UNIVERSIDAD ESTATAL PENÍNSULA

DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CARRERA DE TELECOMUNICACIONES

TEMA:

"DESARROLLO DE SISTEMAS CON CÓDIGO ABIERTO PARA LA IMPLEMENTACIÓN DE MODULACIÓN BPSK Y QPSK MEDIANTE SDR"

AUTOR

VILLÓN BORBOR, KATHERINE JEANINA

TRABAJO DE TITULACIÓN

TRABAJO COMPLEXIVO, PREVIO A LA OBTENCIÓN DEL GRADO ACADÉMICO EN INGENIERA EN TELECOMUNICACIONES

TUTOR

ING. DANIEL ARMANDO JARAMILLO CHAMBA. M.SC.

Santa Elena, Ecuador

Año 2024

DEDICATORIA

Este logro dedico a quienes estuvieron desde el inicio de mi formación académica, quienes a pesar de las dificultades siempre estuvieron alentándome a seguir, sin duda alguna, a mis padres, por darme un hogar lleno de amor, valores, y principios, además por confiar en cada etapa de mi vida.

A mi madre, gracias por siempre estar conmigo por tu amor y bondad, por tus palabras de aliento, enseñarme a nunca rendirme, a perseguir mis sueños, sin importar los obstáculos.

A mi padre, por ser el pilar fundamental de nuestra familia, gracias por ser mi ejemplo de dedicación, esfuerzo, trabajo y por los consejos impartidos.

A mis hermanos y hermanas, por ser mis compañeros de vida, cada uno de ustedes, con su carácter, han sido una inspiración constante en mi aprendizaje.

A mis sobrino y sobrinas, quienes con su alegría, inocencia y cariño han llenado mis días de motivación y ternura, recordándome siempre que hay razones para seguir adelante.

Y finalmente, a quienes ya no están físicamente, en especial a mis abuelos, cuya presencia sigue viva en mi corazón. Gracias por las enseñanzas que me dejaron, por los momentos compartidos y los recuerdos que siempre me acompañan.

KATHERINE JEANINA VILLÓN BORBOR

AGRADECIMIENTO

Primero, agradezco a Dios, por darme la fuerza, la sabiduría y la oportunidad de llegar hasta aquí.

Agradezco a mis padres: Félix Villón Borbor y Letty Borbor Tomalá, quienes con su guía y apoyo constante han sido fundamentales en cada paso de este camino.

A la universidad, por brindarme la oportunidad de aprender, crecer y formarme como profesional, gracias por el conocimiento y las experiencias que me han acompañado durante este tiempo, las cuales han sido esenciales en mi desarrollo académico y personal.

A los ingenieros, por la formación académica impartida y el tiempo compartido en las aulas de la institución.

Al docente tutor el Ing. Daniel Jaramillo, por guiarme y aportarme los conocimientos necesarios para el desarrollo y culminación de este trabajo, de la misma manera al Ing. Amaya, docente especialista, por su disposición, asesoría y guía profesional.

A mis amigos, por su apoyo incondicional y por las experiencias compartidas, las cuales me ayudaron a crecer, gracias por su amistad y colaboración siempre.

KATHERINE JEANINA VILLÓN BORBOR

DECLARACIÓN AUTORÍA DEL ESTUDIANTE

El presente trabajo de Integración Curricular con el título **"Desarrollo de sistemas con código abierto para la implementación de modulación BPSK y QPSK mediante SDR."** Declaro que la Concepción, análisis y resultados del trabajo presentado son originales y le dan un aporte importante a la actividad educativa dentro del área telecomunicación.

Atentamente,

Katherine

Srta. Villón Borbor Katherine Jeanina

C.I. 2400461279

DECLARACIÓN DE RESPONSABILIDAD

Quienes suscriban, **Villon Borbor Katherine Jeanina** con C.I. **2400461279**, estudiante de la carrera de Telecomunicaciones, declaro que el trabajo de Titulación presentado a la unidad de Integración Curricular cuyo tema es **"Desarrollo de sistemas con código abierto para la implementación de modulación BPSK y QPSK mediante SDR.",** corresponde y es de exclusiva responsabilidad del autor y pertenece al patrimonio intelectual de la Universidad Estatal Península de Santa Elena.

Atentamente,

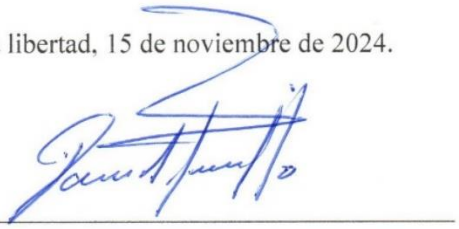
Srta. Villón Borbor Katherine Jeanina

C.I. 2400461279

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación denominado: "Desarrollo De Sistemas Con Código Abierto Para La Implementación De Modulación BPSK Y QPSK Mediante SDR", presentado por la estudiante Villon Borbor Katherine Jeanina, de la carrera de Telecomunicaciones de la Universidad Estatal Península de Santa Elena, me permito declarar que luego de haber orientado, revisado, apruebo en todas sus partes y autorizo a la estudiante inicie con los trámites correspondientes.

La libertad, 15 de noviembre de 2024.



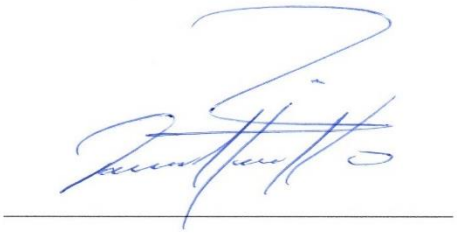
Ing. Daniel Armando Jaramillo Chamba. Msc

DOCENTE TUTOR

TRIBUNAL DE SUSTENTACION DE TRABAJO DE INTEGRACION CURRICULAR

Ing. Ronald Rovira Jurado, Ph.D
DIRECTOR DE LA CARRERA

Ing. Luis Amaya Fariño, Mgtr
DOCENTE ESPECIALISTA



Ing. Daniel Jaramillo Chamba, Mgtr
DOCENTE TUTOR

Ing. Corina Gonzabay De la A, Mgtr
SECRETARIA

ÍNDICE DE CONTENIDO

Índice de Figuras

Índice de Tablas

ACRÓNIMOS

SDR - Software defined Radio

BER - *Bit Error Rate*

RF - *Radio Frequency*

IF - *Intermediate Frequency*

BB - *Base Band*

SoC – *System on Chip*

ADC – *Analog to Digital Converter*

DAC – *Digital to Analog Converter*

AM – *Amplitude Modulation*

FM - *Frequency Modulation*

FSK – *Frequency Shift Keying*

ASK - *Amplitude Shift Keying*

PSK – *Phase Shift Keying*

BPSK – *Binary Phase Shift Keying*

QPSK – *Quadrature Phase Shift Keying*

DQPSK – *Differential Quadrature Phase Shift Keying*

NRZ – *Not Return to Zero*

ODFM - *Orthogonal Frequency Division Multiplexing*

LTE - *Long Term Evolution*

WLAN – *Wireless Local Area Network*

WiMAX - *Worldwide Interoperability for Microwave Access*

ISDB-Tb - *Integrated Services of Digital Broadcasting – Terrestrial*

DFT – *Discrete Fourier Transform*

FFT - *Fast Fourier Transform*

ISI - *Inter Symbol Interference*

ICI - *Inter Carrier Interference*

RESUMEN

Este proyecto se centra en el desarrollo de sistemas de comunicación a través de SDR, con el objetivo de mejorar la enseñanza de las telecomunicaciones en estudiantes universitarios, facilitando la comprensión a través de prácticas de laboratorios que ayuden a comprender la teoría.

Se emplea una metodología que investigación bibliográfica y experimental como utilizando software como GNU radio y Matlab para realizar estas prácticas que demuestren efectividad en la transmisión de recepción de datos.

Primera práctica corresponde al diseño de un sistema de comunicación básica utilizando modulación ASK, permitiendo comprender los fundamentos de transmisión y recepción, la segunda práctica corresponde a las modulaciones digitales BPSK y QPSK evaluando la tasa de error en ambas modulaciones, la tercera práctica consiste en desarrollar un sistema de comunicación de banda ancha utilizando modulación OFDM, donde se analiza el ancho de banda y eficiencia del sistema, en cada práctica se realiza una comparativa entre ambos softwares.

Los resultados obtenidos demuestran la importancia en la calidad de la señal transmitida y recibida y en la integridad de los datos evidenciando la necesidad de recursos educativos basados en software de código abierto mejorando la formación académica en estudiantes.

Palabras claves: Radio Definida por Software (SDR), Modulación digital, Tasa de error de bits (BER), HackRF One, MATLAB, GNU Radio.

ABSTRACT

This project focuses on the development of communication systems through SDR, with the aim of improving the teaching of telecommunications in university students, facilitating understanding through laboratory practices that help to understand the theory.

A methodology is used that includes bibliographic and experimental research, using software such as GNU radio and Matlab to carry out these practices that demonstrate effectiveness in the transmission and reception of data.

The first practice corresponds to the design of a basic communication system using ASK modulation, allowing to understand the fundamentals of transmission and reception, the second practice corresponds to BPSK and QPSK digital modulations evaluating the error rate in both modulations, the third practice consists of developing a broadband communication system using OFDM modulation, where the bandwidth and efficiency of the system are analyzed, in each practice a comparison is made between both software.

The results obtained demonstrate the importance of the quality of the transmitted and received signal and the integrity of the data, evidencing the need for educational resources based on open-source software improving academic training in students.

Keywords: Software Defined Radio (SDR), Digital Modulation, Bit Error Rate (BER), HackRF One, MATLAB, GNU Radio

CAPÍTULO I

1.1 Antecedentes Investigativos

En la actualidad las comunicaciones inalámbricas han impulsado el desarrollo y adopción de tecnologías SDR *Software defined Radio*, donde los elementos de la capa física, implementados a través de un hardware se programan mediante software, permitiendo desarrollar sistemas de comunicación flexibles y adaptables para la trasmisión de datos[1].

La primera implementación de SDR inicio en el año de 1991 y concluye en al año de 1995, en el proyecto militar *SpeakEasy* de Estados Unidos, cuyo objetivo fue desarrollar un sistema de radio definida por software que tenga como mínimo 10 tecnologías de comunicaciones inalámbricas en un solo equipo programable (Hardware) y su sistema operaba en el rango de las bandas de 2MHz a 20 MHz[2].

El proyecto desarrollado por Nelson Enrique Arias, en la Universidad Militar Nueva Granada, titulado "Trasmisión y Recepción de Imagen con Modulación QPSK, usando Radio Definido por Software", tiene como objetivo principal diseñar y simular un sistema que permita la trasmisión de imágenes utilizando la modulación QPSK *Quadrature Phase Shift Keying*, el proyecto realiza la implementación mediante el software GNU Radio. Las pruebas realizadas evaluaron el rendimiento del sistema, analizando la tasa de error de bits o *BER Bit Error Rate* y la efectividad en la recepción de imágenes, los resultados obtenidos demuestran la viabilidad del sistema propuesto, evidenciando su capacidad en realizar excelentes transmisiones a pesar de las interferencias comunes en los canales de comunicación, al concluir el trabajo siguieren mejoras en los algoritmos de corrección de errores y optimización del hardware[3].

En la investigación "Laboratorio de Comunicaciones Digitales Radio Definida por Software" presentado por Iván Pinar y Juan Murillo, tiene como objetivo desarrollar prácticas para laboratorio de radiocomunicación utilizando tecnología SDR, mediante un hardware de modelo USPR-1, con el fin de mejorar la enseñanza e investigación en las telecomunicaciones, utilizando metodología implementada, con programación de SDR en Matlab y el uso de librerías *toolbox SDR4all* para transmitir señales, así como la aplicación de técnicas como DQPDK *Differential Quadrature Phase Shift Keying* y BPSK *Binary Phase Shift Keying*, mediante pruebas realizadas de transmisión secuencia de bits y evaluación de la tasas de error de bits, demostrando efectividad a pesar, que al igual que la investigación anterior presento ciertas complicaciones en la

implementación, concluyendo que el SDR permite el desarrollo de experimentos prácticos para comprender los principios del área de telecomunicaciones[4].

En Simposium Nacional de la unión científica Internacional de radio 2020, Carmen Botella y su equipo presentaron el proyecto titulado "Evaluación del impacto del uso de dispositivos de radio definida por software como herramienta docente en la materia de comunicaciones digitales", se enfoca en la implementación de tecnologías de Radio Definida por Software en la asignatura de comunicaciones digitales con el fin de mejorar la motivación e interés de los estudiantes mediante prácticas de laboratorio que reflejan los sistemas de comunicación en la actualidad, la metodología del estudio se basa en la comparación entre tres grupos que se dividen uno en experimental con dispositivo SDR y dos grupos de control que siguen las prácticas tradicionales, mediante encuestas se evalúa el compromiso e importancia de los estudiantes en las actividades realizadas, mostrando resultados positivos en la capacidad de los estudiantes para afrontar nuevos desafíos, el proyecto concluye resaltando la importancia de integrar nuevas tecnologías en la educación, para comprender los concepto teóricos en prácticas para el área de telecomunicaciones[5].

El uso de plataformas de SDR con HackRF One y herramientas de código abierto como GNU Radio y Matlab, permite desarrollar diferentes escenarios, en este proyecto se desarrollará el diseño y simulación de los sistemas de comunicaciones enfocadas a modulación digital mediante tres prácticas de laboratorio.

1.2 Planteamiento del Problema y Justificación

En el ámbito académico, el diseño de sistemas de comunicación digital es un componente clave en la formación de estudiantes en la carrea de telecomunicaciones o carreras afines, las prácticas o talleres ayudan a reforzar el aprendizaje teórico mediante sistemas para la trasmisión y recepción de datos, sin embargo, una de las problemáticas que se enfrenta son la falta de recursos educativos y guías prácticas que detallen como utilizar SDR y software de código abierto en esquemas de modulación, teniendo en cuenta, que estas actividades resultan esenciales para estudiantes que se les dificulta comprender conceptos teóricos.

El fin de este trabajo es proporcionar guías y recursos prácticos permite a los estudiantes de la universidad interactuar con tecnologías avanzadas en un entorno educativo, estas ayudaran a los estudiantes a visualizar y comprender el funcionamiento de trasmisión y recepción de datos utilizando modulación digital.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar sistemas de modulación digital con software de código abierto y software definido por radio, con el fin de evaluar su desempeño en la trasmisión y recepción de datos.

1.3.2 Objetivos Específicos

- Describir los conceptos teóricos necesarios para comprender los sistemas de comunicación digital.
- Desarrollar los diagramas de bloques para trasmisión y recepción en GNU Radio y Matlab.
- Realizar análisis comparativo los resultados obtenidos de cada práctica en los softwares empleados.

1.4 Alcance

El proyecto consta de tres prácticas, enfocadas en aplicación de conceptos teóricos mediante simulaciones de sistemas de comunicación, teniendo en cuenta que cada práctica permite al estudiante mejorar el conocimiento en trasmisión y recepción de datos.

Práctica 1: Diseño y Simulación de un Sistema de Comunicación Simple.

Descripción: La práctica consiste en desarrollar un esquema básico de comunicación que permita al estudiante trasmitir y recibir datos utilizando técnica de modulación digital ASK *Amplitude Shift Keying* mediante datos binarios que desee transmitir o por datos aleatorios generados por el software.

Práctica 2: Implementación y Evaluación de Esquemas de Modulación Digital

Descripción: La práctica consiste en desarrollar y simular dos esquemas de modulaciones digitales, BPSK *Binay Phase Shift Keying* y QPSK *Quadrature Phase Shift Keying*, que permitan la trasmisión y recepción de datos, además de evaluar la tasa de error de bits de las modulaciones.

Práctica 3: Desarrollo de un Sistema de Comunicación de Banda Ancha

Descripción: La práctica consiste en desarrollar un sistema de comunicación de banda ancha para transmitir y recibir datos utilizando modulación OFDM *Orthogonal Frequency Division Multiplexing*, evaluando la eficiencia del sistema y el ancho de banda.

1.5 Fundamentación Teórica

1.5.1 SDR - Software Defined Radio

En 1991 Joseph Mitola, utiliza el concepto de SDR Radio Definida por software, con una plataforma de hardware llamado radio, empleados para trasmitir y recibir señales, dando uso para aplicaciones militare, civiles, tarjetas de radio que están en los diferentes dispositivos como teléfonos, celulares, computadoras, entre otras.

Los dispositivos SDR básicamente está compuesto por un sintonizador con un oscilador de frecuencia, un mezclador, un amplificador y un filtro encargado de convertir la señal a frecuencia intermedia también contiene un conversor analógico - digital que procesa la señal y pasa al software, este dispositivo permite enviar y recibir información de manera inalámbrica mediante una parte del espectro electromagnético con diferentes métodos de modulación[6].

1.5.2 Esquema de SDR

SDR es una tecnología donde los módulos de software son ejecutados en tiempo real mediante hardware como microprocesadores, circuitos lógicos programables, en la figura 1 se visualiza el esquema de un SDR en tres secciones: RF - *Radio Frequency*, IF - *Intermediate Frequency* y BB *Base Band* [7].

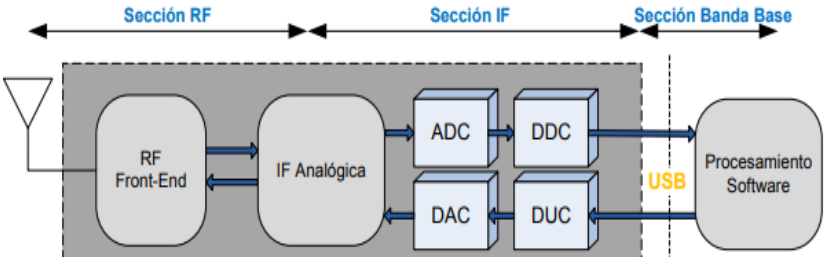


Figura 1. Esquema de SDR

Fuente: [7]

Radio Frequency - RF: Se encarga de enviar y recibir la señal de radio frecuencia por medio de la antena, adaptándola, realizando una conversión ascendente en la transmisión de señal IF a RF y conversión descendente para una señal IF en la recepción[7].

Intermediate Frequency – IF: Su función es pasar la señal IF a BB, digitaliza en la recepción convirtiendo la señal BB a IF y para la transmisión realiza conversión digital a analógica, los bloques convertidore analógico - digital y viceversa, son encargados de convertir la señal para la

trasmisión y recepción de información, los bloques convertidor digital hacia abajo realizan la conversión digital descendente logrando pasar de señal IF a BB en la recepción, por otro lado el bloque convertidos digital hacia arriba realiza la conversión ascendente y se encarga que la señal IF suba a BB en la trasmisión[7].

Base Band – BB: se encarga de realizar operación de configuración para la conexión, recuperar el sincronismo, salto de frecuencia, entre otras[7].

1.5.3 Arquitectura de SDR

La arquitectura de SDR se considera tanto el hardware como el software, la ubicación del ADC *Analog to Digital Converte* o DAC *Digital To Analog Converter*, determina si es analógico o digital la señal que está enviando o recibiendo en la figura 2 se observa la arquitectura del SDR.

