el encargado de iniciar la transmisión de la información, para colocar la dirección del nodo, es necesario modificar una de las variables de entorno que se encuentra en el archivo wireless_api.c , esta variable de entorno es remplazada por la variable llamada addpropia, la cual sobrescribe el valor de la dirección del nodo en el archivo de configuración y toma el valor de addpropia en su lugar, dicho valor debe ser convertido a hexadecimal, para lo cual se utiliza la función propia del nodo CCPU_ENDIAN_TO_LE16, esto se realiza de esta manera dado que el nodo entiende las direcciones únicamente en hexadecimal, posteriormente el nodo realiza la transmisión de la información con la ayuda de la función propia del nodo transmit_sample_frame, que para la realización de este trabajo también fue modificada en el archivo de configuración wireless_api.c, donde se añadió un campo adicional en la función que toma el valor de la dirección del nodo destino y se incluye al final de esta, así podemos direccionar la trama, el código del primer nodo se presenta a continuación.

```
/*------------------------------CÓDIGO DEL NODO 1------------------------------*/

#include "usr_wireless.h"
#include "wireless_config.h"

void usr_wireless_app_task(void)
{
	uint16_t addpropia; //Variable de dirección origen declarada
	addpropia=1; //Dirección de origen del nodo transmisor
	addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
	tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la
dirección

	#ifdef TRANSMITTER_ENABLED //Etapa de transmisión
		transmit_sample_frame((uint8_t*)";:", 2, 0x002); //Función modificada
de transmisón
		delay_ms(10000); //Se envía cada 10 segundos
	#endif
}

void usr_frame_received_cb(frame_info_t *frame)
{

}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

2.3.3.2 Código del segundo nodo

El segundo nodo de la red también maneja una dirección con la misma función explicada para el nodo uno, después, cuando le llega la trama, en la función void usr_frame_received este nodo se encarga de procesar el mensaje enviado por el primer nodo, para esto se crean dos variables en las cuales se almacenarán los dos primeros valores del payload enviados en la trama del primer nodo, si estos valores coinciden con el token cuyo valor es el de punto y coma (;) que corresponde 0x3B y el de dos puntos (:) que corresponde a 0x3A seguidos (;:), entra en juego el primer bucle if del código, en este bucle se emplea una variable global auxiliar declarada previamente llamada habilitarTX, la cual cambia su valor de cero a uno, al hacer este cambio, el código entra en la función usr_wireless_app_task, donde la variable habilitarTX cambia su valor a dos y procede a enviar una trama de datos con destino al nodo siguiente (pues se verificó que el token si llegó al nodo), esta acción tuvo que realizarse de esta manera con la finalidad de poder enviar dos tramas simultáneamente, pues debido a limitaciones de la arquitectura del nodo transmisor y el lenguaje de codificación no se puede realizar simplemente apilando dos funciones de transmisión, ahora la variable vuelve a tomar el valor de cero (valor inicialmente declarado) y procede a transmitir el token, al declarar nuevamente la variable habilitarTX en cero, nos aseguramos de que el proceso de transmisión de las tramas no solo suceda una vez, finalmente se coloca un delay para que esto suceda cada medio segundo.

Por otra parte, en caso de que el primer nodo no hubiese enviado el token en su transmisión, el nodo envía una trama en broadcast avisando que no recibió el token del nodo anterior, después procede a salir del bucle y entrar en estado de espera, donde esperará una retransmisión del nodo.

```
/*------------------------------CÓDIGO DEL NODO 2------------------------------*/

#include "usr_wireless.h"
#include "wireless_config.h"
int habilitarTX=0; //Variable que ayuda a la transmisión simultánea de tramas

void usr_wireless_app_task(void)
{
	uint16_t addpropia; //Variable de dirección origen declarada
	addpropia=2; //Direccion de origen del nodo transmisor
	addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
	tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la
direción
	if (habilitarTX==1) //La variable toma el valor de 1 al verificar el token en
el Payload
	{
		habilitarTX=2; //La variable toma el valor de 2 y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp1", 8, 0x003); //Funcion
modificada de transmisión
	}
```

```
	else if (habilitarTX==2) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=0; //La variable toma el valor de 0 lo cual hace que esta
acción no se repita
		transmit_sample_frame((uint8_t*)";:", 2, 0x003); //Funcion modificada
de transmisión
	}
	delay_ms (500); //Se envía cada 0.5 segundos
}

void usr_frame_received_cb(frame_info_t *frame) //Cuando el nodo recibe una trama
ejecuta esta función
{
	memset(&trama_recibida,0,sizeof(trama_recibida));//Encera el vector
	memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));//Copia la
informacion del buffer al destino
	bmm_buffer_free(frame->buffer_header);//Limpia los datos del buffer, evita
superposicion.

	uint8_t token1; //Primera variable para el primer caracter del payload
	uint8_t token2; //Segunda variable para el segundo caracter del payload
	token1=trama_recibida.payloadd[0]; //Almaceno el primer caracter del payload
en "token1"
	token2=trama_recibida.payloadd[1]; //Almaceno el segundo caracter del payload
en "token2"

	//Siendo el token el caracter ";:" al principio del payload
	if (token1==0x3B && token2==0x3A) //El caracter ; en hexadecimal corresponde
a 0x3B y : corresponde a 0x3A
	{
		//Si cumple la condicion, el nodo transmite
		habilitarTX=1;
	}
	else
	{
		//Si no cumple la condicion, el nodo sale del bucle y entra en estado
de espera
		transmit_sample_frame((uint8_t*)"TKN no recibido", 11, 0xFFFE);
//Funcion modificada donde el ultimo valor es la direccion destino
		goto salir;
	}
	salir:
	delay_ms(0);
}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

2.3.3.3 Código del tercer nodo

El código del tercer nodo de la red posee las funciones de las direcciones idénticas a las de los dos nodos anteriores, sin embargo, existe un cambio considerable en este código ya que ahora se realiza una comparación de la trama de datos además de la del token, se considera que toda trama de datos empieza con los caracteres D mayúscula y A mayúscula, los cuales nos avisan que efectivamente se trata de una trama de este tipo, el

valor de la D mayúscula en hexadecimal corresponde a 0x44 y la A mayúscula corresponde a 0x41, la primera acción a realizar es declarar otras dos variables en donde se almacena el primer y segundo carácter de la trama de datos y después se procede a realizar la comparación, en caso de tratarse de una trama de datos existe un contador el cual va reduciendo su valor según el número de datos que se hayan transferido del nodo anterior y declarados en la variable cantDatos, es importante recalcar que este valor no considera a la trama del token como una trama de datos, pues no es así, una vez que el contador haya llegado a cero el nodo habrá procesado todas las tramas de datos entrantes del nodo anterior y el valor de cantDatos habrá tomado el valor de 0, ahora se realiza otra comparación en donde se verifica que la trama del token haya llegado (comparando sus respectivos valores 0x3B y 0X3A) y también que la variable cantDatos sea cero, si se cumple con esta condición el nodo emplea la variable auxiliar habilitarTX que justamente se encarga de seguir con la transmisión de las tramas, enviando al nodo siguiente la trama de información del nodo anterior y la trama de información que el nodo actual tenga para enviar.

```
/*------------------------------CÓDIGO DEL NODO 3------------------------------*/

#include "usr_wireless.h"
#include "wireless_config.h"
#include "periodic_timer.h"
int habilitarTX=0;
int cantDatos=1; //Datos a enviar

void usr_wireless_app_task(void)
{
	uint16_t addpropia; //Variable de dirección origen declarada
	addpropia=3; //Dirección de origen del nodo transmisor
	addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
	tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la
direccion

	if (habilitarTX==1) //La variable flag toma el valor de 1 al verificar el
token en el Payload
	{
		habilitarTX=2; //La variable toma el valor de 2 y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp1", 8, 0x004); //Funcion
modificada de transmisión
	}
	else if (habilitarTX==2) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=3; //La variable toma el valor de 3 y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp2", 8, 0x004); //Funcion
modificada de transmisión
	}
	else if (habilitarTX==3) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=0; //La variable toma el valor de 0 lo cual hace que esta
acción no se repita
```

```
            transmit_sample_frame((uint8_t*)";:", 2, 0x004); //Funcion modificada
de transmisión
            cantDatos=1; //Vuelve a tomar el valor global para repetir la acción
      }
      delay_ms (500); //Se envía cada 0.5 segundos
}

void usr_frame_received_cb(frame_info_t *frame) //Cuando el nodo recibe una trama
ejecuta esta función
{

      memset(&trama_recibida,0,sizeof(trama_recibida));//Encera el vector
      memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));//Copia la
informacion del buffer al destino
      bmm_buffer_free(frame->buffer_header);//Limpia los datos del buffer, evita
superposicion.

      uint8_t token1; //Primera variable para el primer caracter del payload
      uint8_t token2; //Segunda variable para el segundo caracter del payload
      uint8_t datos1; //Primera variable para el primer caracter del payload de la
trama de datos
      uint8_t datos2; //Segunda variable para el segundo caracter del payload de la
trama de datos
      token1=trama_recibida.payloadd[0]; //Almaceno el primer caracter del payload
del token
      token2=trama_recibida.payloadd[1]; //Almaceno el segundo caracter del payload
del token
      datos1=trama_recibida.payloadd[0]; //Almaceno el primer caracter del payload
de la trama de datos
      datos2=trama_recibida.payloadd[1]; //Almaceno el segundo caracter del payload
de la trama de datos

      //Considerando que una trama de datos empieza con los caracteres DA ambos en
mayúsculas
      if (datos1==0x44 && datos2==0x41) //El caracter D en hexadecimal corresponde
a 0x44 y A corresponde a 0x41
      {
            cantDatos=cantDatos-1; //Al procesar la trama de datos, la variable
disminuye y llega a cero
      }
      //Siendo el token el caracter ";:" al principio del payload y la condición de
no tener más datos
      if (token1==0x3B && token2==0x3A && cantDatos==0) //El caracter ; en
hexadecimal corresponde a 0x3B y : corresponde a 0x3A
      {
            //Si cumple la condicion, el nodo transmite
            habilitarTX=1;
      }
delay_ms(0);
}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

2.3.3.4 Código del cuarto nodo

El código del cuarto nodo de la red es muy similar al código del tercer nodo, por lo que no se ahondará más en este código, la diferencia a recalcar radica en que en este nodo se

consideran más tramas de datos tanto recibidas como para ser enviadas, teniendo en cuenta que el nodo anterior envía las tramas acumuladas de los anteriores.

```
/*------------------------------CÓDIGO DEL NODO 4------------------------------*/

#include "usr_wireless.h"
#include "wireless_config.h"
#include "periodic_timer.h"
int habilitarTX=0;
int cantDatos=2; //Datos a enviar

void usr_wireless_app_task(void)
{
uint16_t addpropia; //Variable de dirección origen declarada
addpropia=4; //Dirección de origen del nodo transmisor
addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la direccion

	if (habilitarTX==1) //La variable flag toma el valor de 1 al verificar el
token en el Payload
	{
		habilitarTX=2; //La variable toma el valor de 2 y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp1", 8, 0x005); //Funcion
modificada de transmisión
	}
	else if (habilitarTX==2) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=3; //La variable toma el valor de 3  y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp2", 8, 0x005); //Funcion
modificada de transmisión
	}
	else if (habilitarTX==3) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=4; //La variable toma el valor de 4  y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp3", 8, 0x005); //Funcion
modificada de transmisión
	}
	else if (habilitarTX==4) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=0; //La variable toma el valor de 0 lo cual hace que esta
acción no se repita
		transmit_sample_frame((uint8_t*)";:", 2, 0x005); //Funcion modificada
de transmisión
		cantDatos=2; //Vuelve a tomar el valor global para repetir la acción
	}
delay_ms (500); //Se envía cada 0.5 segundos
}

void usr_frame_received_cb(frame_info_t *frame) //Cuando el nodo recibe una trama
ejecuta esta función
{

		memset(&trama_recibida,0,sizeof(trama_recibida));//Encera el vector
		memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));//Copia la
informacion del buffer al destino
		bmm_buffer_free(frame->buffer_header);//Limpia los datos del buffer,
evita superposicion.
```

```
            uint8_t token1; //Primera variable para el primer caracter del payload
del token
            uint8_t token2; //Segunda variable para el segundo caracter del
payload del token
            uint8_t datos1; //Primera variable para el primer caracter del payload
de la trama de datos
            uint8_t datos2; //Segunda variable para el segundo caracter del
payload de la trama de datos
            token1=trama_recibida.payloadd[0]; //Almaceno el primer caracter del
payload en "token1"
            token2=trama_recibida.payloadd[1]; //Almaceno el segundo caracter del
payload en "token2"
            datos1=trama_recibida.payloadd[0]; //Almaceno el primer caracter del
payload de la trama de datos
            datos2=trama_recibida.payloadd[1]; //Almaceno el segundo caracter del
payload de la trama de datos

            //Considerando que una trama de datos empieza con los caracteres DA
ambos en mayúsculas
            if (datos1==0x44 && datos2==0x41) //El caracter D en hexadecimal
corresponde a 0x44 y A corresponde a 0x41
            {
            cantDatos=cantDatos-1; //Al procesar la trama de datos, la variable
disminuye y llega a cero
            }
            //Siendo el token el caracter ";:" al principio del payload y la
condición de no tener más datos
            if (token1==0x3B && token2==0x3A && cantDatos==0) //El caracter ; en
hexadecimal corresponde a 0x3B y : corresponde a 0x3A
            {
            //Si cumple la condicion, el nodo transmite
            habilitarTX=1;
            }