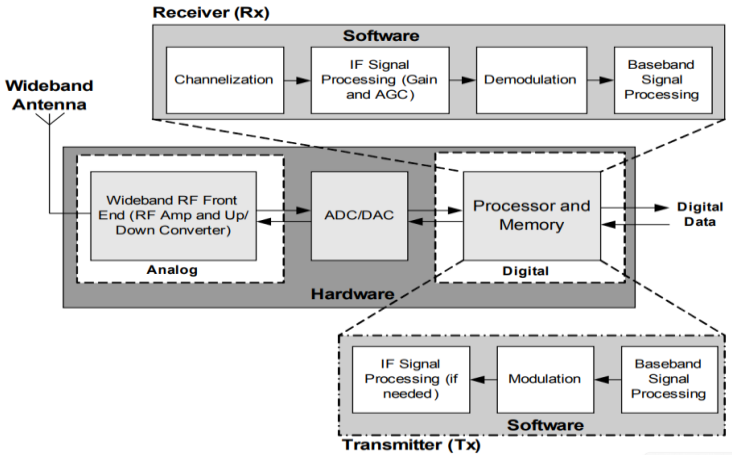


Figura 2. Arquitectura general de SDR

Fuente: [8]

Hardware: Se divide en dos subsistemas como es digital y analógico, el subsistema analógico realiza funciones de filtrado, combinación de RF, amplificador de potencia, preamplificador, generar frecuencia de referencia[8].

Software: Se divide en dos componentes como es la trasmisión, empieza con el procesamiento de la señal de banda base para modificar la señal o datos a enviar, en el proceso de la señal la portadora IF es modulada, convirtiéndole en señal RF, para luego enviarle al canal de comunicación por medio de antena, por parte de la recepción empieza con la selección y conversión ascendente de

la señal RF a señal IF, aplicando técnicas para la demodulación y extraer la señal, finalmente el procesamiento de banda base obtiene la información que envía el transmisor[8].

1.5.4 Características de SDR

Las características del SDR pueden tener destinos puntos de vista dependiendo los usuarios que se beneficien dividendo en diferentes secciones[7], como se observa en la tabla 1.

Usuarios	Ventajas
Fabricantes	• Reutilización de software en equipos de radio, reduce costos • Corrige fallos del equipo de radio mediante software, reduce costos de mantenimiento y operación.
Proveedores	• Descargas de software de manera remota, permite aumentar la capacidad. • Añade más capacidades y características a la estructura sin necesidad de aumentar costos. • Uso de plataforma de radio común, reduce costo de operación y logística.
Usuario Final	• Cambia las funciones del equipo sin necesidad de cambiar el hardware. • Capacidad para controlar funciones de un dispositivo de comunicación.

Tabla 1. Características de SDR dependiendo el Usuario

Fuente: Elaborado por autor

CAPíTULO II

Metodología

En esta sección se detalla los componentes necesarios en el desarrollo del proyecto, incluye los métodos empleados en la investigación y materiales utilizados como: hardware, software, para alcanzar los objetivos planteados en el capítulo I.

2.1 Metodología de la Investigación

Investigación Bibliográfica

La investigación bibliográfica es un proceso de búsqueda, recopilación y análisis de información provenientes de fuentes como libros, artículos científicos, artículos académicos, tesis, entre otros, con el objetivo de profundizar un tema específico, ayudando a identificar teorías o enfoques previos en la investigación[9].

Investigación Experimental

Esta es un tipo de investigación en el que el investigador manipula uno a más variables independientes para observa su efecto en las variables dependientes mediante un entorno controlado[10].

Para el desarrollo del proyecto se divido en dos fases. En la primera fase para el desarrollo del proyecto se realiza un estudio del estado de arte sobre los temas abordados en las diferentes prácticas como modulación digital y herramienta del SDR, realizando la investigación bibliográfica en distintos sitios, revistas científicas, publicaciones realizadas en Google académico. En la segunda fase se realiza la implementación de los esquemas de comunicación, realizando pruebas de trasmisión y recepción de datos, con el fin de evaluar los resultados obtenidos en los softwares empleados y realizar comparaciones en cada práctica.

El desarrollo incluye la configuración de los bloques estableciendo parámetros específicos y condiciones necesarias para la evaluación de los distintos escenarios propuestos, primero se procede a la implementación del transmisor si los resultados son exitoso se continua con la implementación del receptor, de lo contrario, se realiza ajustes en los parámetros del transmisor una vez obtenidos los resultados tanto del trasmisor y del receptor, los bloques son convertidos a línea de código en software Matlab, al igual que en GNU Radio si los resultados no exitosos se

vuelven a modificar las líneas correspondientes y si los resultados son exitosos se continua con el receptor, finalmente se realiza comparaciones de los resultados, documentando todo el proceso y realización de guías prácticas para los estudiantes, esperando obtener resultados que proporcionen un mejor entendiendo en las modulación digitales, como se muestra en la figura 3.

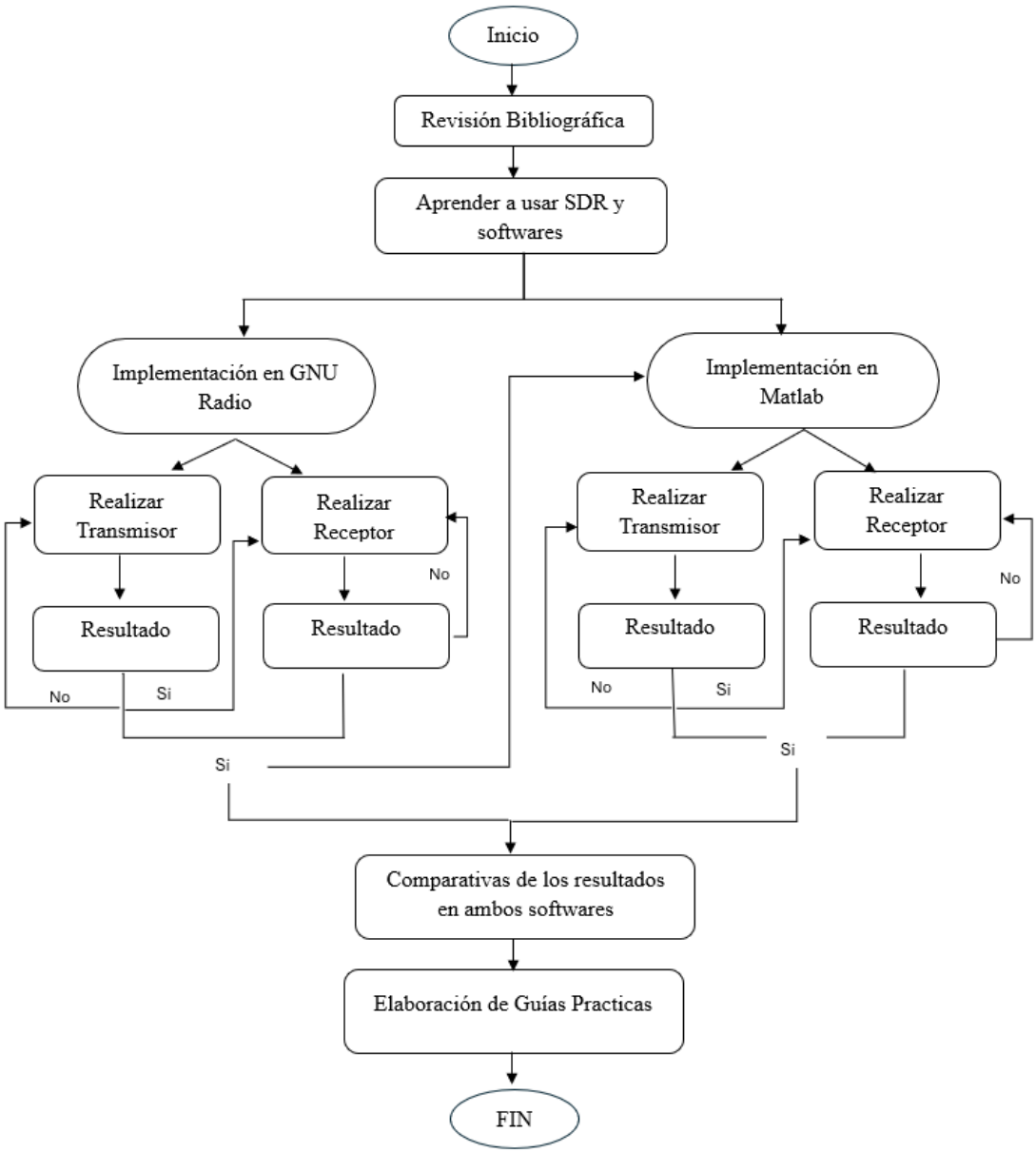


Figura 3. Esquema para la elaboración del proyecto

Fuente: Elaborado por autor

2.2 Materiales

2.2.1 Software

GNU Radio

GNU Radio es un software libre de código abierto, que permite al usuario desarrollar sistemas de procesamiento de señal de manera ilustrativa, conectando bloques que realizan las funciones específicas, como filtrados de señal, modulación o demodulación de señal y generar señales, se puede utilizar con Hardware de RF para crear entorno de Radio Definida por Software, además, si el usuario no cuenta con un hardware el software proporciona bloques que permiten generar las señales[11].

La programación en este software se puede realizar en Python o C++ dependiendo la aplicación que se necesita crear, sus tres elementos esenciales son fuentes, bloques para el procesamiento de señales y sumideros, donde las fuentes y sumideros son archivos, inputs, outputs, hardware de radio, mientras que los bloques para procesar las señales son filtros, amplificadores, moduladores, lógicos, operadores matemáticos, entre otros[12].

Al abrir la ventana de trabajo se cuenta con dos bloques iniciales por defecto como se muestra en la figura 4, donde el primer bloque corresponde al bloque de *options* indicando el nombre del proyecto y el tipo de instrumentación con las que se trabajara puede ser de tipo WX o tipo QT, el otro bloque es el de *variable* con el parámetro inicial de *sample_rate* con un valor inicial de 32KHz, y el software posee con diferentes tipos de datos diferenciados por colores[13].

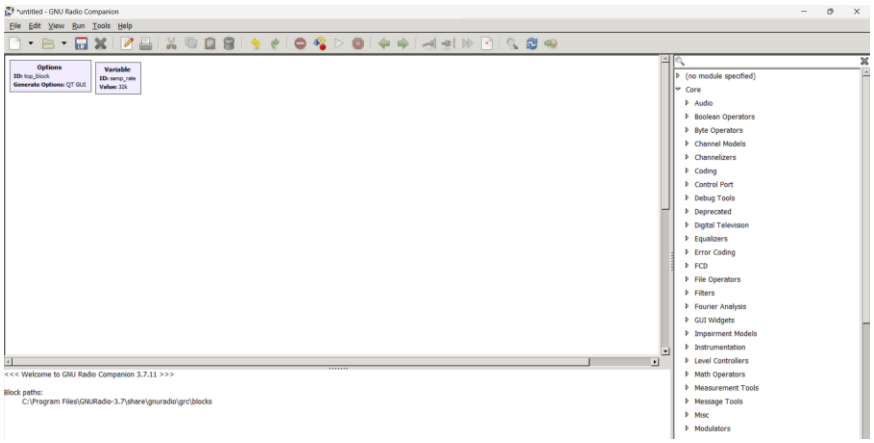


Figura 4. Ventana de trabajo de GNU Radio

Fuente: Elaborado por autor

Matlab

Es un lenguaje de programación desarrollado por MathWorks, específicamente para resolución de problemas matemáticos y análisis numérico, es característico por su capacidad de visualización, que permite la creación de gráficos en 2D y 3D facilitando el análisis e interpretación de datos. El software contiene bibliotecas *Toolboxes* especializadas en el área de procesamiento de señal, sistemas de control e inteligencia artificial[14].

Simulink es una extensión que contiene Matlab, el cual permite modelado y simulación de sistemas mediante bloques al igual que GNU radio, Matlab integra otros lenguajes de programación como C, C++ y Python, en la figura 5 se visualiza la ventana de trabajo del software Matlab[14].

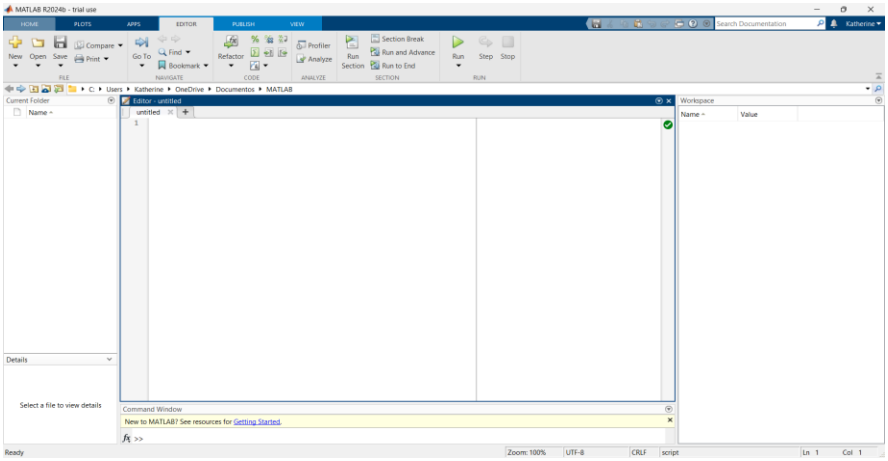


Figura 5. Ventana de trabajo de Matlab

Fuente: Elaborado por autor

Matlab tiene un dispositivo hardware disponible en el mercado para aplicaciones de SDR el cual contiene FPGA *Field Programmable Gate Array* o un SoC *System on Chip*, capaz de transmitir y recibir señales de distintas señales inalámbricas como 5G, LTE - *Long Term Evolution,* WLAN - *Wireless Local Area Network* y FM - *Frequency Modulation*, en la figura 6 se puede observar un esquema típico de la conectividad con Matlab[15].

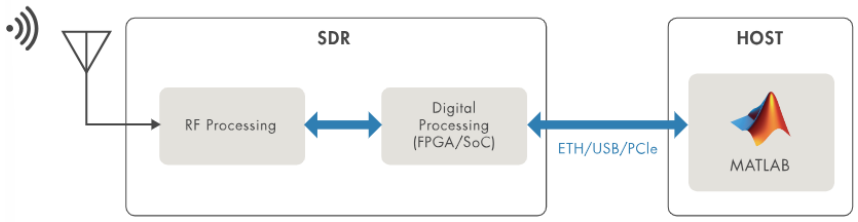


Figura 6. Esquema de conectividad de SDR y Matlab

Fuente: [14]

2.2.2 Hardware

La tecnología SDR se caracteriza por el uso de dos componentes principales la PC y el dispositivo SDR, en la figura 7 se observa el esquema general.

La PC es la encargada de procesar la señal en banda base, con funciones de codificar, modular, demodular por medio de software, mientras el dispositivo de radio definido por software está diseñado para implementar los sistemas de radio[12].

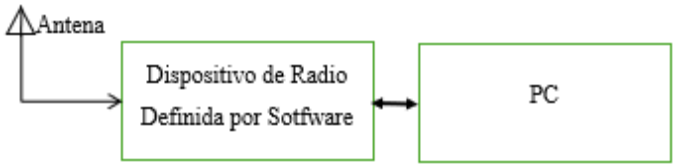


Figura 7. Esquema de componentes Hardware

Fuente: Elaborado por autor

Para el desarrollo de las simulaciones propuestas en el proyecto, se utilizó laptop Huawei como se muestra en la figura 8, con las especificaciones mostradas en la tabla 2.

Características	Descripción
Procesador	Intel(R) Core (TM) i3-10110U CPU @ 2.10GHz 2.59 GHz
RAM instalada	8 GB
Sistema Operativo	Sistema operativo de 64 bits, procesador x64
Versión de Windows	Windows 11

Tabla 2. Características de la PC

Fuente: Elaborado por autor

Figura 8. Laptop Marca Huawei utilizada para el desarrollo del proyecto

Fuente: Elaborado por autor

HackRF One

HackRF One Great Scott Gadgets es un dispositivo de radios definido por software con carcasa plástica, que permite transmitir y recibir señales de radio en el rengo de 1 MHz a 6 GHz, se conecta mediante cable USB a la computadora o se puede programar para que funcione de manera independente[16]. En la tabla 3 se muestran las características del dispositivo y en la figura 9 se observa el hardware de manera física.

Características	Descripción
Frecuencia de operación	1MHz a 6GHz
Modo de operación	semidúplex
Tasa de muestreo	20 millones/segundos
Resolución de muestras	8 bits en cuadratura
Softwares compatibles	GNU Radio, SDR#
Alimentación de la antena	50 mA a 3,3V
Conector de la antena	Hembra SMA
Interfaz USB	USB 2.0 de alta velocidad
Tipo de hardware	Código abierto

Tabla 3. Características del HackRF One

Fuente: [16]

Figura 9. Hardware HackRF One

Fuente: Elaborado por autor

Antena ANT500

Es un tipo de antena sencilla de tipo dipolo como se muestra en la figura 10, consta de una varilla conductora colocada de forma vertical, de tamaño pequeña, la antena tiene una impedancia de 50 ohm, y su rango de funcionamiento va de 75MHz a 1GHz, con una longitud total de 20 centímetros mínimo y 88 centímetro máximo, utilizada para banda de frecuencias moduladas FM y banda de onda corta[17].

Figura 10. Antena ANT500

Fuente: Elaborado por autor

CAPÍTULO III

Desarrollo de las Prácticas

La instalación del Software GNU Radio para el desarrollo de las prácticas se realizó mediante el paso a paso y el instalador versión 3.7.11 que proporciona la tesis titulada “Implementación de sistemas definidos por radio basados en código abierto para modulaciones PKS y QAM en fase de cuadratura”, realizada por Sáenz Valero Ruby, a partir de la página 64 hasta la pagina 72, misma que se encuentra en el repositorio de la Universidad Estatal Península de Santa Elena.

3.1 Práctica 1.- Diseño y Simulación de un Sistema de Comunicación Simple

3.1.1 Descripción de la importancia de los sistemas de comunicación simple

Un sistema de comunicación simple es esencial en el ámbito de las telecomunicaciones debido a que representa el inicio para comprender los principios de trasmisión, recepción, modulación y demodulación, este consta de tres elementos principales como son fuente, canal, destino, con el fin de transmitir y recibir señales de tipo analógica o digital como audio, datos, voz, videos, entre otros[18].

Estos sistemas permiten transmitir información de manera eficiente por diferentes medios pueden ser cableado, inalámbrico, óptico, teniendo en cuenta que el diseño y funcionamiento son importantes para garantizar las comunicaciones en diferentes aplicaciones como son radio, televisión, redes de datos y comunicaciones móviles[19].

Además, son importantes para la experimentación y el desarrollo de nuevas tecnologías en el área de telecomunicaciones, proporcionando un entorno controlado donde se logra probar y ajustar diferentes parámetros y técnicas sin las complicaciones adicionales que representan los sistemas avanzados.

Objetivo de la práctica: desarrollar y simular un sistema de comunicación básico de transmisión y recepción de datos utilizando HackRF One, GNU Radio y Matlab.

3.1.2 Revisión teórica

El proceso de transmisión y recepción de datos permite el intercambio de información por medio de diferentes canales de comunicación o en cualquier sistema de comunicación.

Trasmisión de datos

Fuente de datos: Es el origen de los datos que se van a enviar en el sistema de comunicaciones, es el responsable de generar la información a transmitir, puede ser texto, audio, video, imágenes o cualquier tipo de datos digitales.

Codificación: Se encarga de convertir los datos obtenidos de la fuente da datos en un formato adecuado para la trasmisión aplicando diferentes técnicas de modulación y conversión de datos en bits.

Modulación: Permite convertir una señal analógica o digital a un medio de transmisión, la modulación analógica incluye técnicas de modulación AM *Amplitude Modulation* y FM *Frequency Modulation*, mientras que la modulación digital incluye técnicas ASK *Amplitude Shift Keying*, FSK *Frequency Shift Keying*, PSK *Phase Shift Keying*.

Transmisión: una vez modulada la señal se envía a través del medio de trasmisión.

Medio de transmisión: Medio o canal por el que viaja la señal, puede ser un medio físico o medio inalámbrico.

Sincronización: Sincroniza la velocidad con la que se transmite en el emisor como en el receptor para que los datos se interpreten de manera correcta.