delay_ms(0);
}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

2.3.3.5 Código del quinto nodo

El código del quinto nodo de la red es muy similar al código del cuarto y tercer nodo, por lo que no se ahondará más en este código, la diferencia a recalcar radica nuevamente en que en este nodo se consideran más tramas de datos tanto recibidas como para ser enviadas, teniendo en cuenta que el nodo anterior envía las tramas acumuladas de los anteriores y además este nodo se encarga enviar un mensaje que indica que la transmisión ha finalizado.

```
/*------------------------------CÓDIGO DEL NODO 5------------------------------*/

#include "usr_wireless.h"
#include "wireless_config.h"
#include "periodic_timer.h"
```

```
int habilitarTX=0;
int cantDatos=3; //Datos a enviar

void usr_wireless_app_task(void)
{
uint16_t addpropia; //Variable de dirección origen declarada
addpropia=5; //Dirección de origen del nodo transmisor
addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la direccion

	if (habilitarTX==1) //La variable flag toma el valor de 1 al verificar el
token en el Payload
	{
		habilitarTX=2; //La variable toma el valor de 2 y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp1", 8, 0x001); //Funcion
modificada de transmisión
	}
	else if (habilitarTX==2) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=3; //La variable toma el valor de 3 y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp2", 8, 0x001); //Funcion
modificada de transmisión
	}
	else if (habilitarTX==3) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=4; //La variable toma el valor de 4 y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp3", 8, 0x001); //Funcion
modificada de transmisión
		cantDatos=1;
	}
	else if (habilitarTX==4) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=5; //La variable toma el valor de 5 y transmite una trama
		transmit_sample_frame((uint8_t*)"DA temp4", 8, 0x001); //Funcion
modificada de transmisión
		cantDatos=1;
	}
	else if (habilitarTX==5) //La variable entra en otro condicional para enviar
otra trama
	{
		habilitarTX=0; //La variable toma el valor de 0 lo cual hace que esta
acción no se repita
		transmit_sample_frame((uint8_t*)"TX FINALIZADA", 13, 0x001); //Funcion
modificada de transmisión
		cantDatos=3; //Vuelve a tomar el valor global para repetir la acción
	}
delay_ms (500); //Se envía cada 0.5 segundos
}

void usr_frame_received_cb(frame_info_t *frame) //Cuando el nodo recibe una trama
ejecuta esta función
{

		memset(&trama_recibida,0,sizeof(trama_recibida));//Encera el vector
		memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));//Copia la
informacion del buffer al destino
		bmm_buffer_free(frame->buffer_header);//Limpia los datos del buffer,
evita superposicion.
```

```
            uint8_t token1; //Primera variable para el primer caracter del payload
del token
            uint8_t token2; //Segunda variable para el segundo caracter del
payload del token
            uint8_t datos1; //Primera variable para el primer caracter del payload
de la trama de datos
            uint8_t datos2; //Segunda variable para el segundo caracter del
payload de la trama de datos
            token1=trama_recibida.payloadd[0]; //Almaceno el primer caracter del
payload en "token1"
            token2=trama_recibida.payloadd[1]; //Almaceno el segundo caracter del
payload en "token2"
            datos1=trama_recibida.payloadd[0]; //Almaceno el primer caracter del
payload de la trama de datos
            datos2=trama_recibida.payloadd[1]; //Almaceno el segundo caracter del
payload de la trama de datos

            //Considerando que una trama de datos empieza con los caracteres DA
ambos en mayúsculas
            if (datos1==0x44 && datos2==0x41)  //El caracter D en hexadecimal
corresponde a 0x44 y A corresponde a 0x41
            {
            cantDatos=cantDatos-1; //Al procesar la trama de datos, la variable
disminuye y llega a cero
            }
            //Siendo el token el caracter ";:" al principio del payload y la
condición de no tener más datos
            if (token1==0x3B && token2==0x3A && cantDatos==0) //El caracter ; en
hexadecimal corresponde a 0x3B y : corresponde a 0x3A
            {
            //Si cumple la condicion, el nodo transmite
            habilitarTX=1;
            }

delay_ms(0);
}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

3.1.1 IMPLEMENTACIÓN DEL PROTOTIPO DE ENTORNO DE PRUEBAS

Los componentes necesarios para el entorno de pruebas se presentan en la tabla 3.1.

Tabla 3.1 Componentes del entorno de pruebas

Componentes	Cantidad
Computador	1
Nodos Atmega256RFR2	5
Baterías AAA	10
Sniffer	1
Interfaz JTAG	1