Recepción de datos

Recepción de señal: Se encarga de recibir la señal transmitida por medio de un dispositivo receptor, el cual debe tener la sensibilidad adecuada para detectar la señal.

Demodulación: Es el proceso inverso a la modulación este se encarga de recuperar los datos de la señal recibida.

Decodificación: Una vez con los datos demodulados se codifican para obtener la información original generada por la fuente.

Procesamiento de datos: Los datos recuperados se procesan y presentan al usuario para su verificación y corrección de errores.

Modulación ASK *Amplitude Shift Keying*

La modulación ASK es el cambio de amplitud de la onda portadora en forma de bits que se transmiten, representando ausencia o presencia de señal en un determinado tiempo mediante "1" y "0", estos datos pueden estar generados por un computador, hardware o tipo de archivo que se necesite transmitir, son utilizados en sistemas de corto alcance comúnmente en controles remotos de juguetes o televisores [20]. En la figura 11 se puede observar la modulación ASK en su forma digital.

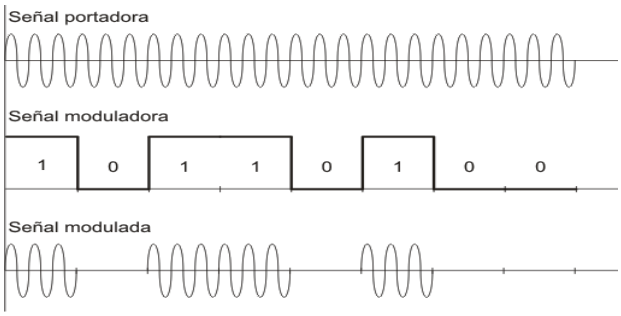


Figura 11. Modulación ASK

Fuente: [19]

Se puede decir que la modulación ASK es la modulación OOK, permitiendo transmitir pulso sinusoidal para un determinado bit y el valor de cero para el otro bit, donde 1 se codifica de la misma manera que la señal ASK y el bit 0 es codificado con el valor 0 en secuencia de bit en bit, en la ecuación 1 se muestra la expresión matemática de la modulación ASK, mientras que en la ecuación 2 se muestra la expresión matemática para la modulación OOK, donde se observa que solo cambia para el bit 0 la ecuación, generalmente la modulación OOK, es más utilizada que la modulación ASK, debido a que proporciona mejor rendimiento en la trasmisión de información[21].

$$S_{ASK}(t) = \begin{cases} A\cos(2\pi\, f_c t)\,; bit\ 1 \\ B\ \cos(2\pi\, f_c t)\,; bit\ 0 \end{cases} \qquad \text{Ecuación (1)}$$

$$S_{OOK}(t) = \begin{cases} A\cos(2\pi\, f_c t)\,; bit\ 1 \\ 0; bit\ 0 \end{cases} \qquad \text{Ecuación (2)}$$

En la figura 12 se puede observar la modulación de la señal OOK en tiempo vs amplitud

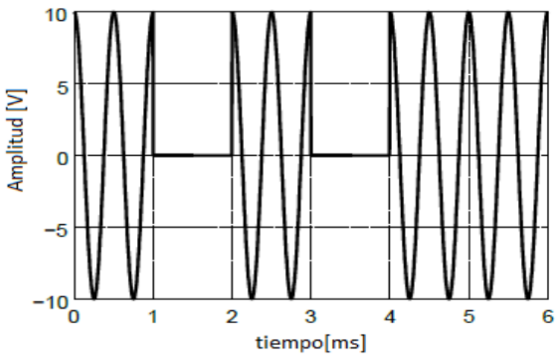


Figura 12. Modulación OOK

Fuente: [21]

3.1.3 Diseño del sistema

3.1.3.1 Configuración del flujo de transmisión y recepción en GNU Radio

3.1.3.1.1 Transmisor ASK

Para el desarrollo del diagrama transmisor se modulación digital ASK se emplearon los bloques que se describen a continuación con sus respectivas configuraciones.

- **Vector Source**

Este vector se encarga de generar varias muestras de bits en un arreglo por un determinado tiempo, pueden ser de tipo float, complex, byte, short, int, el bloque es utilizado para transmitir datos predefinidos, este puede repetirse indefinidamente hasta que el flujo será terminado o seleccionar que sea repetido una sola vez. En la figura 13 se muestran las configuraciones necesarias donde se define datos de tipo byte con el arreglo 10110100, repitiendo los datos hasta que se cierre la ejecución.

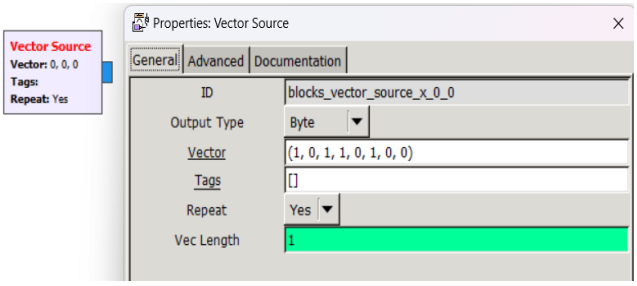


Figura 13. Propiedades del bloque Vector Source

Fuente: Elaborado por autor

- **Signal Source**

Este bloque es un generador de señales con forma de ondas predeterminadas pueden ser seno, coseno, triangulares, ondas cuadradas, mediante su frecuencia, número de muestras enviadas en su periodo y su amplitud, las configuraciones necesarias del bloque se muestran en la figura 14, donde se define el tipo de onda senoidal, tipo de datos float, con una frecuencia de 10 MHz y amplitud de 2.

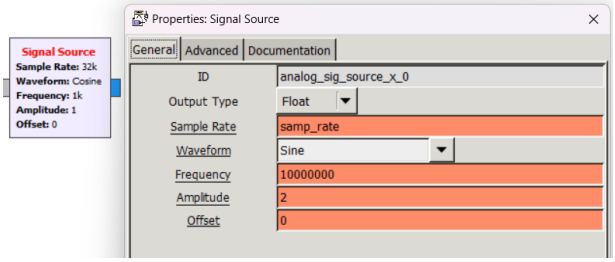


Figura 14. Propiedades del bloque Signal Source

Fuente: Elaborado por autor

- **Repeat**

Este bloque permite repetir la entrada ya sea señal o número de veces que se asigne antes de pasar a la siguiente etapa del procesamiento, es decir actúa como factor de interpolación, se debe tener en cuenta que el bloque aumenta la tasa de muestreo de la señal debido a su función. Las configuraciones realizadas para el desarrollo del diagrama transmisor se muestran en la figura 15, donde se define el tipo de datos en forma de byte, con un factor de repetición de 10 y longitud del vector se establece en 1.

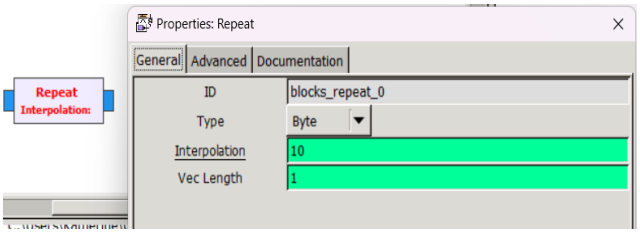


Figura 15. Propiedades del bloque Repeat

Fuente: Elaborado por autor

- **Multiply**

Su función es multiplicar dos o más señales o constantes entre si es decir aplica operación punto a punto, este bloque es esencial en modulaciones de señales, debido a que une frecuencia y procesamiento de señal, para el desarrollo del diagrama se realiza las configuraciones mostradas en la figura 16 en este caso la señal es de tipo float con dos entradas.

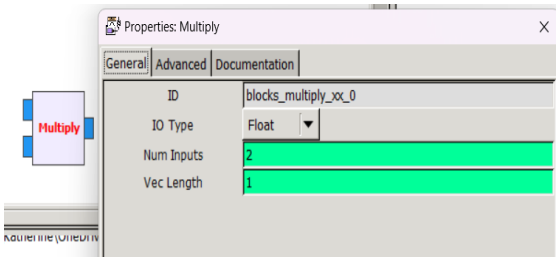


Figura 16. Propiedades del bloque Multiply

Fuente: Elaborado por autor

- **Throttle**

Este bloque se encarga de la velocidad que se procesa los datos, es decir es un bloque de aceleración, evita la sobre carga del sistema y debe ser utilizado solo si el diagrama no tiene un bloque de limitación, en la figura 17 se muestras las configuraciones necesarias del bloque, donde está definido por la variable *samp_rate* con el valor de 32 KHz y es de tipo float.

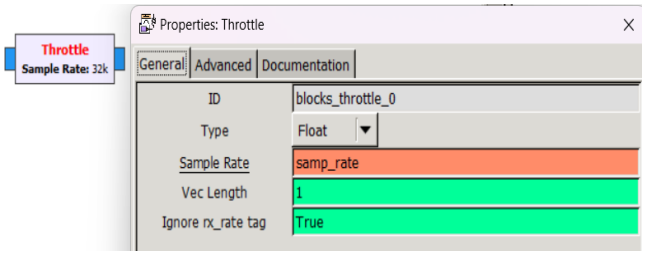


Figura 17. Propiedades del bloque Throttle

Fuente: Elaborado por autor

- **UChart to Float**

Este bloque permite convertir los datos de un tipo a otro específicamente de bytes a float, en la figura 18 se muestra el bloque correspondiente al software empleado.

Figura 18. Bloque UChart to Flaot

Fuente: Elaborado por autor

- **Float to Complex**

El bloque es utilizado para convertir dos entradas float (imaginaria y real) en una señal compleja, la primera entrada es el componente real, mientras la segunda entrada es para el componente imaginario, en la figura 19 se muestra el bloque correspondiente.

Figura 19. Bloque Float to Complex

Fuente: Elaborado por autor

- **WX GUI Scope Sink**

Este bloque permite visualizar las señales en dominio del tiempo, hace la función de un osciloscopio, mostrando la forma de onda de las señales en tiempo real, representando una o varias señales, además tiene la opción para realizar acercamiento, logrando tener una mejor precisión de datos. En la figura 20 se muestra las configuraciones respectivas del bloque para visualizar la señal, se configuro el tipo de datos que llegan al osciloscopio, la posición donde se encuentra la gráfica, en tipo de disparo se coloca en modo auto donde la señal se actualiza automáticamente y la tasa de muestreo que está definida por la variable *samp_rate*.

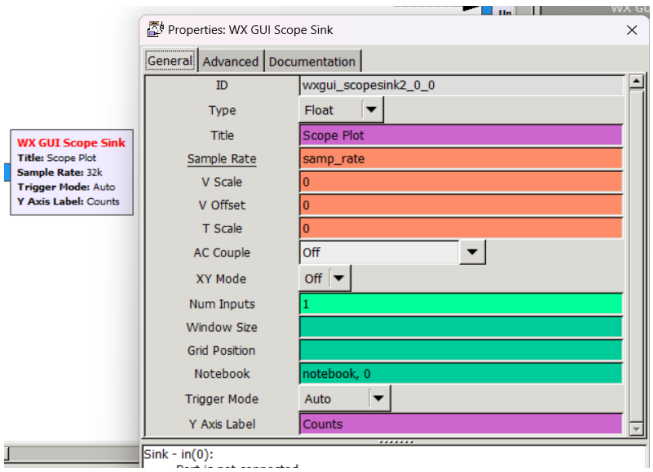


Figura 20. Propiedades del bloque WXGUI Scope Sink

Fuente: Elaborado por autor

- **File Sink**

Este bloque se utiliza para guardar los datos de una señal en forma de archivo, se utiliza para almenar información y posteriormente usarlos para analizar los resultados o para reutilizarlos en otros software, para desarrollar la práctica se guarda los datos para reutilizarlos en el software Matlab y realizar el análisis en ambos software, la ruta determinada es la carpeta donde se guardan los archivos necesarios para el desarrollo de las prácticas, en la figura 21 se muestra las configuraciones necesarias.

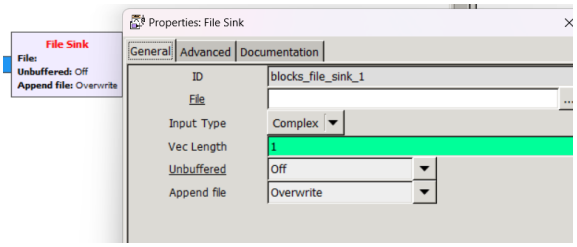


Figura 21.Propiedades del bloque File Sink

Fuente: Elaborado por autor

El esquema para la trasmisión de datos correspondiente a la modulación ASK se muestra en la figura 2, donde se visualiza de manera detalla las conexiones de los bloques antes mencionados.

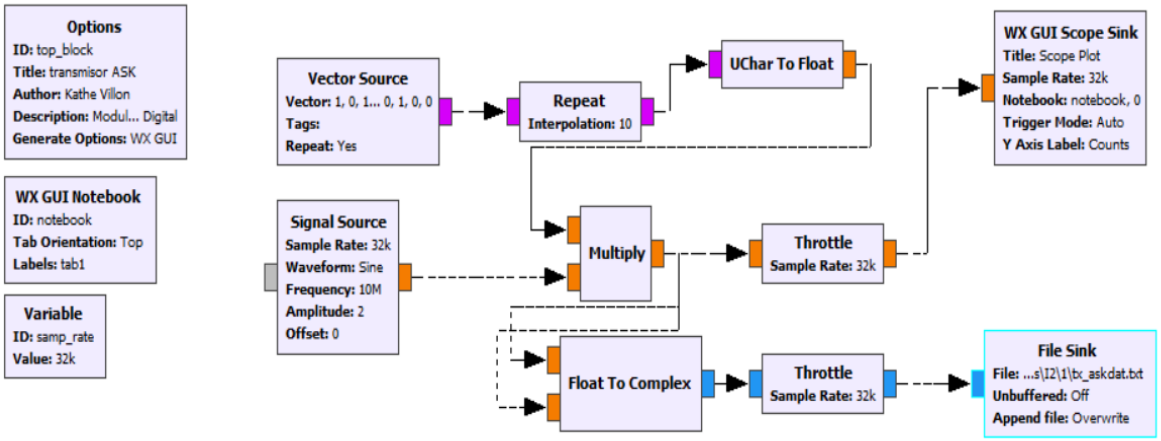


Figura 22. Esquema Transmisor de la señal ASK

Fuente: Elaborado por autor

3.1.3.1.2 Receptor ASK

Para el desarrollo del diagrama receptor de modulación digital ASK se emplearon los siguientes bloques, a continuación, se detalla cada bloque con sus respectivas configuraciones.

- **File Source**

El bloque se utiliza para leer los datos almacenados de un archivo y es utilizado como flujo para el procesamiento de señales, son señales previamente guardadas, en la figura 23 se observa las configuraciones correspondientes del bloque, el tipo de datos que leerá es *complex,* y los datos enviados se van a repetir hasta que la simulación sea terminada.

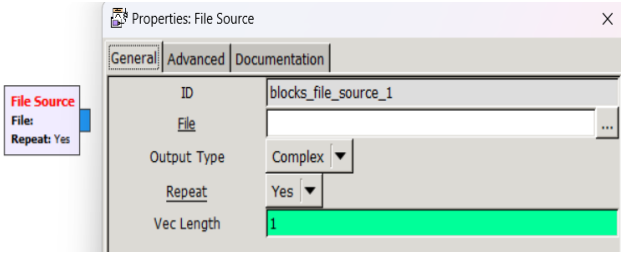


Figura 23. Propiedades del bloque File Source

Fuente: Elaborado por autor

- **Delay**

Este bloque es un retraso en el flujo de datos de una o más señales, compuesta de desfases temporales en el procesamiento de la señal, que se transmite, em la figura 24 se muestra la

configuración para que las señales estén sincronizadas se aplica un delay de 600 y el tipo de datos es complex.

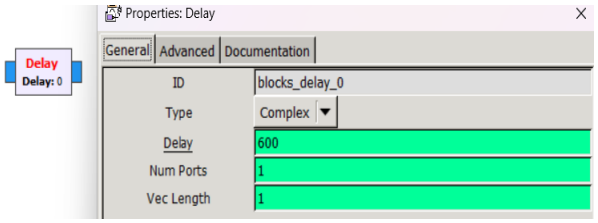


Figura 24. Propiedades del bloque Delay

Fuente: Elaborado por autor

- **Complex to float**

El bloque es utilizado para convertir una señal compleja a dos salidas float (imaginaria y real), la primera salida es para el componente real, mientras la segunda salida es para el componente imaginario, en la figura 25 se muestra el bloque de GNU Radio.

Figura 25. Bloque Complex to Float

Fuente: Elaborado por autor

- **Threshold**

Este bloque se utiliza para comparar la amplitud de una señal de entrada con umbrales, siendo necesario para la recuperación de datos, para el desarrollo del diagrama receptor en la figura 26 se muestra las configuraciones respectivas, estableciendo tanto el umbral inferior como el superior en 500u.

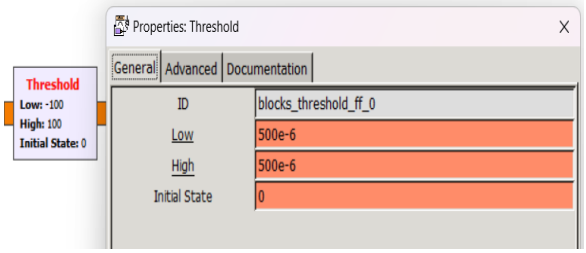


Figura 26. Propiedades del bloque Threshold

Fuente: Elaborado por autor

- **Throttle**

Es el bloque que se encarga de copiar los elementos de entrada a la salida, es la velocidad con la que se procesan los datos, es decir que es un bloque de aceleración, evitando sobrecarga del sistema, en la figura 27 se observa las configuraciones necesarias el cual está definido por la variable *samp_rate* con el valor de 32KHz y de tipo float.

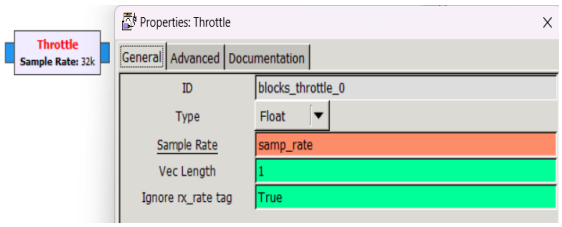


Figura 27. Propiedades del bloque Throttle

Fuente: Elaborado por autor

- **WX GUI Scope Sink**

Este bloque permite visualizar las señales en dominio del tiempo, hace la función de un osciloscopio, mostrando la forma de onda de las señales en tiempo real, representando una o varias señales, además tiene la opción para realizar acercamiento, logrando tener una mejor precisión de datos. En la figura 28 se muestra las configuraciones respectivas del bloque para visualizar la señal, se configuro el tipo de datos que llegan al osciloscopio, la posición donde se encuentra la gráfica, en tipo de disparo se coloca en modo auto donde la señal se actualiza automáticamente y la tasa de muestreo que está definida por la variable *samp_rate*.

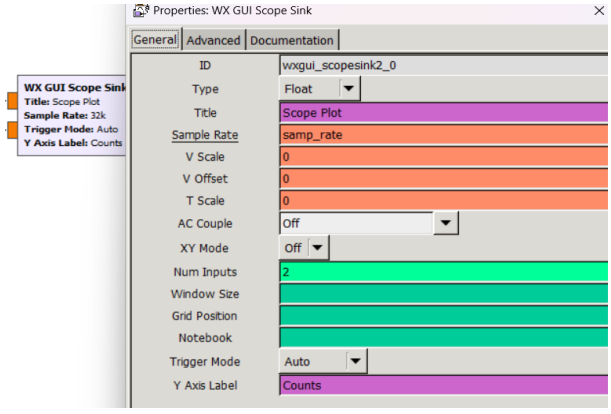


Figura 28. Propiedades del bloque WX GUI Scope Sink

Fuente: Elaborado por autor

El esquema final para la recepción de datos de modulación ASK se muestra en la figura 29, donde se puede visualizar de manera detallada las conexiones de cada bloque antes mencionado con sus respectivas configuraciones.

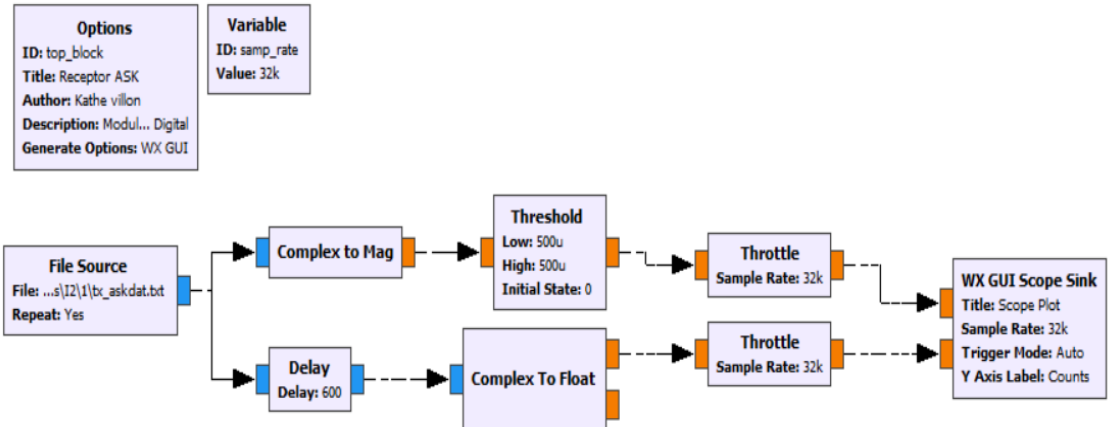


Figura 29. Esquema receptor de la señal ASK

Fuente: Elaborado por autor

3.1.4 Implementación

En la primera parte de la práctica, se utilizó los bloques *file soruce* y *file sink* para la simulación de transmisión y recepción de datos, sin necesidad de un hardware físico, sin embargo, el objetivo es la implementación realista, en esta sección se agregan los bloques *Osmocom Sink* para la transmisión y *OsmocomSource* para la etapa de recepción.

3.1.4.1 Transmisor ASK

El bloque *Osmocom Sink* es utilizado para transmitir datos hacia el receptor, en la figura 30, se muestra las configuraciones necesarias para la trasmisión de datos en modulación ASK, para esto se realiza la configuración en la frecuencia que se desea trasmitir los datos, este caso se utiliza la frecuencia de 902 MHz, el tipo de datos que se visualiza es de tipo *complex float 32*, y se debe tener en cuenta que para la transmisión y recepción de datos deben estar en la misma frecuencia para recibir adecuadamente los datos enviados.

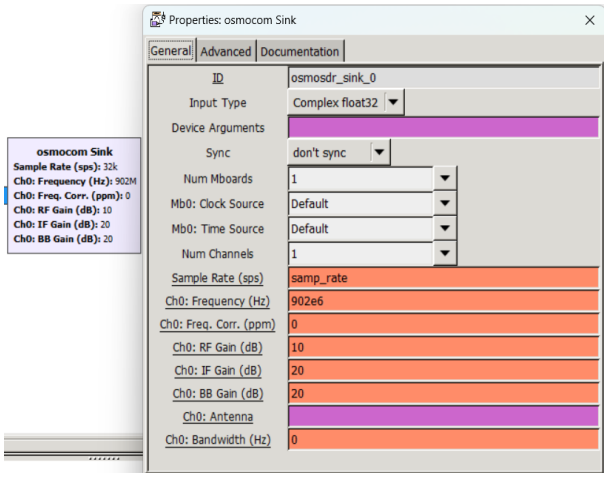


Figura 30. Propiedades del bloque Osmocom Sink

Fuente: Elaborado por autor

En la figura 31 se muestra el esquema final de la modulación ASK implementado el bloque *Osmocom Sink*, que permite la transmisión de datos mediante el HackRF One.

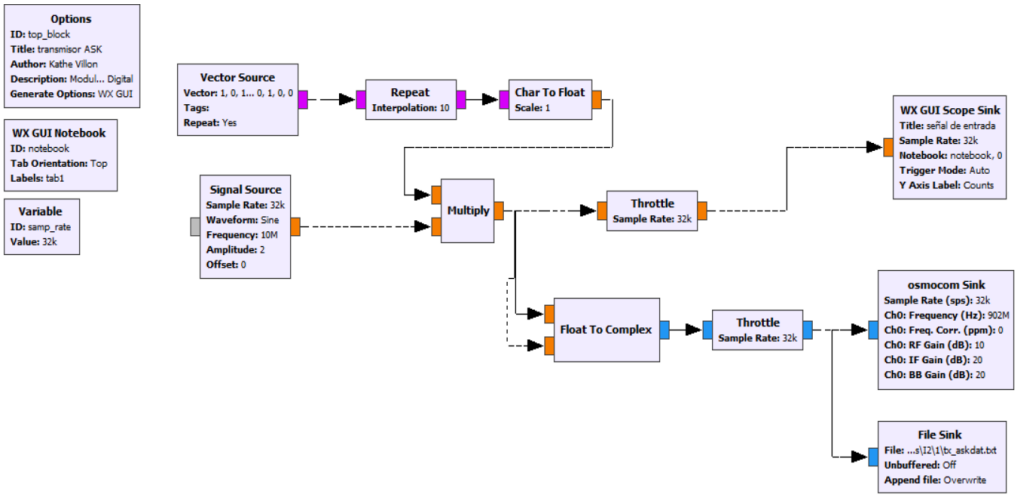


Figura 31. Esquema final transmisor de señal ASK

Fuente: Elaborado por autor

3.1.4.2 Receptor ASK

El bloque *Osmocom Source* es utilizado para recibir señales o datos desde un trasmisor, los parámetros con sus respectivas configuraciones se detallan en la figura 32 para la recepción de datos de la modulación ASK.

Para la recepción de los datos se configura con la misma frecuencia del transmisor en 902 MHz para poder recibir los datos de manera correcta y evitar errores en las simulaciones, a diferencia del bloque *Osmocom Sink* los datos ya no ingresan al bloque, al contrario, salen, pero con el mismo tipo de datos *Complex Float 32*.

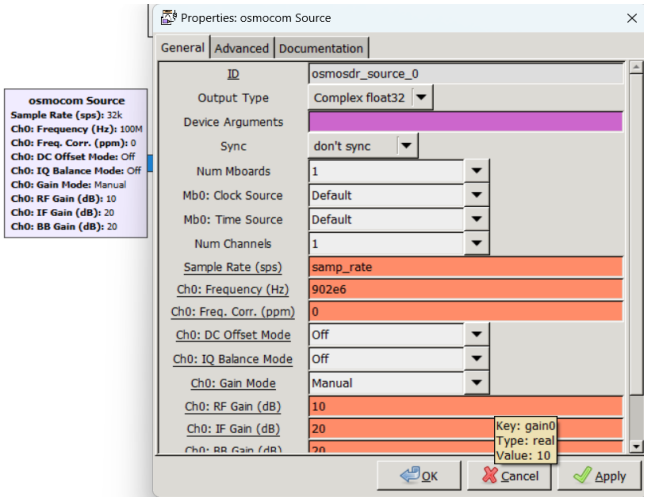


Figura 32. Propiedades del bloque Osmocom Source

Fuente: Elaborado por autor

El esquema final se muestra en la figura 33, donde el bloque *file source* es remplazado por el bloque *Osmocom Source* el cual permite recibir datos que envía el trasmisor.

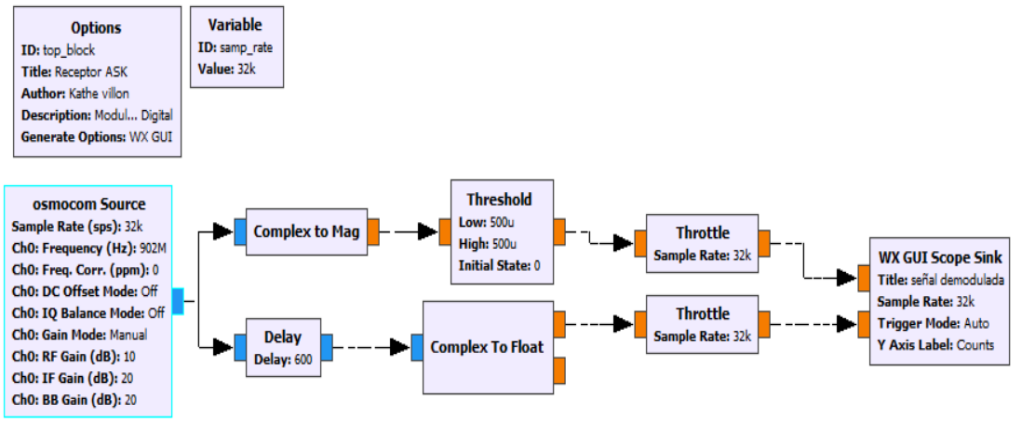


Figura 33. Esquema final receptor de la señal ASK

Fuente: Elaborado por autor

3.1.5 Simulación y Verificación

Uso de Matlab para simular y verificar el diseño

3.1.5.1 Configuración de transmisión y recepción en Matlab

3.1.5.1.1 Transmisor ASK en Matlab

Para desarrollar la trasmisión en ASK y la verificación de datos en GNU Radio, se implementaron en Matlab las líneas de código que se muestran en el anexo 1, a continuación, se detallan las líneas de código las cuales fueron remplazadas por los bloques de GNU Radio correspondientes, junto con una pequeña explicación de cada segmento del código.

- En GNU Radio *Vector Source* por la línea *data = [10110100*] en Matlab

En GNU Radio, el bloque "Vector Source" es utilizado para definir los datos binarios a transmitir, mientras que en Matlab la misma tarea es realizada por el vector "data", que contiene datos binarios 0 y 1 que se desea modular y transmitir, este vector representa la información en bits que luego será modulada por la señal portadora.

- En GNU Radio *Signal Source* por la línea *f_carrirer = 1e6* en Matlab

En GNU Radio, el bloque "Signal Source" genera la señal portadora con la frecuencia deseada, en Matlab esta señal portadora está definida por *f_carrier* y *samp_rate*, que representa la frecuencia portadora y la tasa de muestreo, respectivamente, la frecuencia de la portadora determina la base sobre la cual se modulan los datos binarios.

- En GNU Radio *Repeat* por la línea *data_repeated = repelem (data, interpolation_factor)* en Matlab

El bloque "Repeat" en GNU Radio se utiliza para ampliar los bits a través del factor de interpolación, en Matlab la función *repelem* realiza esta operación de repetir cada valor del vector data por el *interpolation_factor*, la interpolación es importante para la sincronización de la velocidad de los datos con la frecuencia de la señal portadora.

- En GNU Radio *Chart to float* por la línea *data_float (data_repeated)* en Matlab.

En GNU Radio, el bloque "Chart to float" convierte los datos de formato binario a un formato punto flotante, mientras que en Matlab se logra usando double(data_repeated), convierte los

valores binarios a un formato punto flotante para que se puedan multiplicar con la portadora, esta conversión permite manipular los datos en operaciones matemáticas para la modulación de los datos.

- En GNU Radio *Multiply* por la línea *Modulated_signal = data_float. * carrier* en Matlab

El bloque "Multiply" en GNU Radio realiza la modulación ASK al multiplicar los datos con la señal portadora, en Matlab se logra mediante la lineade código *moodulated_signal = data_float*carrier*, donde la amplitud de la portadora varía de acuerdo con el valor *data_float* 1 o 0, la multiplicación de la portadora con los binarios genera modulación ASK donde los 1 mantienen la amplitud y los 0 apaga la señal.

- En GNU Radio *File Sink* por la línea *fileID = fopen ('modulated_ask_data.txt', 'w'); fclose (fileID)* en Matlab

En GNU Radio, file sink guarda la señal modulada en un archivo, mientras que, en Matlab, se hace mediante *fopen* y *fprinft*, que se encargan de escribir los valores de *moulated_signal* en un archivo .txt, y guardar la señal modulada para cargarla en la recepción y demodulación.

- En GNU Radio *WX GUI Scope Sink* por la línea *figure, subplot* en Matlab

En GNU Radio, el bloque "WX GUI Scope Sink", permite visualizar la señal modulada en tiempo real y en Matlab mediante *subplot* y *plot* se muestra la representación binaria de los datos transmitido, para verificar el proceso de la modulación.

3.1.5.1.2 Receptor ASK en Matlab

Para el desarrollo de la recepción de la señal modulada ASK y verificar los datos de GNU Radio, se desarrolló las líneas de código mostradas en el anexo 2. En esta sección, se detallan los bloques de recepción de GNU Radio que fueron convertidos en líneas de código.

- En GNU *Radio File Source* por la línea *load ('modulated_ask_data.txt')* en Matlab

Carga los datos modulados desde un archivo en ambos casos GNU Radio para leer los datos de un archivo y Matlab mediante load.

- En GNU Radio *Complex to Mag* por la línea received_magnitude = abs (received_signal) en Matlab

Convierte la señal compleja en su magnitud en MATLAB mediante el comando *abs*.

- En GNU Radio *Threshold* por la línea *threshold = max (received_magnitude)/2; demodulated_data = received_magnitude* en Matlab

Establece un umbral para demodular la señal, calculando la mistad del valor máximo de la magnitud para identificar 0 y 1.

- En GNU Radio *WX GUI Scope Sink* por la línea plot (t* 1e3, received_signal) en Matlab

Visualiza la señal demodulada para ver la amplitud en función del tiempo.

3.1.6 Evaluación y Análisis

En esta sección se presenta los resultados obtenidos de la transmisión y recepción de un sistema de comunicación básica utilizando modulación ASK.

3.1.6.1 Resultados en GNU Radio

En la figura 34 se muestra la señal transmitida de los datos binarios 10110100 mediante el vector Source, la amplitud de la señal oscila entre 2 y -2, indicando valores alternando entre niveles de bits altos 1 y niveles bajos 0, la frecuencia es de 10 MHz, permitiendo observar diferentes estados de la señal.

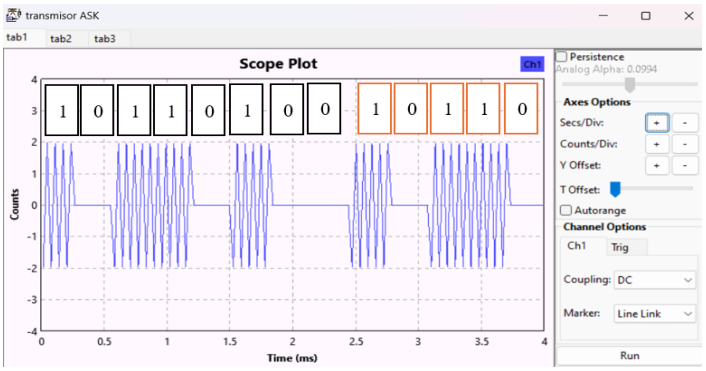


Figura 34. Resultados del transmisor ASK en GNU Radio

Fuente: Elaborado por autor

En la figura 35 se observa la señal recibida donde la amplitud disminuye a uno y la señal esta invertida a diferencia de la señal trasmitida, sin embargo, los bits se mantienen en el orden adecuado para la transmisión.

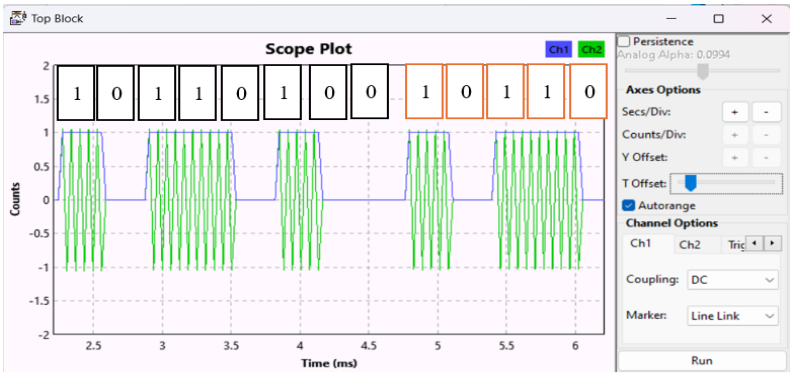


Figura 35, Resultados del receptor de señal ASK en GNU Radio

Fuente: Elaborado por autor

Transmisor ASK

En la figura 36 se muestra la señal transmitida de los valores binarios 10110100 ingresados de forma manual en el software, la amplitud de la señal varia de 0 a 1, donde los niveles altos se representan por 1 y los bajos por 0 utilizando la frecuencia portadora 10 MHz como en GNU Radio.

En el gráfico de la izquierda correspondiente a “Datos Binarios Transmitidos”, se observan los valores binarios correspondientes a la secuencia binaria ingresada. Por otro lado, en el gráfico de la derecha corresponden a la “transmisión de la señal ASK” donde se muestra la señal modulada en amplitud, permitiendo visualizar los diferentes estados de la señal en función del mensaje binario de entrada.

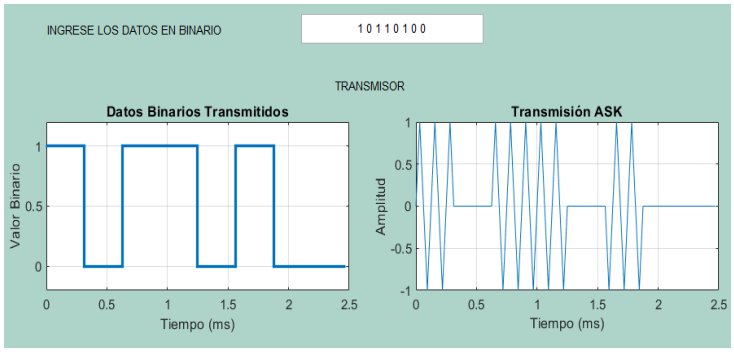


Figura 36. Resultados del transmisor de la señal ASK en Matlab

Fuente: Elaborado por autor

Receptor ASK

En la figura 37, se observa la señal recibida en Matlab, donde se puede evidenciar cierta diferencia a comparación de la señal recibida en GNU Radio que presentaba inversión de la señal donde los niveles altos están definidos por 1 y los niveles bajos por 0, manteniéndose estables y permitiendo una correcta identificación de cada bit.

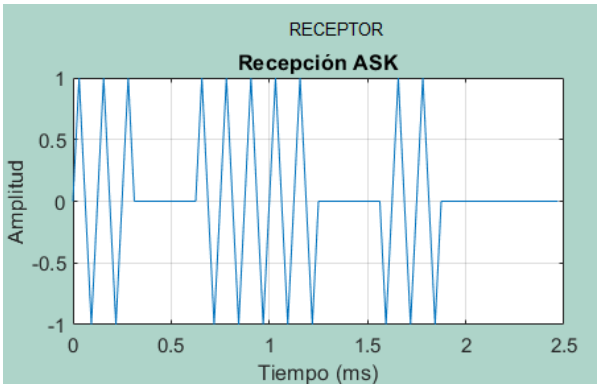


Figura 37. Resultados del receptor de señal ASK en Matlab

Fuente: Elaborado por autor

Hipótesis: El uso combinado de GNU Radio y Matlab permite el diseño y verificación más robusto de los sistemas de comunicación simple.

3.1.7 Conclusión de la práctica

Mediante la práctica se logró demostrar el diseño y simulación de un sistema de comunicación simple utilizando modulación ASK mediante los softwares empleados, donde se implementó el transmisor y receptor, teniendo en cuenta diferentes aspectos en la transmisión y recepción de la señal.