El prototipo de la red implementado en topología lineal se presenta en la figura 3.1, como se mencionó a lo largo de este trabajo de titulación, se contemplan un total de 5 nodos transceptores, además en la figura 3.2 se presenta un escenario real con los nodos implementados.

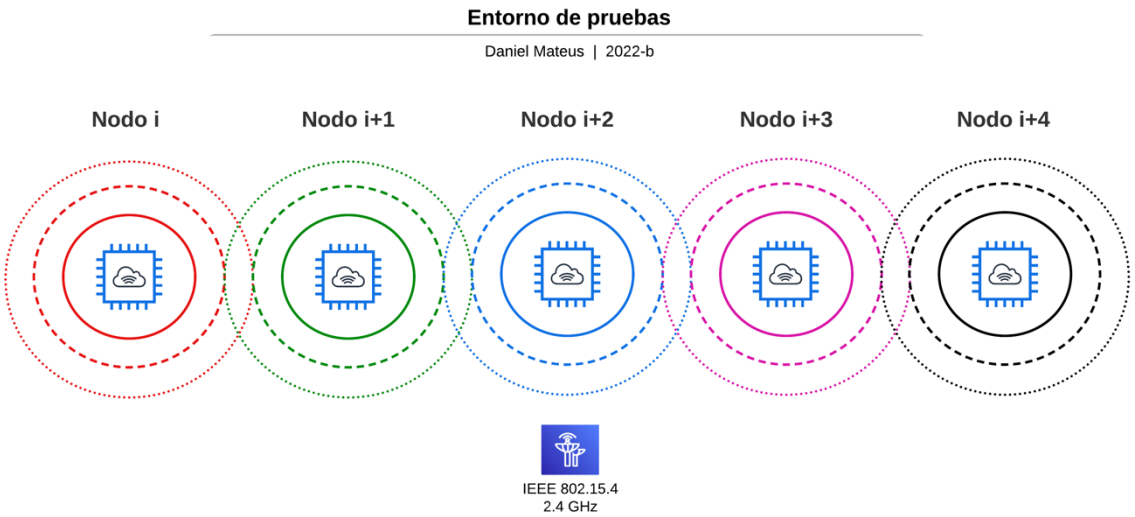


Figura 3.1 Entorno de pruebas

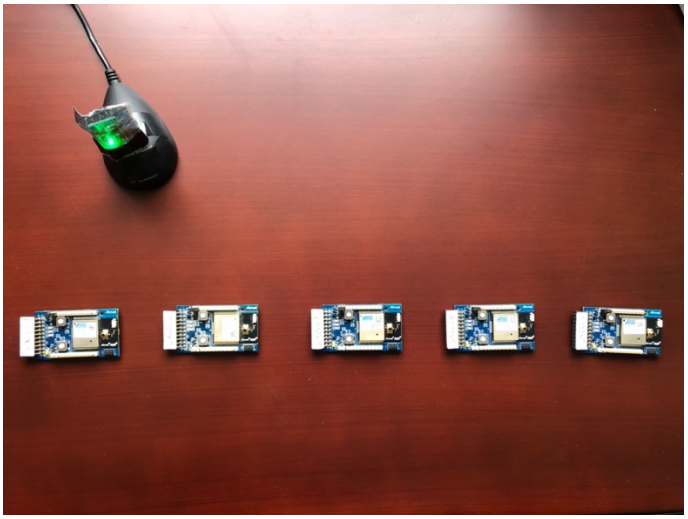


Figura 3.2 Escenario real de pruebas

3.1.1.1 Proceso de quemado de los nodos

Para grabar la información en el nodo, primero se debe compilar el código y verificar que este no posea ningún error, esto se logra con la ayuda de la tecla F7 que inicia el proceso "build solution", como se puede observar en la figura 3.3 en donde se obtiene un resultado de "build succeded" que nos indica que el código trabaja de manera correcta.

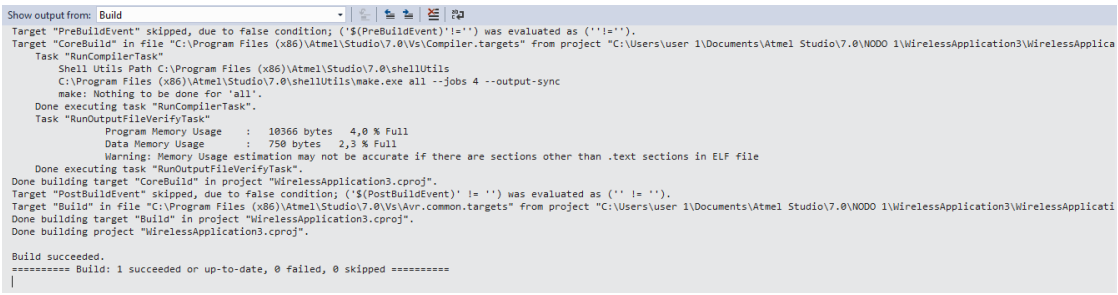


Figura 3.3 Salida de "build solution"

Posteriormente, con la sucesión de comandos ctrl+Shift+P, se accede al programador de dispositivos, como se puede observar en la figura 3.4, en esta ventana se debe seleccionar la interfaz JTAG con la cual se quemará el nodo y que previamente fue conectada al puerto USB 3.0 del computador y dar clic en "Apply", ahora se puede leer la firma del dispositivo y el voltaje de las baterías que alimentan al mismo, donde en caso de contar con un voltaje menor a 1.8, se recomienda un cambio de baterías.

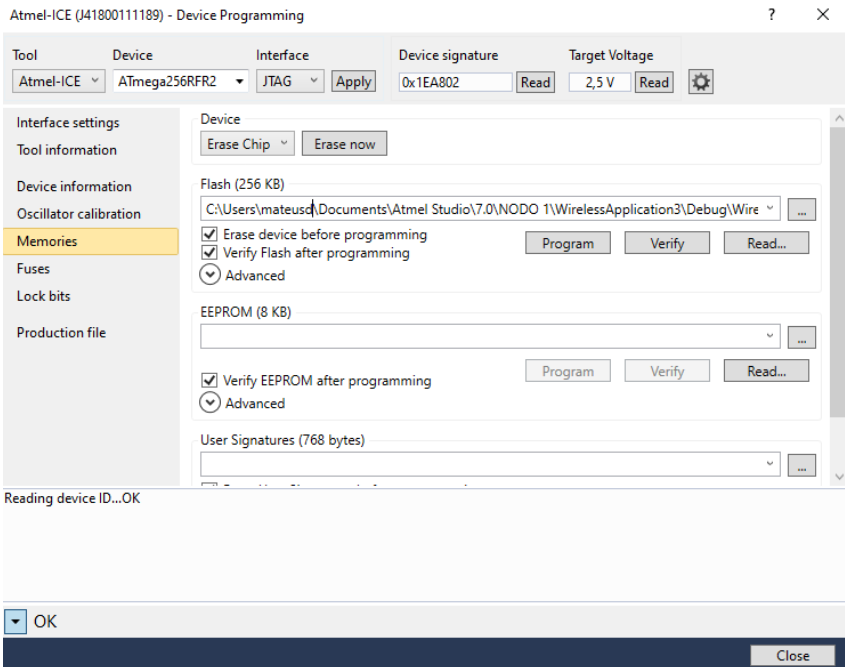


Figura 3.4 Ventana “Device Programming”

3.1.2 PRUEBAS REALIZADAS

3.1.2.1 Empleando únicamente el algoritmo diseñado

Para comprobar que nuestro algoritmo funciona de una manera adecuada, debemos quemar los 5 nodos de la red con sus respectivos códigos y energizarlos, al tener en funcionamiento la red, podemos empezar a capturar los paquetes de esta en el canal 20 de frecuencia (0x14 - 2450 MHz) con la ayuda del sniffer como se puede observar en la figura 3.5.

Figura 3.5 Sniffer capturando en el canal 20 de frecuencia

Los resultados del algoritmo con el campo del payload en texto se presentan en la figura 3.6 y con el payload en hexadecimal en la figura 3.7, en ambas figuras se hace un resalte por colores de las tramas que envía cada nodo, con el fin de facilitar la comprensión.

P.nbr.	Time (ms)	Length	Frame control field	Dest. Address	Source Address	MAC payload
RX 1	+0 =0	13	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0002	0x0001	; :
RX 2	+642 =642	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0003	0x0002	DA t emp1
RX 3	+831 =1473	13	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0003	0x0002	; :
RX 4	+501 =1974	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0004	0x0003	DA t emp1
RX 5	+838 =2813	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0004	0x0003	DA t emp2
RX 6	+838 =3651	13	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0004	0x0003	; :
RX 7	+50 =3701	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0005	0x0004	DA t emp1
RX 8	+828 =4530	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0005	0x0004	DA t emp2
RX 9	+828 =5359	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0005	0x0004	DA t emp3
RX 10	+828 =6187	13	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0005	0x0004	; :
RX 11	+257 =6445	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0001	0x0005	DA t emp1
RX 12	+838 =7284	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0001	0x0005	DA t emp2
RX 13	+838 =8123	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0001	0x0005	DA t emp3
RX 14	+839 =8962	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0001	0x0005	DA t emp4
RX 15	+839 =9802	24	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0001	0x0005	TX FINA LIZADA

Figura 3.6 Captura de tramas con el payload en texto

Si nos fijamos en el campo del payload, podemos apreciar más detalladamente que, el primer nodo transmite el token (;:) que habilita la transmisión en la red, el segundo lo recibe y envía primero su trama de datos, posteriormente envía el token al siguiente nodo, el tercer nodo entonces recibe la trama de datos del nodo dos y la envía junto con su propia trama de datos, situación que se repite para el nodo cuatro y para el nodo cinco, todo ocurre de acuerdo a lo establecido en el diagrama de flujo de tiempo de la figura 2.13.

En la figura 3.7 podemos observar que ahora el payload se encuentra en formato hexadecimal, y podemos comprobar que los valores del token (;:) y de las tramas de datos corresponde a los valores planteados durante los condicionales en la codificación, siendo 0x3B y 0x3A para el token y 0x44 y 0x41 para la trama de datos.

Figura 3.7 Captura de tramas con el payload en hexadecimal

Como se puede observar, se envían en total 15 tramas, si analizamos en detalle la transmisión tenemos la información presentada en la tabla 3.2.