En el software GNU Radio, el sistema, permitió transmitir y recibir datos binarios utilizando bloques de procesamiento de señal, resaltando la capacidad del entorno en integrar varias fuentes de datos estableciendo umbrales y visualizar los datos en tiempo real, el sistema mostró diferentes variaciones de la señal debido a la características propias del sistema, como cambios de amplitud en función de los valores establecido, las herramientas de visualización permitieron observar en la aparte de la recepción la inversión ante mencionadas.

Por otro lado, Matlab facilito el diseño de un sistema equivalente, donde el procesamiento de datos binarios y la modulación ASK, se implementaron directamente en código, permitió la verificación

del proceso de modulación con gráficos que demuestran la relación entre los bits transmitidos y la señal modulada, y la señal recibida mantuvo su amplitud sin inversión en su fase, evidenciando que para esta práctica resultó más eficiente el software Matlab.

Ambos entornos, proporcionaron una experiencia de aprendizaje complementaria, GNU Radio es una implementación rápida y en tiempo real, mientras que Matlab, permitió un control más preciso, sobre el procesamiento de la señal.

3.2 Práctica 2.- Implementación y Evaluación de Esquemas de Modulación Digital

3.2.1 Descripción de la importancia de la modulación digital

La comunicación digital es utilizada para transmitir señales digitales a través de un canal de comunicación qué consiste, en realizar cambios en la portada puede ser amplitud, frecuencia, fase, para enviar datos, mediante técnicas de modulación digital que han aportado de manera significativa en las comunicaciones inalámbricas, incrementando seguridad en datos, alta velocidad de transmisión, mayor capacidad de información, calidad de las comunicaciones[22].

En las modulaciones digitales se puede mencionar la modulación por desplazamiento de fase PSK qué se subdivide en dos principales como es por fase binaria BPSK y fase cuaternaria QPSK, ambas utilizadas en redes inalámbricas; WLAN *Wireless Local Area Network*, WIMAX *Worldwide Interoperability for Microwave Access*, LTE *Long Term Evolution*, para la sincronización de las señales o trasmisión de datos[23].

La modulación BPSK es una de las formas más fáciles de modulación en fase con alto nivel de ruido, utiliza dos fases distintas para representar los estados de un bit, mientras que QPSK permite transmitir dos bits por símbolo, duplicando la tasa de transmisión y utilizando cuatro fases distintas[24].

Objetivo de la práctica: Implementar y evaluar diferentes esquemas de modulación digital BPSK, QPSK, utilizando HackRF One, GNU Radio y Matlab.

3.2.2 Revisión Teórica

Estudio de los esquemas de modulación digital

Los esquemas de modulación digital son esenciales en los sistemas de comunicación moderna, permitiendo la transmisión de información digital en forma eficiente mediante medio analógicos.

Modulación BPSK - *Binary Phase Shift Keying*

La modulación por desplazamiento de fase binaria es un esquema donde se envía un mensaje y se visualiza dos cambios de fase en 0º y 180º variando un valor positivo 1 y un valor negativo 0, que son convertidos en NRZ *Not Return to Zero,* tiene un índice de error bajo [25] .En la ecuación 3 se muestra la expresión matemática de la modulación M-PSK.

$$S_m\,(t)\;=\;g(t)\,cos\,(2\pi f_c\,t+\;\vartheta_m) \qquad \text{Ecuación (3)}$$

Datos

$$M\;=\;2$$

$$m\;=\;secuencia$$

Donde $\vartheta_m = 2\pi\left(\frac{m-1}{M}\right)$

$\vartheta_m = 2\pi\left(\frac{1-1}{2}\right) = \;0$

$\vartheta_m = 2\pi\left(\frac{2-1}{2}\right) = \pi$

En la tabla 4 se muestran las transiciones de estado de acuerdo los bits de información y grados que se representa.

Bits de información	En grados
0	180
1	0

Tabla 4. Transición de estado de la información de bits en BPSK

Fuente: Elaborado por autor

El diagrama de constelación para BPSK se emplean con dos símbolos, un bit de información para cada uno como se muestra en la figura 38.

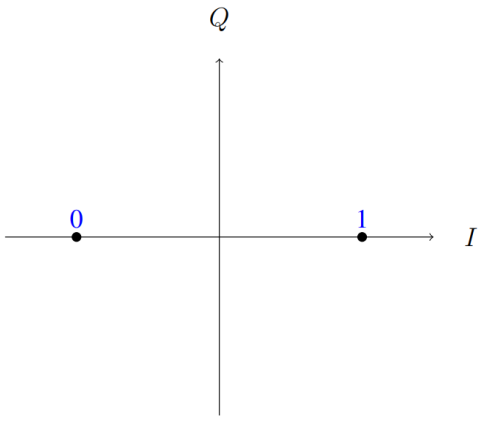


Figura 38. Diagrama de constelación BPSK

Fuente: [26]

Modulación QPSK - *Quadrature Phase Shift Keying*

La modulación por desplazamiento de fase cuaternaria QPSK consiste en variar la fase por medio de una señal digital, esta modulación de la fase tiene cuatro cambios de 0, 90, 180 y 270 grados en una sola frecuencia portadora, cada fase se codifica en dos bits agrupándolos en un símbolo mediante código Gray 11, 01, 00, 10, el ancho de banda no depende de la magnitud de la alteración y es la mitad de la modulación BPSK[25].

En la ecuación 3 se muestra la expresión matemática de la modulación M-PSK.

$$S_m\,(t)\ =\ g(t)\,cos\,(2\pi f_c\,t+\ \vartheta_m) \qquad \text{Ecuación (4)}$$

Datos

$$M\ =\ 2$$

$$m\ =\ secuencia$$

$\vartheta_m = 2\pi\left(\frac{1-1}{4}\right) = 0$

$\vartheta_m = 2\pi\left(\frac{2-1}{4}\right) = \frac{\pi}{2}$

$\vartheta_m = 2\pi\left(\frac{3-1}{4}\right) = \pi$

$\vartheta_m = 2\pi\left(\frac{4-1}{4}\right) = \frac{3\pi}{2}$

En la tabla 5 se muestran las transiciones de estado de acuerdo los bits de información y grados que se representa.

Bits de información	En grados
00	45
01	135
11	255
10	325

Tabla 5. Transición de estado de la información de bits en QPSK

Fuente: Elaborado por autor

El diagrama de constelación para QPSK se emplea con cuatro símbolos, dos bits de información para cada uno y se desplaza entre sí en 90º como se muestra en la figura 39.

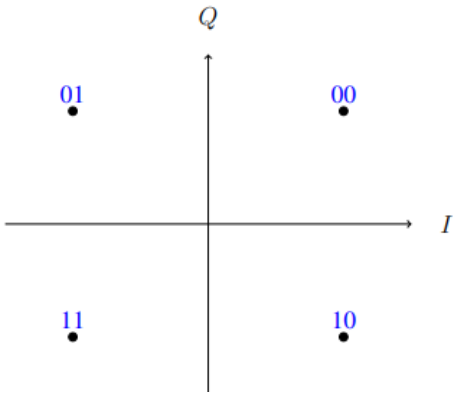


Figura 39. Diagrama de constelación QPSK

Fuente: [26]

BER - *Bit Error Rate*

Es una medida que se utiliza para verificar la calidad de la transmisión de datos se define cómo la porción de bits transmitidos que se recibe de manera incorrecta en comparación con los números de bits[26], se calcula mediante la ecuación 4.

$$BER = \frac{Numero\ de\ bits\ erroneos}{Numero\ total\ de\ bits\ trasnmitidos} \qquad \text{Ecuación (5)}$$

3.2.3 Diseño del sistema

Implementación de BPSK, QPSK en GNU Radio

3.2.3.1 Configuración del flujo de transmisión y recepción en GNU Radio

3.2.3.1.1 Transmisor BPSK

El desarrollo del diagrama transmisor de modulación digital BPSK se emplearon los siguientes bloques, a continuación, se detalla cada bloque con sus respectivas configuraciones.

- **Random Uniform Source**

Este bloque genera los datos que se van a transmitir en secuencias de 1 y -1 los cuales se interpretan como bits 0 y 1. Para el desarrollo de la práctica se realizó las configuraciones que se muestra en la figura 40, donde se establece como valor inicial 0, el valor mínimo es 1 y el valor máximo de -1, los datos que se generan son de tipo de short.

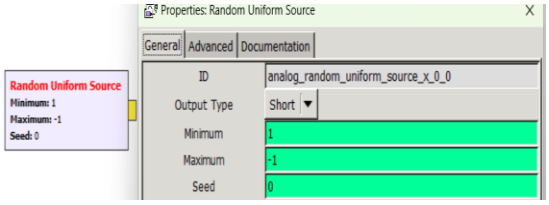


Figura 40. Propiedades del bloque Random Uniform Source

Fuente: Elaborado por autor

- **Short to float**

Este bloqueo permite convertir un tipo de datos a otro tipo de datos específicamente de short a float, en la figura 41 se muestran los parámetros del bloque y los valores establecidos son por defecto del software.

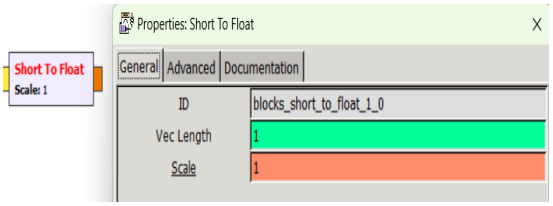


Figura 41. Propiedades del bloque Short to Float

Fuente: Elaborado por autor

- **Throttle**

Este bloque se encarga de copiar todos los elementos en la entrada a la salida, es la velocidad en la que se va a procesar los datos es decir un bloque de aceleración, evita sobrecargas del sistema en la figura 42 se observan las configuraciones necesarias del bloque utilizado para el diagrama transmisor lo cual está definido por la variable *samp_rate* con el valor de 2048 KHz y es un tipo de short.

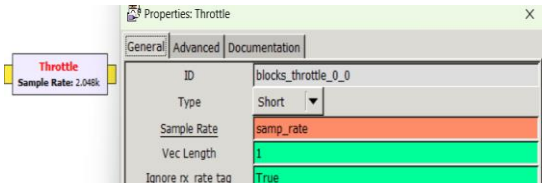


Figura 42. Propiedades del bloque Throttle

Fuente: Elaborado por autor

- **Packed to Unpacked**

El bloque convierte los datos empaquetados en bits individuales toma una secuencia de datos y en cada byte contiene varios bits de información generalmente 8 bits forma un byte y los descomponen en bits separados en la figura 43 se detallan las configuraciones realizadas para el desarrollo de la transmisión de la señal y establece los tipos de datos short, un valor de 1 para *bits per chunck.*

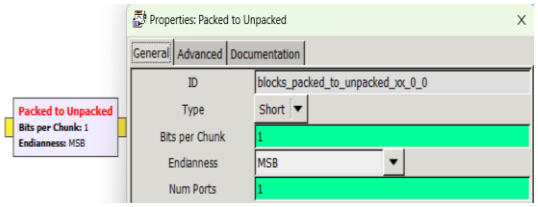


Figura 43. Propiedades del bloque Packed to Unpacked

Fuente: Elaborado por autor

- **File sink**

Este bloque se utiliza para guardar los datos de una señal en forma de archivo, es para almacenar información y posteriormente utilizarlos para analizar los resultados o para reutilizarlos en otro software para el desarrollo de esta práctica serán reutilizados en el software Matlab y realizará un análisis en ambos, la ruta predeterminada es la carpeta donde se guardan los archivos necesarios

para el desarrollo de las prácticas en la figura 44 se muestran las configuraciones necesarias además el tipo de datos guardados es de tipo float con una longitud de un elemento a cada ciclo.

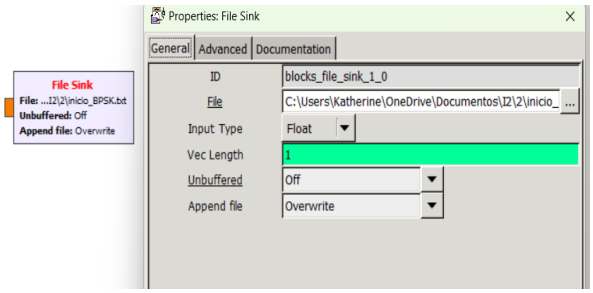


Figura 44. Propiedades del bloque File Sink

Fuente: Elaborado por autor

- **Float to Complex**

El bloque convierte una señal compleja con dos entradas float imaginario y real en una salida compleja la primera entrada es el componente real mientras que la segunda entrada corresponde el componente imaginario, en la figura 45 se muestra el bloque correspondiente en el software.

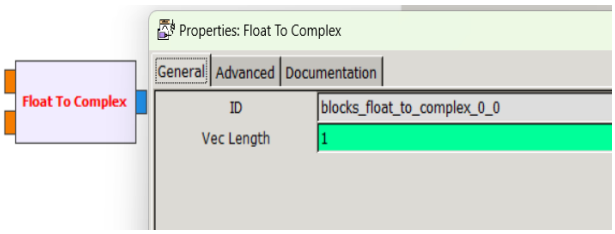


Figura 45. Propiedades del bloque Float to Complex

Fuente: Elaborado por autor

- **Virtual Sink**

Este bloque es utilizado para interconectar el diagrama del transmisor y receptor en una sola ventana de trabajo generalmente se utiliza cuando no se cuenta con un dispositivo hardware, hace la función del canal de comunicación dentro del mismo software en la figura 46 se muestra el bloque utilizado en la transmisión de datos.

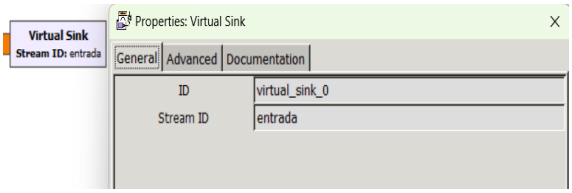


Figura 46. Propiedades del bloque Virtual Sink

Fuente: Elaborado por autor

- **QT GUI Constellation:**

Ese bloque me permite visualizar en un plano complejo la representación gráfica de la constelación de la señal modulada, en la figura 47 se muestran las configuraciones de los parámetros necesarios para el desarrollo de la práctica y establece los tipos de datos complejos con un número de puntos 1024 en X y Y el mínimo se establece -2 mientras que en X y Y el máximo se establece 2.

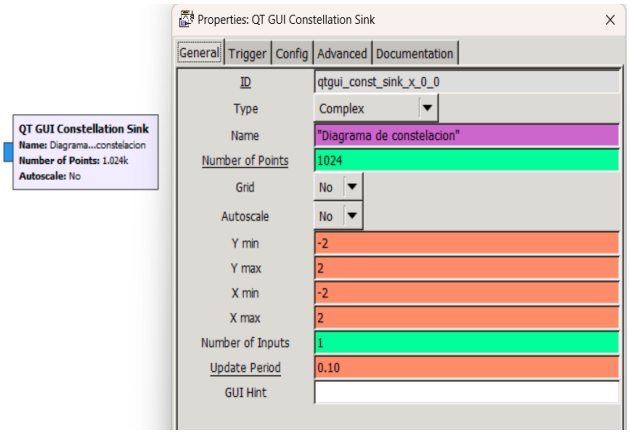


Figura 47. Propiedades del bloque QT GUI Constellation Sink

Fuente: Elaborado por autor

- **QT GUI Time Sink**

Este bloque permite visualizar la señal recibida, realiza la función de un oscilador mostrando cómo varía la señal a largo del tiempo, en la figura 48 se muestran las configuraciones necesarias de bloque para el desarrollo de la modulación BPSK, se establece el tipo de datos como float el número de puntos a presentarse es de 1024 y la tasa de muestra está definida por la variable *samp_rate* con un valor de 2048 KHz.

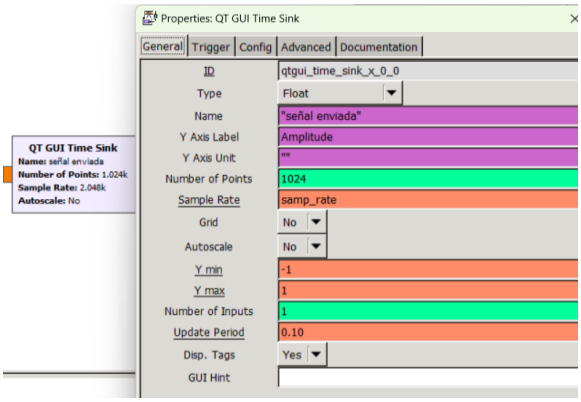


Figura 48. Propiedades del bloque QT GUI Time Sink

Fuente: Elaborado por autor

El esquema para la transmisión de modulación BPSK se muestra en la figura 49 permitiendo desarrollar el envío de datos hacia el rector, donde se puede visualizar de manera detallada las conexiones en los bloques antes mencionados con sus respectivas configuraciones.

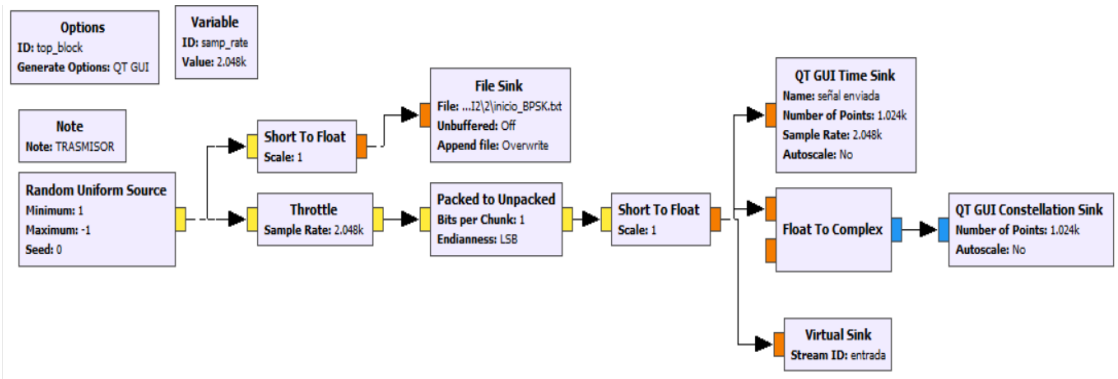


Figura 49. Diagrama trasmisor para la modulación BPSK

Fuente: Elaborado por autor

3.2.3.1.2 Receptor BPSK

- **Virtual Source**

Este bloque sirve como punto de conexión para recibir la señal del transmisor, es utilizado cuando se necesita interconectar sistemas, en la figura 50 se muestra en el bloque para el desarrollo del receptor como el canal de transmisión en un caso ideal sin ruidos ni interferencias debido a que se desarrolla internamente en el software.

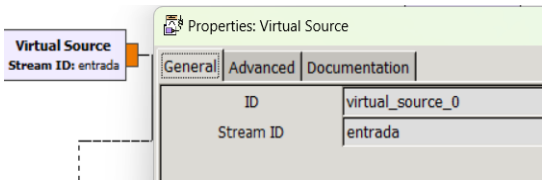


Figura 50. Propiedades del bloque Virtual Source

Fuente: Elaborado por autor

- **Repeat**

Este bloque permite repetir la señal un número determinado de veces el parámetro de interpolación de bloque aumenta la tasa de muestreo de la señal, en la figura 51 se muestra la configuración, que establece una interpolación de 128 y los tipos de datos que se procesa este tipo float.

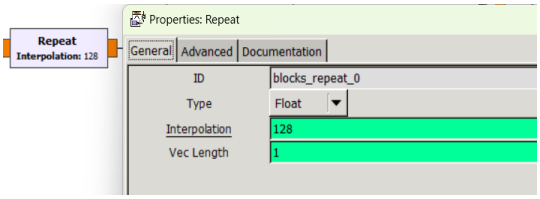


Figura 51. Propiedades del bloque Repeat

Fuente: Elaborado por autor

- **Add Const**

El bloque suma una constante a la señal permitiendo ajustar el nivel, asegurando que las señales sean procesadas adecuadamente sin perder ninguna información, en la figura 52 se muestran las configuraciones establecidas en el parámetro *constant* es de -500m el tipo de datos procesados es de tipo float.

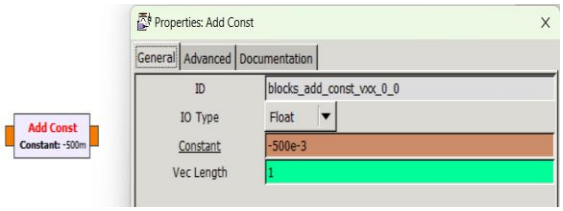


Figura 52 . Propiedades del bloque Add Const

Fuente: Elaborado por autor

- **Multiply**

Eso es lo que permite mezclar la señal recibida con la señal de la portadora local en la figura 53 se puede observar las configuraciones respectivas del bloque estableciendo dos entradas y datos de tipo float.

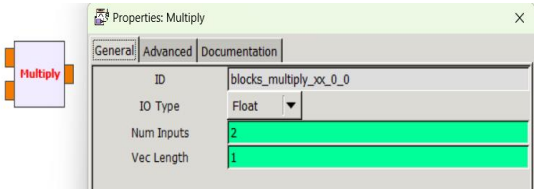


Figura 53. Propiedades del bloque Multiply

Fuente: Elaborada por autor

- **Signal Source**

Esto es lo que es un generador de señal con forma de onda predeterminada pueden ser seno, coseno, triangulares u ondas cuadradas mediante su frecuencia, número de muestras enviadas en su periodo y amplitud las configuraciones necesarias para el diagrama de bloque a realizarse observan en la figura 54 definiendo el tipo de ondas sinusoidal, datos generados de tipo float y se establece una frecuencia de 1 y la amplitud de - 2.

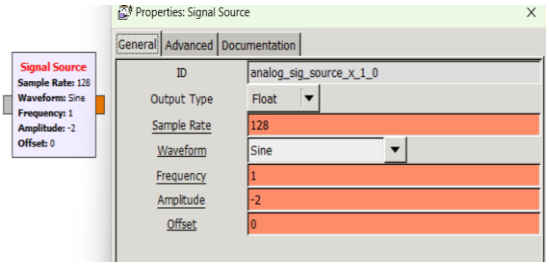


Figura 54. Propiedades del bloque Signal Source

Fuente: Elaborado por autor

- **QT GUI Time sink**

Ese bloque permite visualizar la señal recibida realiza la función de un oscilador, mostrando cómo varía la señal a largo del tiempo, en la figura 55 se muestra las configuraciones necesarias y establece el tipo de datos float en la entrada del bloque con 2048 puntos y la tasa de muestras definida por la variable *samp_rate* con el valor de 2048.

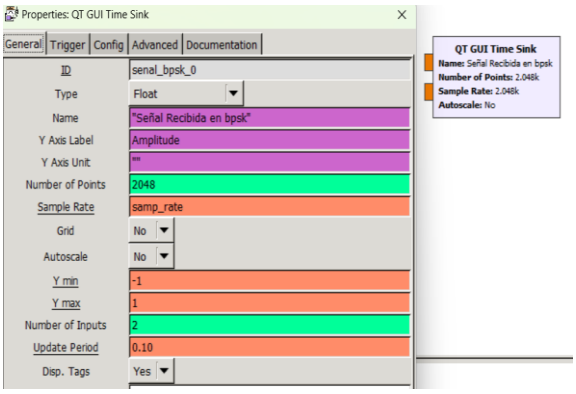


Figura 55. Propiedades del bloque QT GUI Time Sink

Fuente: Elaborado por autor

- **File sink**

Este bloque se utiliza para guardar los datos de una señal en forma de archivo, es para almacenar información y posteriormente utilizarlos para analizar los resultados o para reutilizarlos en otro software para el desarrollo de esta práctica serán reutilizados en el software Matlab y realizará un análisis en ambos, la ruta predeterminada es la carpeta donde se guardan los archivos necesarios para el desarrollo de las prácticas en la figura 56 se muestran las configuraciones necesarias además el tipo de datos guardados es de tipo float con una longitud de un elemento a cada ciclo.

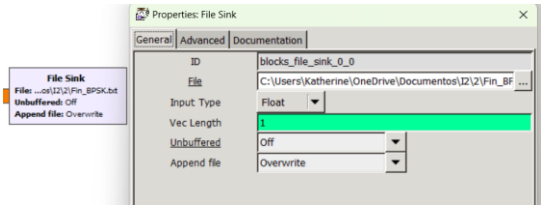


Figura 56. Propiedades del bloque File Sink

Fuente: Elaborado por autor

En la figura 57 se puede observar el esquema receptor para la modulación BPSK con sus respectivas conexiones de los bloques mencionados anteriormente.

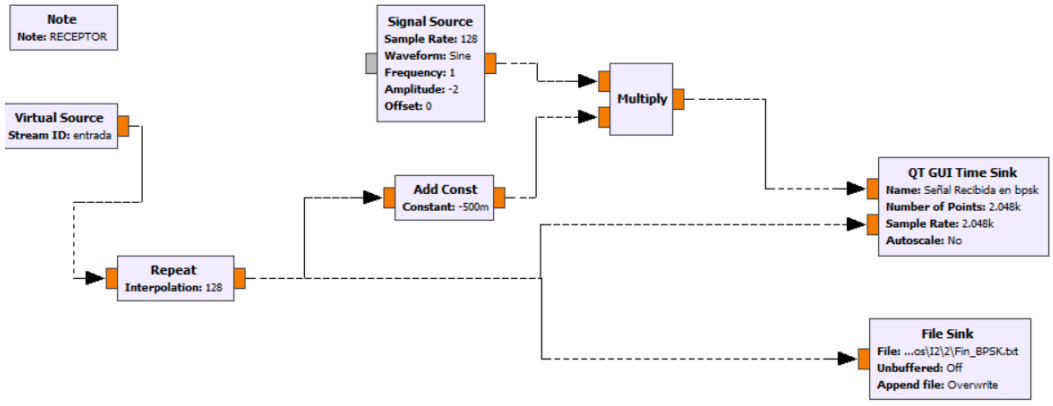


Figura 57. Diagrama receptor para la modulación BPSK

Fuente: Elaborado por autor

3.2.3.1.3 Transmisor QPSK

Pero el desarrollo del diagrama correspondiente a la modulación digital QPSK se emplearon los siguientes bloques, a continuación, se detalla cada bloque con sus respectivas configuraciones.

- **File Source**

Eso es lo que permite leer datos desde un archivo y enviarlos en forma de bytes en secuencia para el desarrollo de la práctica se utiliza un archivo .txt para enviar hacia el receptor permitiendo visualizar las constelaciones en este caso se decide probar que no solo con datos *random* se puede realizar las modulaciones como en la modulación BPSK, en la figura 58 se pueden observar las configuraciones necesarias del bloque.

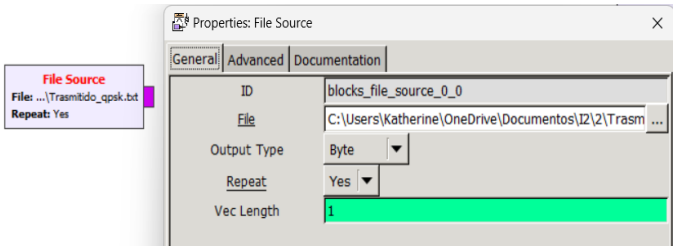


Figura 58. Propiedades del bloque File Source

Fuente: Elaborado por autor

- **Constellation Modulator**

El bloque se encarga de modular la entrada de datos en forma de vida utilizando una modulación predeterminada como puede ser BPSK, QPSK, QAM entre otras para el desarrollo de la práctica se realiza la modulación QPSK llamando al objeto de constelación por medio de la variable QPSK definida dos números de muestras por símbolos y el factor del ancho de banda es de 0.35 para la modulación.

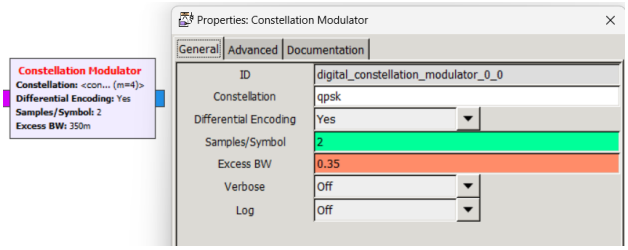


Figura 59. Propiedades del bloque Constellation Modulator

Fuente: Elaborado por autor

- **Constellation object**

Ese bloque define el tipo de constelación a realizar es decir los puntos complejos en el plano como los mapea es definido como variable para permitir ser llamado dentro de otro bloque como se mencionó antes este bloque llamado en el bloque *constellation modulator* por medio de la variable independiente QPSK se establecen los puntos complejos que forman la constelación observa en la figura 39 es para esto los puntos sería [-1-1j, -1+1j, 1+1j, 1-1j], especificando el mapeo de los bits [0,1,3,2], cómo se muestra en la figura 60.

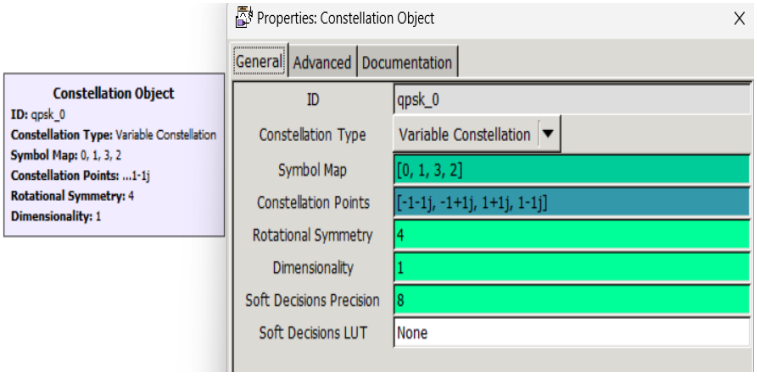


Figura 60. Propiedades del bloque Constellation Object

Fuente: Elaborado por autor

- **Unpack K Bits**

Este bloque se encarga de tomar una secuencia de bytes de entradas y extraer los datos en forma individual en la figura 61 se puede visualizar que para el desarrollo de la práctica se establece en 8 por lo antes indicado.

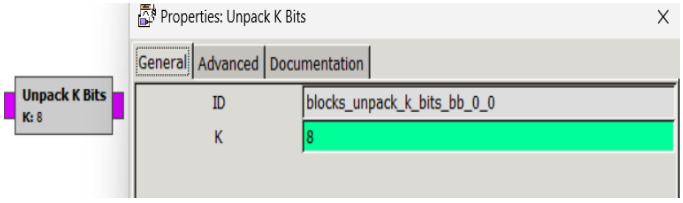


Figura 61. Propiedades del bloque Unpack K Bits

Fuente: Elaborado por autor

- **Chart to float**

Esto es lo que permite compartir los datos de un tipo a otro específicamente de bytes a float en asegura 62 se muestra cómo es el bloque en el software.

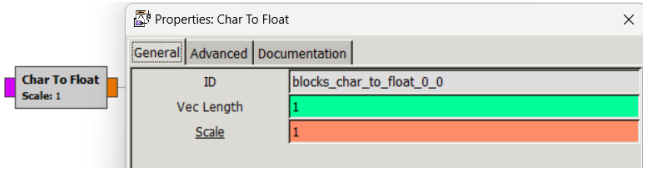


Figura 62. Bloque Chart to Float

Fuente: Elaborado por autor

- **WX GUI Scope Sink**

Este bloque permite visualizar las señales en dominio del tiempo, hace la función de un osciloscopio, mostrando la forma de onda de las señales en tiempo real, representando una o varias señales, además tiene la opción para realizar acercamiento, logrando tener una mejor precisión de datos. En la figura 63 se muestra las configuraciones respectivas del bloque para visualizar la señal, se configuro el tipo de datos que llegan al osciloscopio, la posición donde se encuentra la gráfica, en tipo de disparo se coloca en modo auto donde la señal se actualiza automáticamente y la tasa de muestreo que está definida por la variable *samp_rate* definido con el valor de 64 KHz.

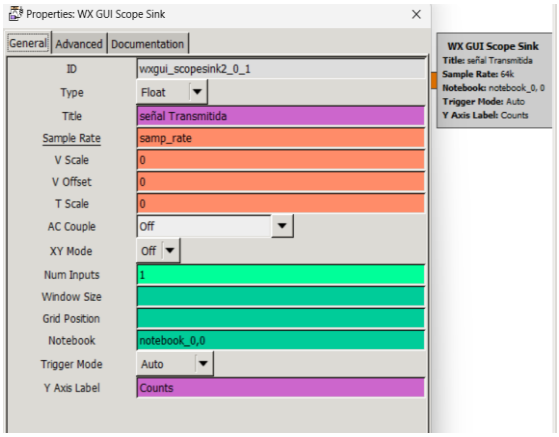


Figura 63. Propiedades del bloque WX GUI Scope Sink

Fuente: Elaborado por autor

- **Throttle**

Este bloque se encarga de copiar todos los elementos en la entrada a la salida, es la velocidad en la que se va a procesar los datos es decir un bloque de aceleración, evita sobrecargas del sistema en la figura 64 se observan las configuraciones necesarias del bloque utilizado para el diagrama transmisor lo cual está definido por la variable *samp_rate* con el valor de 64 KHz y es un tipo de complex.

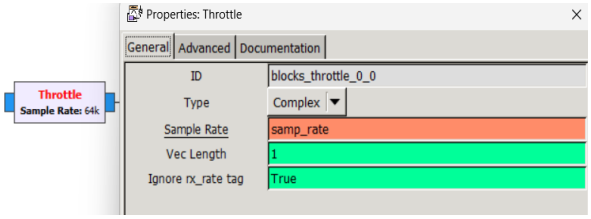


Figura 64. Propiedades del bloque Throttle

Fuente: Elabora por autor

- **Virtual Sink**

Este bloque es utilizado para interconectar el diagrama del transmisor y receptor en una sola ventana de trabajo generalmente se utiliza cuando no se cuenta con un dispositivo hardware, hace la función del canal de comunicación dentro del mismo software en la figura 65 se muestra el bloque utilizado en la transmisión de datos.

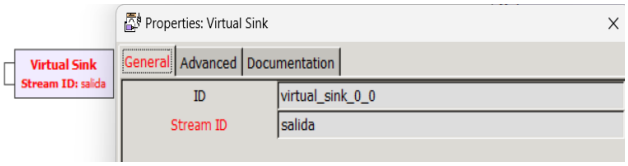


Figura 65. Propiedades del bloque Virtual Sink

Fuente: Elaborado por autor

- **WX GUI Constellation Sink**

Permite de visualizar en un plano complejo la representación gráfica de la constelación de la señal modulada, para el desarrollo de la práctica en la figura 66 se muestran las configuraciones respectivas donde se establece que el ancho de banda es de 62.8m con una frecuencia máxima de 0.06 y 1 tasa de muestreo de 64 KHz

Figura 66. Propiedades del bloque WX GUI Constellation Sink

Fuente: Elaborado por autor

Esquema para la transmisión de modulación QPSK se muestra en la figura 67 permitiendo enviar datos hacia el receptor donde se puede visualizar de manera detallada las conexiones de los bloques antes mencionados con sus respectivas configuraciones además las variables utilizadas en la recepción y transmisión de datos se encuentran encerradas en un recuadro verde.

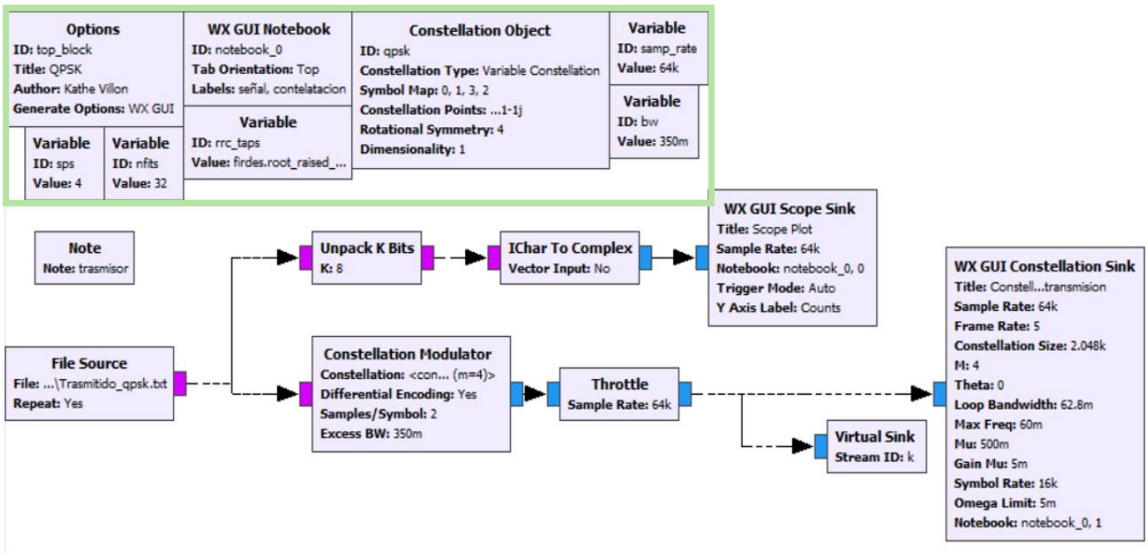


Figura 67. Diagrama transmisor para la modulación QPSK

Fuente: Elaborado por autor

3.2.3.1.4 Receptor QPSK

Desarrollo del diagrama receptor de la modulación digital QPSK se emplearon los siguientes bloques, a continuación, se detalla cada uno con sus respectivas configuraciones.

- **Virtual Source**

Este bloque sirve como punto de conexión para recibir la señal del transmisor, es utilizado cuando se necesita interconectar sistemas, en la figura 68 se muestra en el bloque para el desarrollo del receptor como el canal de transmisión en un caso ideal sin ruidos ni interferencias debido a que se desarrolla internamente en el software.

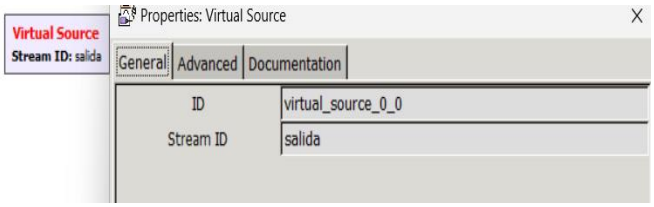


Figura 68. Propiedades del bloque Virtual Source

Fuente: Elaborado por autor

- **Throttle**

Este bloque se encarga de copiar todos los elementos en la entrada a la salida, es la velocidad en la que se va a procesar los datos es decir un bloque de aceleración, evita sobrecargas del sistema en la figura 69 se observan las configuraciones necesarias del bloque utilizado para el diagrama transmisor lo cual está definido por la variable *samp_rate* con el valor de 64 KHz y es un tipo de complex.