Tabla 3.2 Resultados de la transmisión

	# Trama	Payload	Dir Origen	Dir Destino	Tiempo (ms)
Nodo 1	1	;:	0x0001	0x0002	0
Nodo 2	2	DA temp1	0x0002	0x0003	642
	3	;:			831
Nodo 3	4	DA temp1	0x0003	0x0004	501
	5	DA temp2			838
	6	;:			838
Nodo 4	7	DA temp1	0X0004	0X0005	50
	8	DA temp2			828
	9	DA temp3			828
	10	;:			828
Nodo 5	11	DA temp1	0x0005	0x0001	257
	12	DA temp2			838
	13	DA temp3			838
	14	DA temp4			839
	15	TX FINALIZADA			839
				TOTAL	9802

En la figura 3.8 se puede observar el diagrama de flujo de tiempo en el cual se va marcando el tiempo en el que se envía cada trama hasta llegar a los 9802 ms.

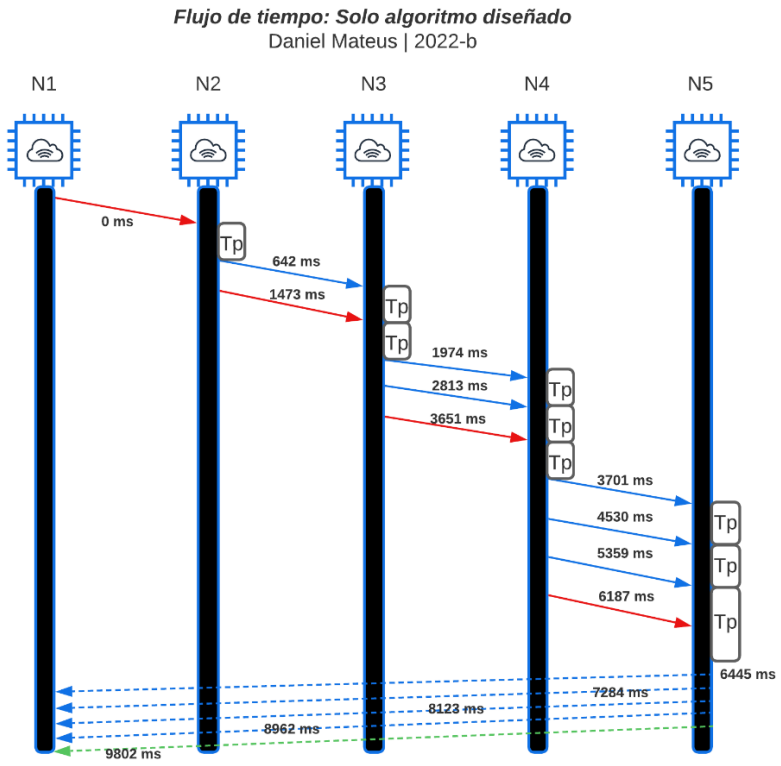
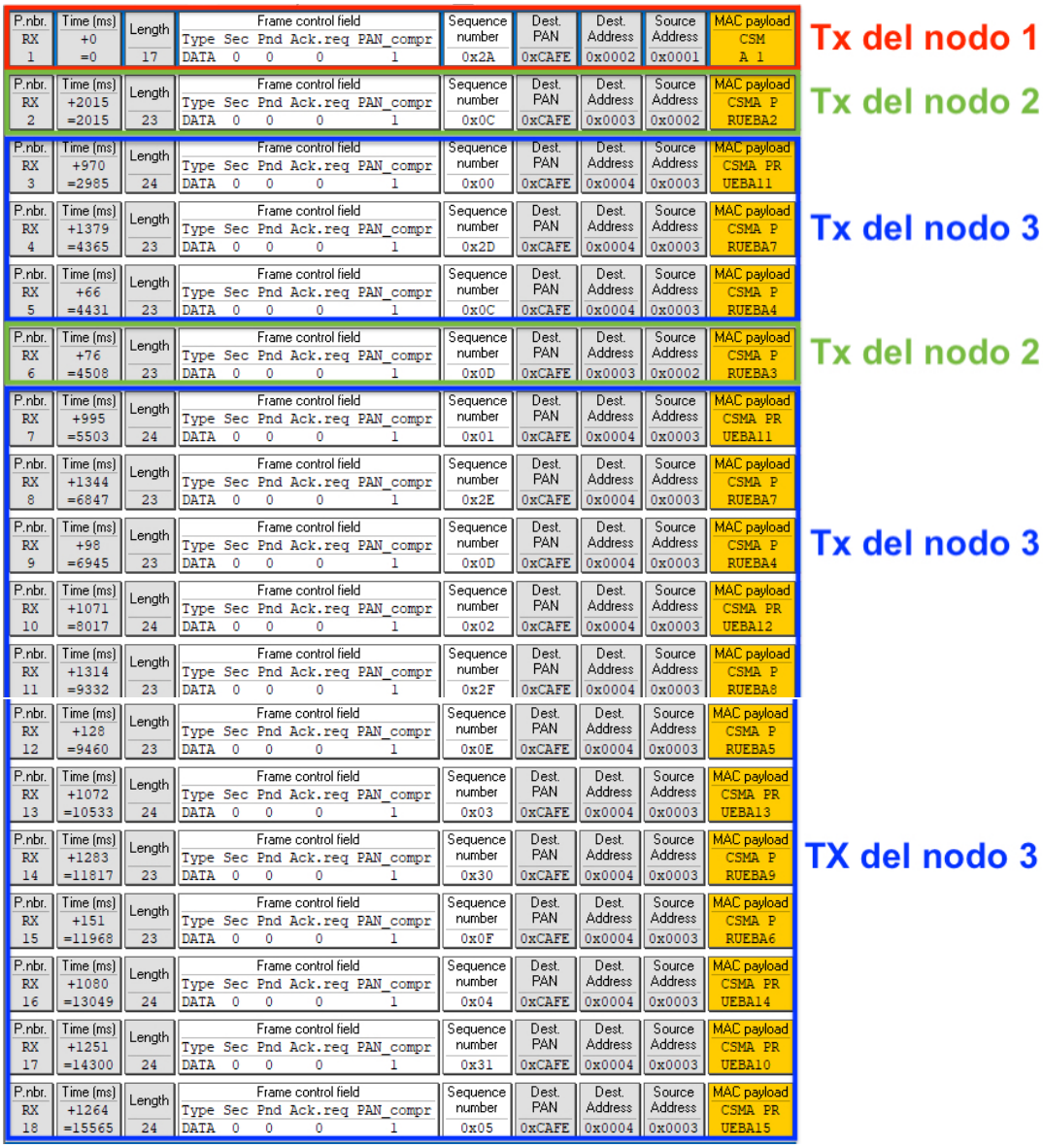


Figura 3.8 Flujo de tiempo empleando el algoritmo diseñado

3.1.2.2 Empleando únicamente CSMA

Ahora se realizará una prueba empleando el protocolo CSMA, el cual puede ser configurado en el asistente al principio de la creación de un proyecto o manualmente en el archivo de configuración wireless_config.h, se simulará un entorno similar al planteado en la sección 3.1.2.1, en donde la transmisión sea en orden y secuencial, los resultados se presentan en la figura 3.9.



P.nbr.	Time (ms)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source Address	MAC payload
RX	+0		Type Sec Pnd Ack.req PAN_compr					CSM
1	=0	17	DATA 0 0 0 1	0x2A	0xCAFE	0x0002	0x0001	A 1
RX	+2015							CSMA P
2	=2015	23	DATA 0 0 0 1	0x0C	0xCAFE	0x0003	0x0002	RUEBA2
RX	+970							CSMA PR
3	=2985	24	DATA 0 0 0 1	0x00	0xCAFE	0x0004	0x0003	UEBA11
RX	+1379							CSMA P
4	=4365	23	DATA 0 0 0 1	0x2D	0xCAFE	0x0004	0x0003	RUEBA7
RX	+66							CSMA P
5	=4431	23	DATA 0 0 0 1	0x0C	0xCAFE	0x0004	0x0003	RUEBA4
RX	+76							CSMA P
6	=4508	23	DATA 0 0 0 1	0x0D	0xCAFE	0x0003	0x0002	RUEBA3
RX	+995							CSMA PR
7	=5503	24	DATA 0 0 0 1	0x01	0xCAFE	0x0004	0x0003	UEBA11
RX	+1344							CSMA P
8	=6847	23	DATA 0 0 0 1	0x2E	0xCAFE	0x0004	0x0003	RUEBA7
RX	+98							CSMA P
9	=6945	23	DATA 0 0 0 1	0x0D	0xCAFE	0x0004	0x0003	RUEBA4
RX	+1071							CSMA PR
10	=8017	24	DATA 0 0 0 1	0x02	0xCAFE	0x0004	0x0003	UEBA12
RX	+1314							CSMA P
11	=9332	23	DATA 0 0 0 1	0x2F	0xCAFE	0x0004	0x0003	RUEBA8
RX	+128							CSMA P
12	=9460	23	DATA 0 0 0 1	0x0E	0xCAFE	0x0004	0x0003	RUEBA5
RX	+1072							CSMA PR
13	=10533	24	DATA 0 0 0 1	0x03	0xCAFE	0x0004	0x0003	UEBA13
RX	+1283							CSMA P
14	=11817	23	DATA 0 0 0 1	0x30	0xCAFE	0x0004	0x0003	RUEBA9
RX	+151							CSMA P
15	=11968	23	DATA 0 0 0 1	0x0F	0xCAFE	0x0004	0x0003	RUEBA6
RX	+1080							CSMA PR
16	=13049	24	DATA 0 0 0 1	0x04	0xCAFE	0x0004	0x0003	UEBA14
RX	+1251							CSMA PR
17	=14300	24	DATA 0 0 0 1	0x31	0xCAFE	0x0004	0x0003	UEBA10
RX	+1264							CSMA PR
18	=15565	24	DATA 0 0 0 1	0x05	0xCAFE	0x0004	0x0003	UEBA15

Figura 3.9 Captura de tramas con el payload en texto

Como se puede observar en la figura 3.9, existen varios problemas al momento de realizar la transmisión sin un token que controle el envío de las tramas, primero hay que recalcar que cada nodo tenía contemplado transmitir al menos una trama, cosa que no sucede puesto que en algún momento la transmisión colapsa y el sniffer detecta que las tramas

“CSMA PRUEBA 4” hasta “CSMA PRUEBA 15” son enviadas por el nodo 3, cuando la codificación de los nodos es distinta (VER ANEXO 1), otro gran problema que presenta la transmisión es el orden de las tramas, la repetición de tramas y por último, se tiene que el tiempo total en enviar las 15 tramas es mucho mayor que usando el algoritmo diseñado en este trabajo, en la tabla 3.3 se presenta en detalle la transmisión.

Tabla 3.3 Resultados de la transmisión

	# Trama	Payload	Dir Origen	Dir Destino	Tiempo (ms)
Nodo 1	1	CSMA 1	0x0001	0x0002	0
Nodo 2	2	CSMA PRUEBA2	0x0002	0x0003	2015
	3	CSMA PRUEBA3			76
Nodo 3	4	CSMA PRUEBA11	0x0003	0x0004	970
	5	CSMA PRUEBA7			1379
	6	CSMA PRUEBA4			66
	7	CSMA PRUEBA11			995
	8	CSMA PRUEBA7			1344
	9	CSMA PRUEBA4			98
	10	CSMA PRUEBA12			1071
	11	CSMA PRUEBA8			1314
	12	CSMA PRUEBA5			128
	13	CSMA PRUEBA13			1072
	14	CSMA PRUEBA9			1283
	15	CSMA PRUEBA6			151
	16	CSMA PRUEBA14			1080
	17	CSMA PRUEBA10			1251
	18	CSMA PRUEBA15			1264
				TOTAL	15565

3.1.2.3 Empleando el algoritmo diseñado y el protocolo CSMA juntos

Ahora como siguiente prueba, se procederá a trabajar tanto con el algoritmo diseñado como con el protocolo CSMA juntos, para ello, debemos activar CSMA de manera manual en cada uno de los archivos wireless_config.h del código de cada nodo, pasando de NO_CSMA_WITH_IFS como se muestra en la figura 3.10 al modo CSMA_UNSLOTTED como se muestra en la figura 3.11.

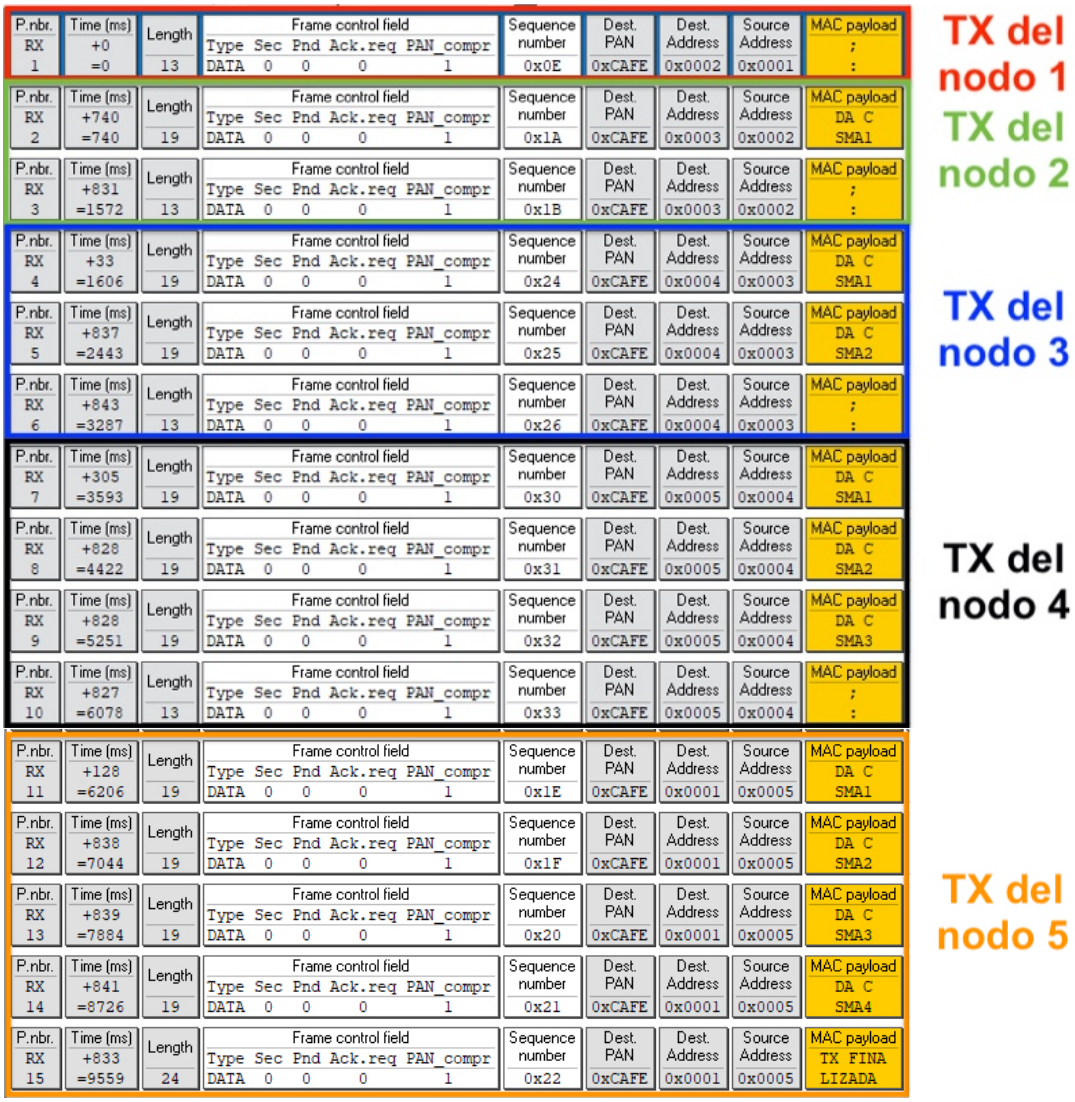
```
#define DST_PAN_ID      0xCAFE
#define ACK_REQ         0
#define FRAME_RETRY     0
#define CSMA_MODE       NO_CSMA_WITH_IFS
```

Figura 3.10 Modo sin CSMA

```
#define DST_PAN_ID      0xCAFE
#define ACK_REQ         0
#define FRAME_RETRY     0
#define CSMA_MODE       CSMA_UNSLOTTED
```

Figura 3.11 Modo con CSMA

Los resultados de la transmisión se presentan en la figura 3.12.



P.nbr.	Time (ms)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source Address	MAC payload	
RX 1	+0 =0	13	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0E	0xCAFE	0x0002	0x0001	; :	TX del nodo 1
RX 2	+740 =740	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x1A	0xCAFE	0x0003	0x0002	DA C SMA1	TX del nodo 2
RX 3	+831 =1572	13	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x1B	0xCAFE	0x0003	0x0002	; :	
RX 4	+33 =1606	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x24	0xCAFE	0x0004	0x0003	DA C SMA1	TX del nodo 3
RX 5	+837 =2443	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x25	0xCAFE	0x0004	0x0003	DA C SMA2	
RX 6	+843 =3287	13	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x26	0xCAFE	0x0004	0x0003	; :	
RX 7	+305 =3593	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x30	0xCAFE	0x0005	0x0004	DA C SMA1	TX del nodo 4
RX 8	+828 =4422	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x31	0xCAFE	0x0005	0x0004	DA C SMA2	
RX 9	+828 =5251	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x32	0xCAFE	0x0005	0x0004	DA C SMA3	
RX 10	+827 =6078	13	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x33	0xCAFE	0x0005	0x0004	; :	
RX 11	+128 =6206	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x1E	0xCAFE	0x0001	0x0005	DA C SMA1	TX del nodo 5
RX 12	+838 =7044	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x1F	0xCAFE	0x0001	0x0005	DA C SMA2	
RX 13	+839 =7884	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x20	0xCAFE	0x0001	0x0005	DA C SMA3	
RX 14	+841 =8726	19	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x21	0xCAFE	0x0001	0x0005	DA C SMA4	
RX 15	+833 =9559	24	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x22	0xCAFE	0x0001	0x0005	TX FINA LIZADA	

Figura 3.12 Captura de tramas con el payload en texto

Como se puede observar, al trabajar de manera conjunta con el algoritmo y el protocolo CSMA se tiene una ventaja en la reducción de tiempos muy pequeña, exactamente de 243 ms, lo cual se puede ver como un gran beneficio dependiendo del uso de la red.

Si analizamos en detalle la transmisión tenemos la información presentada en la tabla 3.4.

Tabla 3.4 Resultados de la transmisión

	# Trama	Payload	Dir Origen	Dir Destino	Tiempo (ms)
Nodo 1	1	;:	0x0001	0x0002	0
Nodo 2	2	DA CSMA1	0x0002	0x0003	740
	3	;:			831
Nodo 3	4	DA CSMA1	0x0003	0x0004	33
	5	DA CSMA2			837
	6	;:			843
Nodo 4	7	DA CSMA 1	0X0004	0X0005	305
	8	DA CSMA 2			828
	9	DA CSMA 3			828
	10	;:			827
Nodo 5	11	DA CSMA 1	0x0005	0x0001	128
	12	DA CSMA 2			838
	13	DA CSMA 3			839
	14	DA CSMA 4			841
	15	TX FINALIZADA			833
				TOTAL	9559

En la figura 3.13 se puede observar el diagrama de flujo de tiempo en el cual se va marcando el tiempo en el que se envía cada trama hasta llegar a los 9559 ms.

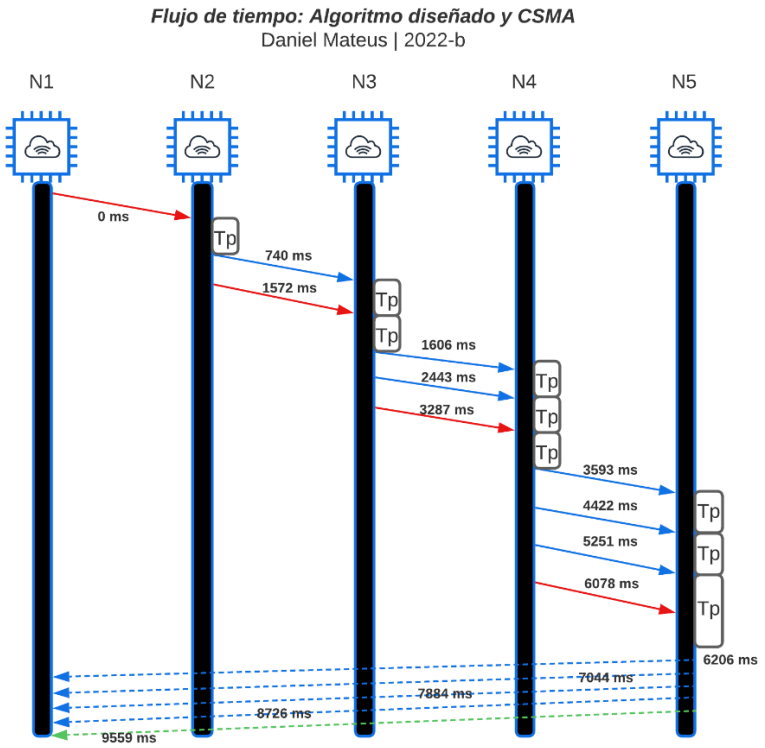


Figura 3.13 Captura de tramas con el payload en texto

En la tabla 3.5 se presenta un resumen de los resultados obtenidos en las pruebas realizadas.

Tabla 3.5 Resumen de resultados

	Solo algoritmo	**Solo CSMA**	**Algoritmo y CSMA**
Tiempo (ms)	9802	15565	9559
Direcciones origen y destino	Correctas	Erroneas	Correctas
Superposición de tramas	No	Si	No

3.1.2.4 Simulando un fallo en la transmisión

Para este escenario, vamos a hacer que el algoritmo falle a propósito, para esto, el token no se enviará del nodo uno al nodo dos, se enviará una trama cualquiera como se puede observar en la figura 3.14 y 3.15.

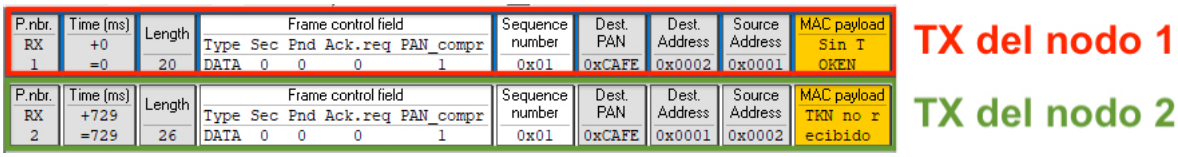


Figura 3.14 Captura de tramas con el payload en texto

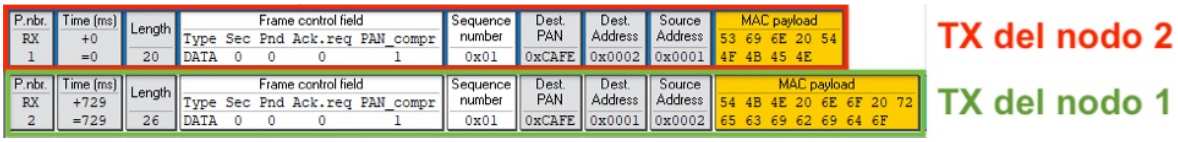


Figura 3.15 Captura de tramas con el payload en hexadecimal

Como podemos observar al no enviarse el envío del token, el nodo dos verifica en el condicional que no se trata de los caracteres 0X3B y 0X3A (token) por lo que no puede continuar con la transmisión de la información y envía un mensaje de que el token no fue recibido de vuelta al nodo transmisor, después procede a volver al estado de espera hasta que el mismo sea enviado correctamente.

3.1.3 ANÁLISIS DE RESULTADOS

Una vez concluidas las pruebas, se pueden analizar los siguientes resultados.

El algoritmo diseñado y planteado en este trabajo de titulación cumple con brindar un acceso óptimo al canal, permitiendo el envío ordenado, correcto y sin pérdida de ninguna

de las tramas en una red con topología lineal, lo cual nos garantiza una transmisión eficaz y eficiente, pues el número de retransmisiones será mínimo y por ende el uso de la batería de los nodos y del espectro radioeléctrico el adecuado.

Como se pudo observar a lo largo de las pruebas realizadas la mejor implementación que se pueda hacer es usando el algoritmo diseñado junto con el protocolo CSMA, puesto que de esta forma se obtienen los tiempos más bajos en la transmisión.

Así mismo, el algoritmo cuenta con la función de enviar una trama avisando si es que el token habilitante no fue recibido lo cual puede ser aprovechado de diferentes maneras.

Los resultados arrojados del uso exclusivo del protocolo CSMA confirman la necesidad de tener que implementar un algoritmo propio sobre todo una red pequeña y de topología lineal, pues si bien con CSMA podemos evitar colisiones no es suficiente al momento de implementarlo en una red donde el número de dispositivos es reducido.

Una limitante en la realización de este trabajo es el lenguaje de codificación C/C++ con el que se programan los nodos transceptores, pues este no permite la depuración en tiempo real sino solo después de la compilación, lo cual complica el trabajo y hace que su codificación requiera de más trabajo.

3.2 CONCLUSIONES

Con lo presentado a lo largo de este trabajo, se puede afirmar que el principal objetivo se cumplió a cabalidad y de manera satisfactoria, pues en este trabajo se diseñó desde cero un algoritmo de acceso al canal en topologías líneas que operan con el protocolo IEEE 802.15.4 y el funcionamiento de este se pudo constatar a lo largo de toda la sección de resultados.