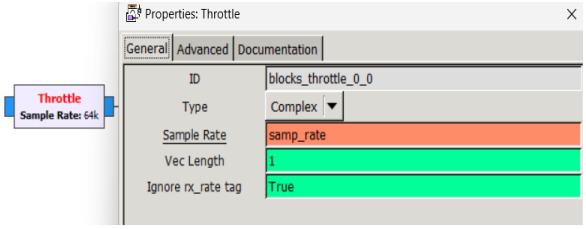


Figura 69. Propiedades del bloque Throttle

Fuente: Elaborado por autor

- **Polyphase Clock Sync**

Ese bloque es utilizado para sincronizar la señal reloj emplea técnicas de muestreo para optimizar y recuperar la fase de la señal recibida permite una mejor detección y de modulación de los símbolos recibidos en la figura 70 se puede visualizar las configuraciones necesarias del bloque y se establece el parámetro *sample/symbol* el valor de cuatro mediante la fórmula es *sps*1* donde *sps* es una variable de valor 4, en *loop bandwindth* es de 62.8m en *taps* se llama la variable *rrc_taps* con fase inicial de 16 y un máximo de desviación de 1.5

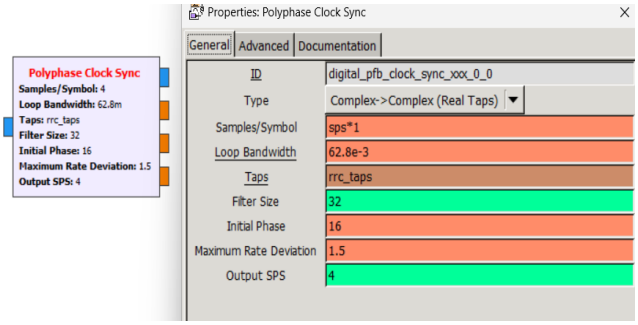


Figura 70. Propiedades del bloque Polyphase Clock Sync

Fuente: Elaborado por autor

- **CMA Equalizer**

Este bloque es un ecualizador adaptativo utilizado para compensar la distorsión de la señal en la entrada, funciona mediante ajustes de coeficientes del ecualizador para minimizar errores entre la magnitud constante de la señal recibida y la señal esperada, en la figura 71 se observa las configuraciones necesarias del bloque donde se establece el número de *taps* en 15 con ganancia de 10m y 4 símbolos por muestra.

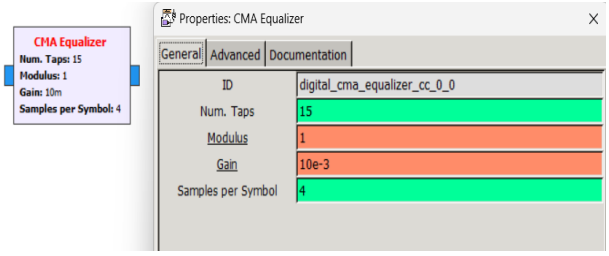


Figura 71. Propiedades del bloque CMA Equalizer

Fuente: Elaborado por autor

- **Costas loop**

Ese bloque es un bucle encargado de controlar la fase para la sincronización de la portadora, permite mejor de modulación en la figura 72 se observó las configuraciones del bloque donde se establece un el mismo *loop bandwindth* que en el bloque *Polyphase Clock Sync* y es de orden 4.

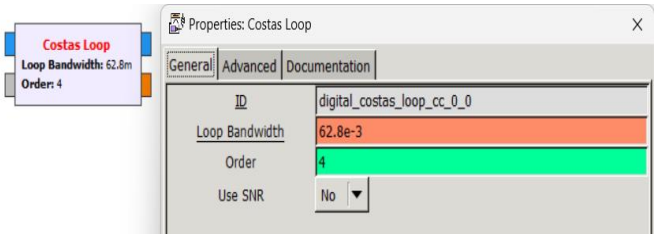


Figura 72. Propiedades del bloque Costas Loop

Fuente: Elaborado por autor

- **Constellation Decoder**

Este bloque es encargado de codificar los símbolos de la constelación recibida, compara la señal recibida con las posiciones de los símbolos en constelación y determina que símbolo va a ser recuperado, en la figura 73 se observan las configuraciones donde se llama la variable QPSK previamente establecida.

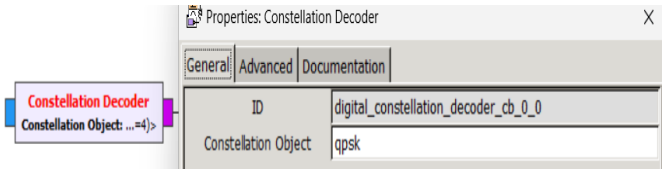


Figura 73. Propiedades del bloque Constellation Decoder

Fuente: Elaborado por autor

- **Diferencial Decoder**

El bloque se encarga de recuperar la información inicial de codificando las diferencias entre símbolos sucesivos en lugar de absoluto, en la figura 74 se visualiza las configuraciones donde se establece que el parámetro *modulus* es 4.

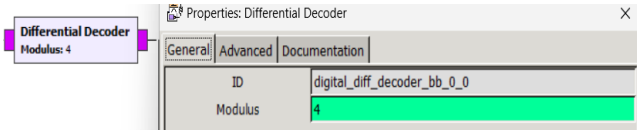


Figura 74. Propiedades del bloque Diferencial Decoder

Fuente: Elaborado por autor

- **Repeack Bits**

El bloque se encarga de organizar los paquetes de bits o reempaquetado los bits de la señal inicial, en la figura 75 se observan las configuraciones establecidas en bits por entradas de bytes en 2 y bits por salidas de bytes en 8.

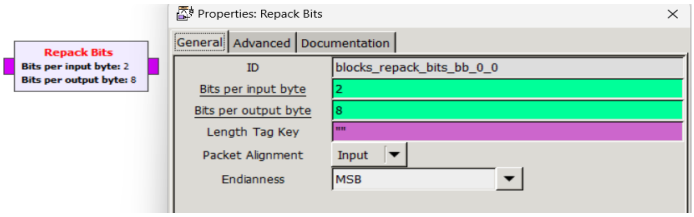


Figura 75. Propiedades del bloque Repack Bits

Fuente: Elaborado por autor

- **Packed to Unpacked**

El bloque convierte los datos empaquetados en bits individuales toma una secuencia de datos y en cada byte contiene varios bits de información generalmente 8 bits forma un byte y los descomponen en bits separados en la figura 76 se detallan las configuraciones realizadas para el

desarrollo de la transmisión de la señal y establece los tipos de datos bytes, un valor de 1 para *bits per chunck.*

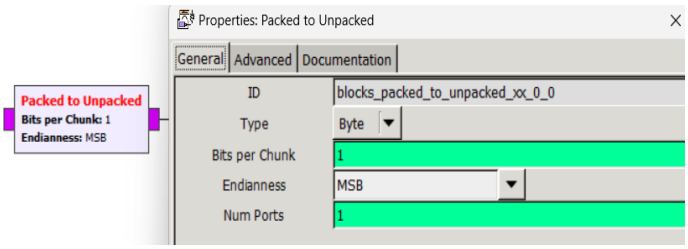


Figura 76. Propiedades del bloque Packed to Unpacked

Fuente: Elaborado por autor

- **File sink**

Este bloque se utiliza para guardar los datos de una señal en forma de archivo, es para almacenar información y posteriormente utilizarlos para analizar los resultados o para reutilizarlos en otro software para el desarrollo de esta práctica serán reutilizados en el software Matlab y realizará un análisis en ambos, la ruta predeterminada es la carpeta donde se guardan los archivos necesarios para el desarrollo de las prácticas en la figura 77 se muestran las configuraciones necesarias además el tipo de datos guardados es de tipo byte con una longitud de un elemento a cada ciclo.

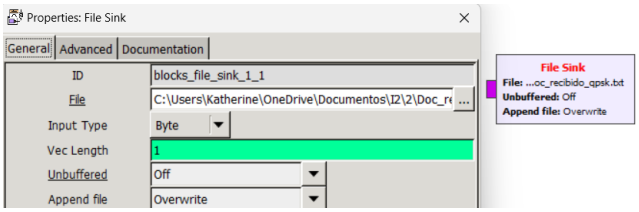


Figura 77. Propiedades del bloque File Sink

Fuente: Elaborado por autor

En la figura 78 se puede observar el esquema receptor correspondiente a la modulación QPSK con sus respectivas conexiones de los bloques mencionados anteriormente permitiendo recibir los datos del transmisor.

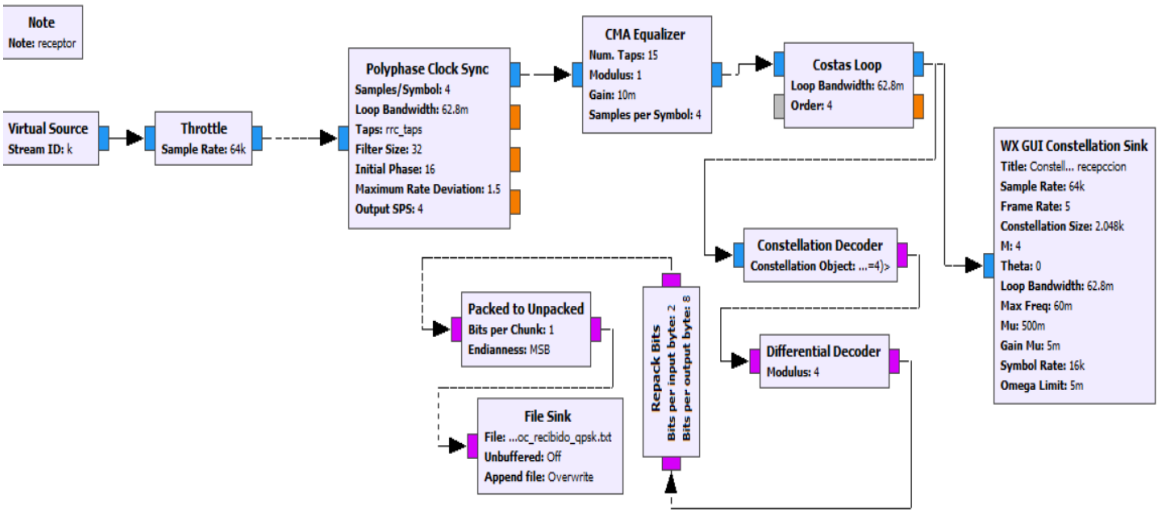


Figura 78. Diagrama receptor para la modulación QPSK

Fuente: Elaborado por autor

3.2.4 Implementación

En la primera parte de la práctica se utilizó los bloques, *virtual sink* y bloque *virtual source* para la simulación de transmisión y recepción de datos sin necesidad de un hardware físico, sin embargo, el objetivo es la implementación realista en esta sección se agregan los bloques *Osmocom Sink* para la etapa de transmisión y *Osmocom Source* para la etapa de recepción en ambas modulaciones.

3.2.4.1 Transmisor de modulación BPSK y QPSK

El bloque *Osmocom Sink* es utilizado para transmitir hacia el receptor en la figura 79 se muestra las configuraciones necesarias para la transmisión de la modulación BPSK y QPSK para la transmisión se realiza las configuraciones en las frecuencias que desea transmitir los datos en este caso se utiliza la frecuencia de 799 MHz de los tipos de datos que se visualizan es tipo *complex float 32* y se debe tener en cuenta que para la transmisión, como para la recepción deben estar en la misma frecuencia para recibir adecuadamente los datos.

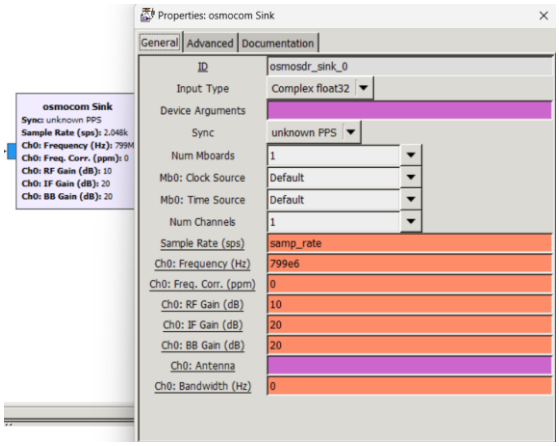


Figura 79. Propiedades del bloque Osmocom Sink

Fuente: Elaborado por autor

El esquema final para la transmisión BPSK se muestra en la figura 80 donde el bloque *virtual sink* es reemplazado por el bloque *Osmocom Sink* el cual permite enviar datos hacia el receptor.

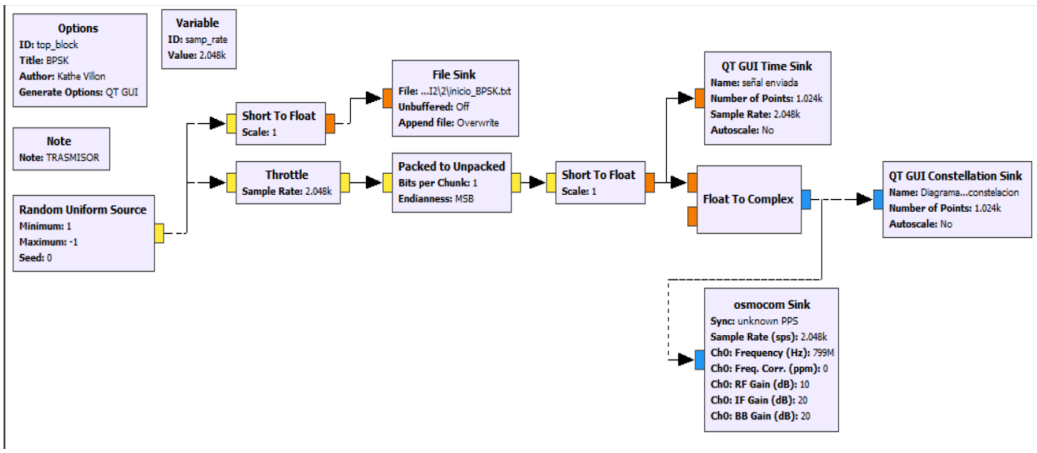


Figura 80. Diagrama final para el transmisor BPSK

Fuente: Elaborado por autor

Mientras para el esquema final para la transmisión QPSK se muestra en la figura 81 donde el bloque *virtual sink* es reemplazado por el bloque *Osmocom Sink* el cual permite enviar datos hacia el receptor.

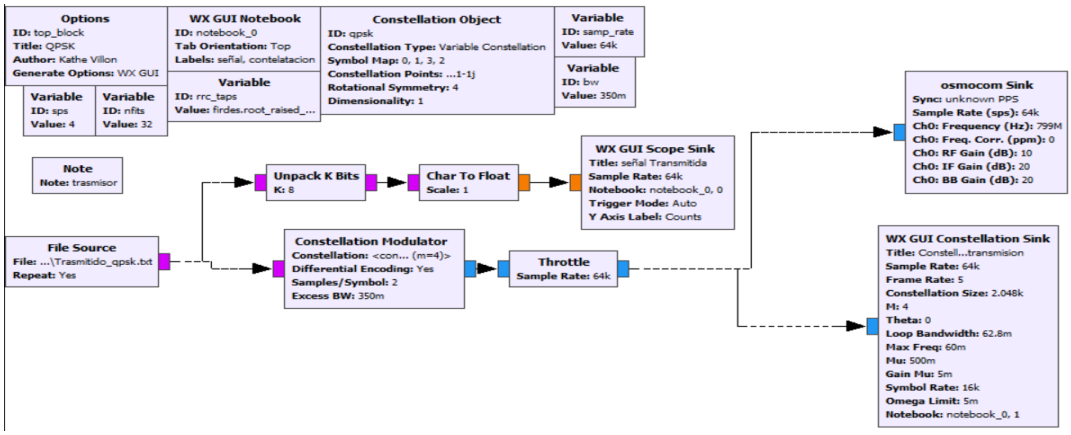


Figura 81. Diagrama final para el transmisor QPSK

Fuente: Elaborado por autor

3.2.4.2 Receptor de modulación BPSK y QPSK

El bloque *Osmocom Source* es utilizado para recibir señales o datos de hasta un transmisor, en la figura 82 se muestra las configuraciones necesarias para la transmisión de datos de modulación BPSK y QPSK, para la recepción de los datos se configura con la misma frecuencia al igual que el transmisor es de 799 MHz para poder recibir los datos de manera correcta y evitar errores en la simulación.

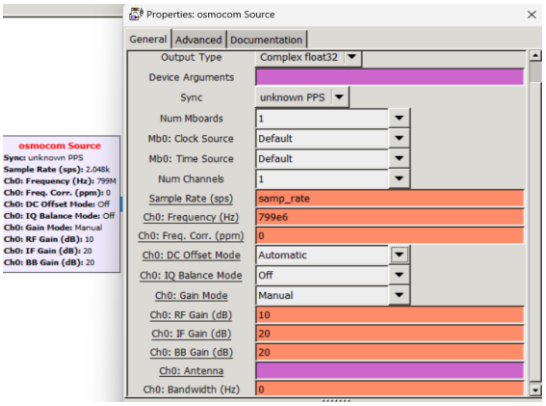


Figura 82. Propiedades del bloque Osmocom Source

Fuente: Elaborado por autor

El esquema final para la modulación BPSK se muestra una figura 83 donde el bloque *virtual source* es reemplazado por el bloque *osmocom source* el cual permite recibir los datos que envía el transmisor de manera correcta.

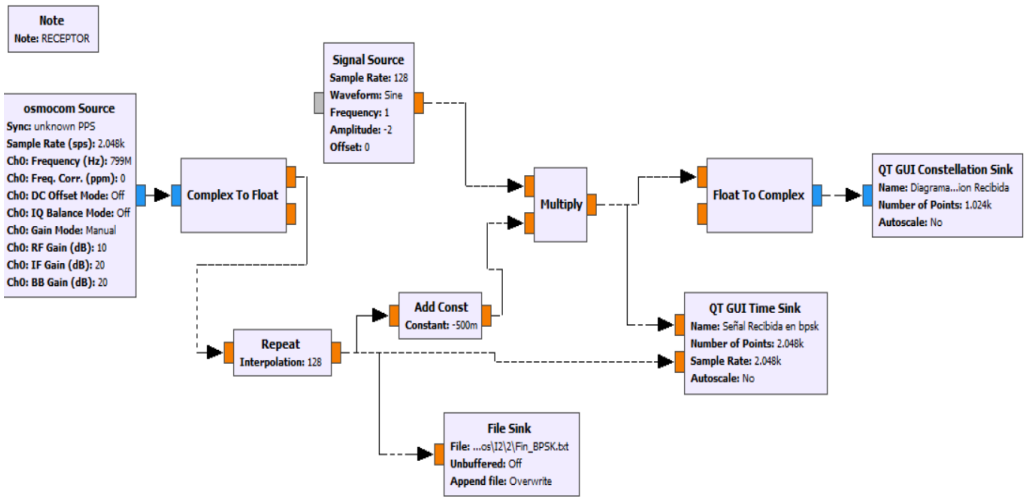


Figura 83. Diagrama final para el receptor de modulación BPSK

Fuente: Elaborado por autor

Así mismo, en la figura 84 se muestra el esquema final del receptor de datos QPSK reemplazando el bloque *virtual source* por el bloque *Osmocom Source*.