Este algoritmo resuelve la problemática planteada pues se presenta como una alternativa viable para redes que tengan una pequeña cantidad de nodos y que empleen una topología lineal como suele ser en una gran cantidad de escenarios que ya fueron mencionados en la introducción de este trabajo, sin embargo, eso no impide que con trabajo adicional este algoritmo sea adaptado a un escenario distinto según la situación lo amerite.

Con la ayuda del prototipo implementado en este trabajo se demostró que efectivamente el protocolo CSMA no es la mejor alternativa al momento de implementar una red pequeña en un entorno real, pues existe pérdida de tramas, errores en la asignación de direcciones y tiempos de transmisión más largos, por lo que se requiere de manera obligatoria un

algoritmo que facilite el acceso al canal y que aproveche los recursos de cada uno de los nodos que contemplan la red, como el presentado en este trabajo de titulación.

En este tipo de redes donde la topología es lineal, se puede aprovechar que el envío de datos es secuencial, por lo que, el algoritmo diseñado facilita el correcto direccionamiento de las tramas a enviar.

Se demostró también que el uso de un token que habilite la transmisión es una de las tantas maneras correctas que se tiene para habilitar la transmisión de tramas entre los nodos, inclusive puede ser enviado junto con información adicional en el payload, lo cual nos permite enviar información y controlar la transmisión simultáneamente.

El nodo transceptor ATZB – 256RFR2 presenta algunas limitaciones debido a su arquitectura y sobre todo al lenguaje de programación (C/C++) con el que trabaja, pues no permite una corrección de errores en tiempo real sino solamente después de compilar el programa, lo cual implica un tiempo mayor de dedicación al trabajo.

Finalmente, como se pudo constatar el algoritmo funciona de una mejor manera al trabajar en conjunto con el protocolo CSMA, pues con la ayuda de este se reduce el tiempo sobre todo en la primera transmisión que realiza el nodo con su consiguiente, es decir en el envío de la primera trama.

3.3 RECOMENDACIONES

Es importante tener en cuenta el canal en el que se transmite la información entre los nodos, puesto que de seleccionar un canal distinto en la configuración inicial de los nodos cuando se crea un nuevo proyecto, la frecuencia será distinta y la comunicación no se concretará o puede haber algún tipo de interferencia con otros nodos experimentales.

Es recomendable colocar tiempos iguales o similares en la función delay de los nodos dos al cinco, puesto que al tener tiempos iguales estos se sincronizan de una manera correcta al momento de realizar el procesamiento de las tramas enviadas, lo cual garantiza un funcionamiento adecuado del algoritmo.

Se debe tener en cuenta que los nodos no pueden transmitir dos tramas simultáneas simplemente apilando dos funciones de transmisión, es necesario idear otra solución al momento de querer enviar más de una trama simultáneamente, como se hizo en este trabajo, usando una variable auxiliar que cambie su valor y permita enviar una trama en cada iteración.

Se recomienda modificar las funciones de la dirección origen y de la dirección destino de la trama, puesto que esto facilita mucho el trabajo al no tener que editar esto en el archivo de configuración de cada uno de los nodos.

Se debe tener en cuenta que los nodos trabajan en su mayoría con valores hexadecimales, para los cuales existen funciones propias del nodo para realizar esta conversión.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] K. Sohraby, D. Minoli, and T. Znati, “Wireless Sensor Networks,” 1st ed. New Jersey: John Wiley & Sons, Inc, 2007

[2] V. A. “Wireless Sensor Networks with Waspmote and Meshlium,” Libelium. Zaragosa, España. Feb. 1, 2013. [En línea]. Disponible en: https://www.libelium.com/wp-content/uploads/2013/02/wsn-waspmote_and_meshlium_eng.pdf

[3] M. Pule, A. Yahya, and J. Chuma, “Wireless Sensor Networks: A survey on monitoring water quality,” IEEE. Journal of Applied Research And Technology, vol. 15, Jan 2018.

[4] T. Brougham, G. M. Nikolopoulos, and I. Jex, “Communication in quantum networks of logical bus topology,” Journal of Physical Review A, vol. 80, Nov 2009. DOI: 10.1103/PhysRevA.80.052325

[5] R. G. Paredes, “Sociedad e Información,” UDLAP. San Andrés Cholula, México. [En línea]. Disponible en: https://sites.google.com/site/is118sei/home

[6] A. Gezer, and S. Okdem, “Improving IEEE 802.15.4 channel access performance for IoT and WSN devices,” IEEE. Journal of Computers and Electrical Engineering, vol. 87, Jun 2020.

[7] Atmel, “8-bit AVR Microcontroller with Low Power 2.4GHz Transceiver for ZigBee and IEEE 802.15.4,” Versión 9.0-14 [En línea]. Disponible en: https://www.mouser.com/datasheet/2/268/Atmel-8266-MCU_Wireless ATmega128RFA1_Summary_Data-464631.pdf

[8] E. A. Salguero Jaya, “Implementación de un medidor de nivel de potencia para el estándar IEEE 802.15.4,” EPN. Quito, Ecuador, 2016 [En línea]. Disponible en: https://bibdigital.epn.edu.ec/handle/15000/15242

[9] Atmel, “Atmel AVR10002: ATmega256RFR2 Evaluation Kit User Guide 8-bit Atmel Microcontroller,” Versión 3.0-13 [En línea]. Disponible en: https://ww1.microchip.com/downloads/en/Appnotes/Atmel-42082-ATmega256RFR2-Evaluation-Kit-User-Guide_Application-Note_AVR10002.pdf

[10] Atmel, “Atmel AVR10004: RCB256RFR2 Hardware User Manual 8-bit Atmel Microcontrollers,” Versión 3.0-13 [En línea]. Disponible en: https://ww1.microchip.com/downloads/en/AppNotes/Atmel-42081-RCB256RFR2-Hardware-User-Manual_Application-Note_AVR10004.pdf

[11] J. X. Arciniega Barahona, “Diseño e implementación de una herramienta de monitoreo y captura de tramas en redes IEEE 802.15.4,” EPN. Quito, Ecuador, 2016 [En línea]. Disponible en: https://bibdigital.epn.edu.ec/handle/15000/16474

5 ANEXOS

ANEXO I

CÓDIGO CSMA

NODO 1

```
#include "usr_wireless.h"
#include "wireless_config.h"

void usr_wireless_app_task(void)
{
	uint16_t addpropia; //Variable de dirección origen declarada
	addpropia=1; //Dirección de origen del nodo transmisor
	addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
	tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la
dirección

	#ifdef TRANSMITTER_ENABLED //Etapa de transmisión
		transmit_sample_frame((uint8_t*)"CSMA 1", 6, 0x002); //Función
modificada de transmisón
		delay_ms(10000); //Se envía cada 10 segundos
	#endif
}

void usr_frame_received_cb(frame_info_t *frame)
{

}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

NODO 2

```
#include "usr_wireless.h"
#include "wireless_config.h"
int habilitarTX=0; //Variable que ayuda a la transmisión simultánea de tramas

void usr_wireless_app_task(void)
{
	uint16_t addpropia; //Variable de dirección origen declarada
	addpropia=2; //Direccion de origen del nodo transmisor
	addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
	tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la
direción

	if (habilitarTX==1)
	{
		habilitarTX=2;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA2", 12, 0x003); //Función
modificada de transmisión
	}
	else if (habilitarTX==2)
	{
		habilitarTX=0;
```

```
            transmit_sample_frame((uint8_t*)"CSMA PRUEBA3", 12, 0x003); //Función
modificada de transmisión
      }

      delay_ms (500); //Se envía cada 0.5 segundos
}

void usr_frame_received_cb(frame_info_t *frame) //Cuando el nodo recibe una trama
ejecuta esta función
{
      memset(&trama_recibida,0,sizeof(trama_recibida));//Encera el vector
      memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));//Copia la
informacion del buffer al destino
      bmm_buffer_free(frame->buffer_header);//Limpia los datos del buffer, evita
superposicion.

      habilitarTX=1;

}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

NODO 3

```
#include "usr_wireless.h"
#include "wireless_config.h"
int habilitarTX=0; //Variable que ayuda a la transmisión simultánea de tramas

void usr_wireless_app_task(void)
{
      uint16_t addpropia; //Variable de dirección origen declarada
      addpropia=3; //Direccion de origen del nodo transmisor
      addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
      tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la
direción

      if (habilitarTX==1)
      {
            habilitarTX=2;
            transmit_sample_frame((uint8_t*)"CSMA PRUEBA4", 12, 0x004); //Función
modificada de transmisión
      }
      else if (habilitarTX==2)
      {
            habilitarTX=3;
            transmit_sample_frame((uint8_t*)"CSMA PRUEBA5", 12, 0x004); //Función
modificada de transmisión
      }
      else if (habilitarTX==3)
      {
            habilitarTX=0;
            transmit_sample_frame((uint8_t*)"CSMA PRUEBA6", 12, 0x004); //Función
modificada de transmisión
      }
      delay_ms (1500); //Se envía cada 0.5 segundos
}
```

```
void usr_frame_received_cb(frame_info_t *frame) //Cuando el nodo recibe una trama
ejecuta esta función
{
	memset(&trama_recibida,0,sizeof(trama_recibida));//Encera el vector
	memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));//Copia la
informacion del buffer al destino
	bmm_buffer_free(frame->buffer_header);//Limpia los datos del buffer, evita
superposicion.

	habilitarTX=1;

}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

NODO 4

```
#include "usr_wireless.h"
#include "wireless_config.h"
int habilitarTX=0; //Variable que ayuda a la transmisión simultánea de tramas

void usr_wireless_app_task(void)
{
	uint16_t addpropia; //Variable de dirección origen declarada
	addpropia=3; //Direccion de origen del nodo transmisor
	addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
	tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la
dirección

	if (habilitarTX==1)
	{
		habilitarTX=2;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA7", 12, 0x004); //Función
modificada de transmisión
	}
	else if (habilitarTX==2)
	{
		habilitarTX=3;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA8", 12, 0x004); //Función
modificada de transmisión
	}
	else if (habilitarTX==3)
	{
		habilitarTX=4;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA9", 12, 0x004); //Función
modificada de transmisión
	}
	else if (habilitarTX==4)
	{
		habilitarTX=0;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA10", 13, 0x004); //Función
modificada de transmisión
	}
	delay_ms (1500); //Se envía cada 0.5 segundos
}
```

```
void usr_frame_received_cb(frame_info_t *frame) //Cuando el nodo recibe una trama
ejecuta esta función
{
	memset(&trama_recibida,0,sizeof(trama_recibida));//Encera el vector
	memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));//Copia la
informacion del buffer al destino
	bmm_buffer_free(frame->buffer_header);//Limpia los datos del buffer, evita
superposicion.

	habilitarTX=1;

}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```

NODO 5

```
#include "usr_wireless.h"
#include "wireless_config.h"
int habilitarTX=0; //Variable que ayuda a la transmisión simultánea de tramas

void usr_wireless_app_task(void)
{
	uint16_t addpropia; //Variable de dirección origen declarada
	addpropia=3; //Direccion de origen del nodo transmisor
	addpropia=CCPU_ENDIAN_TO_LE16(addpropia); //Se pasa a hexadecimal
	tal_pib_set(macShortAddress, (pib_value_t *)&addpropia); //Se modifica la
direción

	if (habilitarTX==1)
	{
		habilitarTX=2;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA11", 13, 0x004); //Función
modificada de transmisión
	}
	else if (habilitarTX==2)
	{
		habilitarTX=3;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA12", 13, 0x004); //Función
modificada de transmisión
	}
	else if (habilitarTX==3)
	{
		habilitarTX=4;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA13", 13, 0x004); //Función
modificada de transmisión
	}
	else if (habilitarTX==4)
	{
		habilitarTX=5;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA14", 13, 0x004); //Función
modificada de transmisión
	}
	else if (habilitarTX==5)
	{
		habilitarTX=0;
		transmit_sample_frame((uint8_t*)"CSMA PRUEBA15", 13, 0x004); //Función
modificada de transmisión
```

```
        }
        delay_ms (1500); //Se envía cada 0.5 segundos
}

void usr_frame_received_cb(frame_info_t *frame) //Cuando el nodo recibe una trama
ejecuta esta función
{
        memset(&trama_recibida,0,sizeof(trama_recibida));//Encera el vector
        memcpy(&trama_recibida,frame->mpdu,sizeof(trama_recibida));//Copia la
informacion del buffer al destino
        bmm_buffer_free(frame->buffer_header);//Limpia los datos del buffer, evita
superposicion.

        habilitarTX=1;

}

void usr_frame_transmitted_cb(retval_t status, frame_info_t *frame)
{

}
```