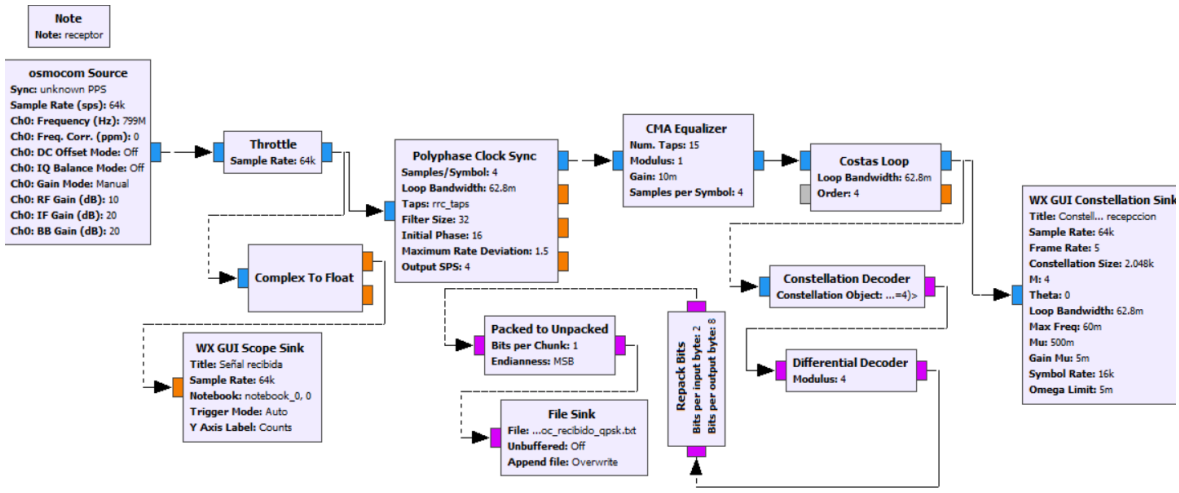


Figura 84. Diagrama final para el receptor de modulación QPSK

Fuente: Elaborado por autor

3.2.5 Simulación y comparación

Uso de Matlab para simular y comparar los esquemas de modulación

3.2.5.1 Configuración de transmisión y recepción en Matlab

3.2.5.1.1 Transmisor BPSK

Para desarrollar la transmisión BPSK y la verificación de los datos de GNU Radio se implementaron en Matlab las líneas de código que se muestra en el anexo 3. A continuación se detallan los bloques de GNU Radio que fueron reemplazados por las líneas de código junto con una breve explicación de cada segmento del código y su función en el proceso.

- En GNU Radio *Random Uniform Source* por la línea *randi ([0 1], 1, N)* en Matlab

El bloque "random uniforme source" se utiliza para generar una secuencia aleatoria de datos bits de 0 y 1 esto se logra mediante la línea *randi ([0 1], 1, N)* que genera un vector aleatorio con longitud N representado de los datos a transmitir.

- En GNU Radio *Throttle* por la línea *sample_rate = 2.048e6* en Matlab

El bloque "throttle" controla la tasa de muestreo de la señal para evitar sobrecarga de procesamiento en Matlab se define la tasa de muestreo mediante la variable *sample_rate* qué establece la frecuencia a la que se muestran los datos para su procedimiento en la siguiente etapa.

- En GNU Radio *Short to Float* por la línea *signal_float = double(bits)* en Matlab

En GNU Radio "Short to float" convierte los datos en un formato punto flotante para ser procesados en el siguiente bloque que realiza mediante la línea *signal_float* que convierte el vector en valores punto flotante.

- En GNU Radio *Packed to Unpacked* por la línea *mod_signal = 2*signal_float -1* en Matlab

En GNU Radio el bloque "Packed to Unpacked" convierte los bits de modulación BPSK asignando 0 y 1 esta operación se replica mediante la línea de código convirtiendo cada bit en su valor modulado donde el 0 se convierte en -1 y el 1 se mantiene como 1

- En GNU Radio *File Sink* por la línea f*ileID = fopen ('BPSK_output.txt', 'w'); fprintf (fileID, '%f\n', mod_signal)* en Matlab

En GNU Radio, el bloque “File Sink” guarda la señal modulada en un archivo. En línea de código su funcionalidad se implementa mediante *fopen, fprintf y fclose*, escribiendo cada valor de *mod_signal* en el archivo ‘BPSK_output.txt’.

- En GNU Radio *QT GUI Time Sink* por la línea *figure; plot(mod_signal); title ('Señal BPSK')* en Matlab

En GNU Radio, el bloque “QT GUI Time Sink” muestra la señal modulada en el tiempo, para Matlab se utiliza plot para gráficar *mod_signal*, lo que permite observar la variación de la señal BPSK en el dominio del tiempo.

- En GNU Radio *QT GUI Constellation Sink* por la línea *figure; scatter (real (mod_signal), imag (mod_signal))* en Matlab

En GNU Radio, el bloque “QT GUI Constellation Sink” muestra el diagrama de constelación de la señal BPSK, para Matlab se realiza con scatter, que gráfica la señal en el plano, permitiendo visualizar los puntos de constelación -1 y 1.

3.2.5.1.2 Receptor BPSK

Para desarrollar la recepción de la señal modulada BPSK y verificar los datos en GNU Radio se implementaron en Matlab las siguientes líneas de código que se muestran en el anexo cuatro a continuación se detallan los bloques del receptor GNU Radio que fueron replicados mediante línea de código en Matlab.

- En GNU Radio *File Source* por la línea *load ('BPSK_output.txt')* en Matlab

El bloque “File Source” en GNU Radio carga los datos de un archivo, mientras que en matlab se utiliza load para cargar los datos desde un archivo llamado ‘BPSK_output.txt’. Este archivo contiene la señal recibida en BPSK.

- En GNU Radio *Threshold* por la línea *signal_bpsk = data_binary_repeated* en Matlab

El bloque de Threshold en GNU Radio actúa como un detector de nivel, separando los valores bajos y altos para diferenciar los bits, para Matlab se realiza una comparación sencilla (como `> 0`) para identificar 1s y 0s.

- En GNU Radio *Throttle* por la línea *fs = 5000* en Matlab

Este bloque limita la velocidad de muestreo para el procesamiento de la señal, para Matlab la frecuencia de muestreo se ajusta con fs.

- En GNU Radio *Scope Sink* por la línea *figure; plot(...)* en Matlab

Este bloque permite visualizar la señal en GNU Radio, para Matlab se utiliza plot para gráficar la señal BPSK y los datos binarios sobrepuestos.

3.2.5.1.3 Transmisor QPSK

Para desarrollar la transmisión QPSK y la verificación de los datos de GNU Radio se implementaron en Matlab las líneas de código que se muestra en el anexo 5. A continuación se detallan los bloques de GNU Radio que fueron reemplazados por las líneas de código junto con una breve explicación de cada segmento del código y su función en el proceso.

- En GNU Radio *Randon Source* por la línea *data = randi ([minVal maxVal], nSamples, 1)* en Matlab

En GNU Radio, el bloque "Random Source" genera números aleatorios, en Matlab se utiliza randi para generar una secuencia aleatoria de valores entre 0 y 255, equivalente a obtener datos aleatorios.

- En GNU Radio *Unpack K Bits (K=8)* por la línea *data_bits = de2bi (data, 8, 'left-msb'); data_bits = data_bits (:)* en Matlab

El bloque "Unpack K Bits" convierte cada valor de 8 bits en bits individuales, para Matlab se utiliza `de2bi` que realiza esta conversión, descomponiendo cada entero en 8 bits y organizándolos en una secuencia continua.

- En GNU Radio *Constellation Modulator* por la línea *unpacked_bits = reshape (data_bits, k, []); symbols = bi2de (unpacked_bits', 'left-msb'); mod_signal = pskmod (symbols, M, pi/M)* en Matlab

En Matlab *reshape* agrupa los bits en bloques de 2, bi2de convierte cada par de bits en un símbolo decimal, *pskmod* aplica la modulación QPSK usando 4 símbolos y un desfase de (pi/M).

- En GNU Radio *WX GUI Scope Sink* por la línea *figure; plot (t, real(mod_signal))* en Matlab

El bloque “Scope Sink” permite observar la señal en el dominio del tiempo, mientras que en Matlab se utiliza plot para visualizar la parte real de la señal modulada en el tiempo.

- En GNU Radio *WX GUI Constellation Sink* por la línea *scatterplot(mod_signal)* en Matlab

El bloque “Constellation Sink” visualiza la constelación transmitida, para Matlab se utiliza el comando scatterplot que muestra la constelación de la señal modulada QPSK.

3.2.5.1.4 Receptor QPSK

Para desarrollar la recepción de la señal modulada QPSK y verificar los datos en GNU Radio se implementaron en Matlab las siguientes líneas de código que se muestran en el anexo cuatro a continuación se detallan los bloques del receptor GNU Radio que fueron replicados mediante línea de código en Matlab.

- GNU Radio *Polyphase Clock Sync* por la línea *sync = comm. SymbolSynchronizer(...) y `synced_signal = sync(mod_signal)* en Matlab

Realiza la sincronización de símbolos, equivalente a la sincronización de reloj en GNU Radio, el SymbolSynchronizer en Matlab ajusta el tiempo de muestreo para asegurar que los símbolos se alineen correctamente para la recepción de la señal.

- En GNU Radio *CMA Equalizer* por la línea *eq = comm. LinearEqualizer ('Algorithm', 'CMA', ...) y equalized_signal = eq(synced_signal)* en Matlab

En GNU Radio se utiliza un ecualizador CMA para corregir distorsiones en el canal, mientras que LinearEqualizer en Matlab con el algoritmo CMA sirve para ajustar la señal y minimizar el ruido.

- En GNU Radio *Costas Loop* por la línea *carrierSync = comm. CarrierSynchronizer(...) y recovered_signal = carrierSync(equalized_signal)* en Matlab

El *CarrierSynchronizer* simula el lazo de Costas, que ajusta la fase de la señal para corregir desplazamientos de frecuencia y de fase.

- En GNU Radio *Constellation Decoder* por la línea *demodulated_symbols = pskdemod (recovered_signal, M, pi/M)* en Matlab

Es encargado de realizar la decodificación diferencial de los datos, el *DifferentialDecoder* en Matlab elimina el efecto de la codificación diferencial en el transmisor, recuperando los datos originales.

- En GNU Radio *Repack Bits* por la línea *received_bits = de2bi (decoded_data, k, 'left-msb'); ... received_bits (:)* en Matlab

Convierte los símbolos decodificados en bits, para Matlab, de2bi convierte cada símbolo en bits y los empaqueta en un solo vector, similar al reempaquetado de bits en GNU Radio.

- En GNU Radio *File Sink* por la línea *dlmwrite ('received_qpsk_data.txt', received_bits, 'delimiter', '\t')* en Matlab

Guarda los bits recibidos en un archivo de texto, esto permite almacenar la salida en un archivo de forma similar al "File Sink" en GNU Radio.

- En GNU Radio *WX GUI Constellation Sink* por la línea *scatterplot(recovered_signal)* en Matlab

Permite visualizar la constelación recibida, para Matlab, el scatterplot muestra la constelación de la señal recuperada, equivalente al bloque de constelación en GNU Radio

- Gráfica de la señal en el dominio del tiempo (Parte Real) por la línea plot (t_recovered, real(recovered_signal)) en Matlab

Esta línea en Matlab corresponde a la gráfica de la parte real de la señal recuperada para el análisis.

3.2.6 Evaluación y Análisis

Comparación de la tasa de error de bits y de los resultados obtenidos entre los esquemas

3.2.6.1 Resultados en GNU Radio

Transmisor BPSK

En la figura 85 muestra la señal enviada por el transmisor utilizando modulación BPSK, se puede apreciar que la señal varía entre los niveles de amplitud de 0 y de 1, correspondiente a dos estados de fase como es de 0 y 180 como se mencionó anteriormente, estos datos son representados en bits binarios.

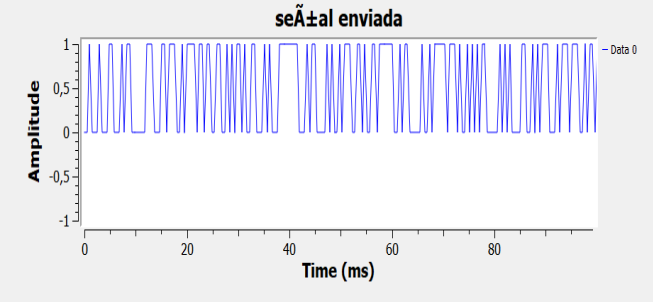


Figura 85. Resultados del Transmisor BPSK en GNU Radio

Fuente: Elaborado por autor

En la figura 86 presenta el diagrama de constelación de la señal BPSK, este diagrama muestra dos puntos definidos los cuales son correspondientes a las fase de 0 y 180 grados, el diagrama de constelación es una herramienta importante para evaluarla calidad de la señal modulada, debido a que permite visualizar los posibles estados de fase, en este caso solo existen dos como se observa en la figura, los puntos se encuentran definidos y sin dispersión, indicando detección de símbolos de forma correcta.

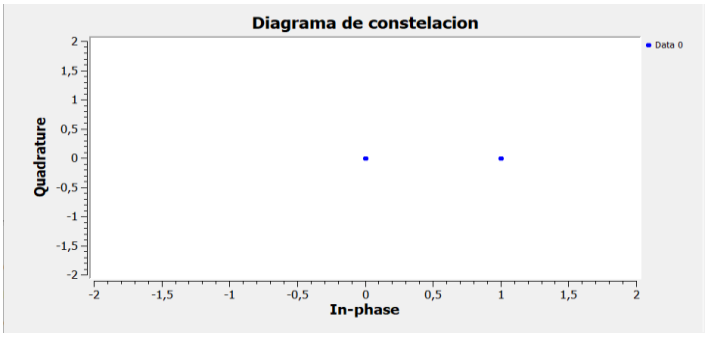


Figura 86. Diagrama de constelación, modulación BPSK en GNU Radio

Fuente: Elaborado por Autor

Receptor BPSK

En la figura 87 se muestra la señal modulada en BPSK correspondiente en tiempo vs amplitud, donde la señal de entrada de color azul tiene variaciones de fase, característico de la modulación BPSK donde los datos binarios 0 y 1 se representan mediante cambios de fase en 180º, es decir 1 y -1. Mientras que los datos digitales están representados de color rojo, donde se superponen a la señal modulada, permitiendo observar los valores de bits de entrada y cambios de fase. La gráfica que proporciona GNU Radio muestra la recepción de datos correspondiente a la señal BPSK, donde las transiciones de fase muestran los cambios de bit de manera adecuada.

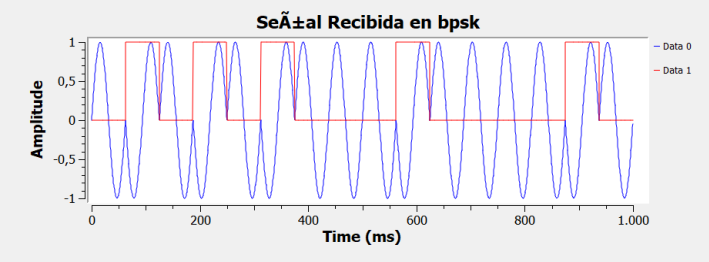


Figura 87. Resultados del receptor de modulación BPSK en GNU Radio

Fuente: Elaborado por autor

Transmisor QPSK

En la figura 88, se observa variaciones en la amplitud que refleja los cambios de fase propios de la modulación QSPK, en la cual cada cambio de señal representa un símbolo que transmite 2 bits de información.

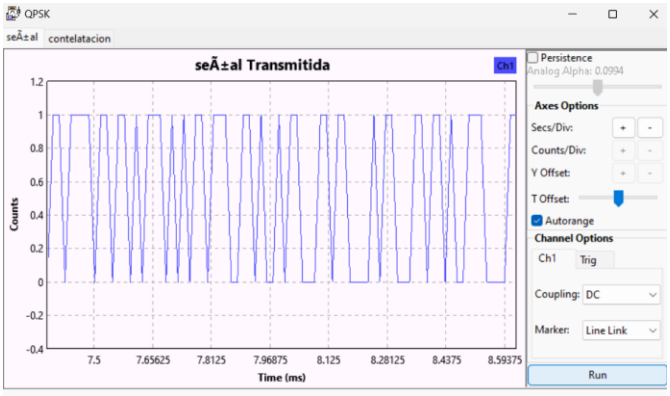


Figura 88. Resultados del transmisor de modulación QPSK en GNU Radio

Fuente: Elaborado por autor

En la figura 89, correspondiente a la constelación de trasmisión, los puntos aparecen agrupados en cuatro posiciones definidas, correspondientes a las cuatro fases de la modulación QPSK (45°, 135°, 225° y 315°), los puntos indican que el transmisor está logrando una la modulación deseada con pequeñas distorsiones, pero mantenido los valores de la fase correspondiente.

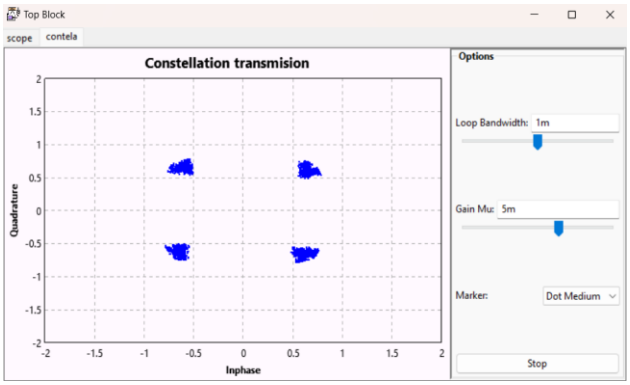


Figura 89. Diagrama de constelación en modulación QPSK en GNU Radio

Fuente: Elaborado por autor

Receptor QPSK

En la figura 90 en el lado izquierdo se puede observar el comportamiento de la señal recibida en el dominio del tiempo, se muestran variaciones correspondientes a los cambios de fase QPSK, presentando pequeñas distorsiones, indicando que los datos enviados no se recuperan en su totalidad, sin embargo, llegan al receptor, en la gráfica de la derecha se muestra los datos recibidos en su forma binaria.

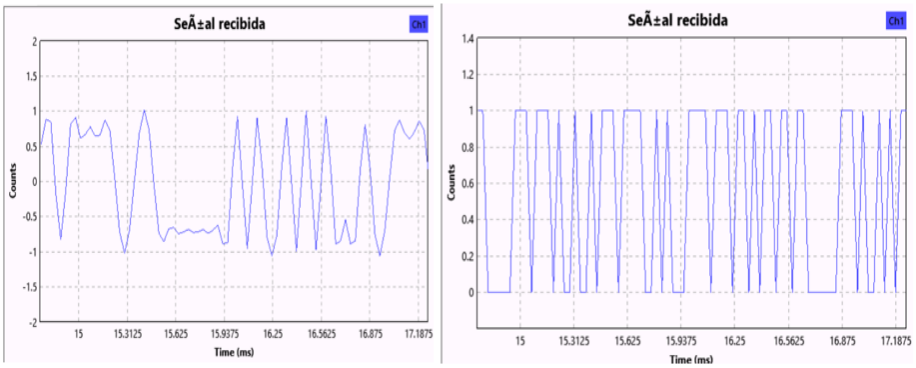


Figura 90. Resultados del receptor de modulación QSPK en GNU Radio

Fuente: Elaborado por autor

En la figura 91 se muestra los puntos los puntos en la constelación agrupados en cuatro fases, característicos de la señal a analizar cómo es la QPSK, indicando que el receptor está identificando correctamente los cambios de fase en 90 grados, pero pueden no estar perfectamente sincronizados o existe un retardo de fase variable, logrando producir que la constelación puede rotarse o expandirse en un patrón circular.

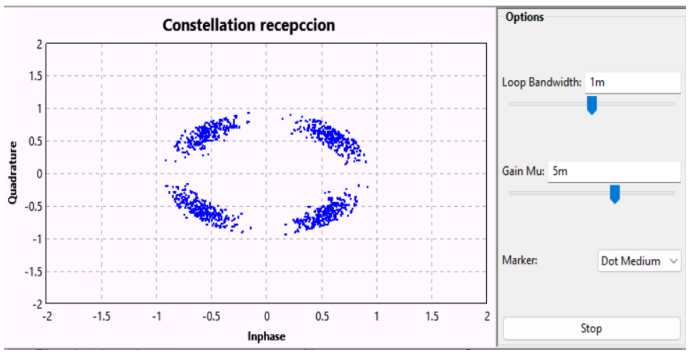


Figura 91. Resultados del receptor en constelación QPSK en GNU Radio

Fuente: Elaborado por autor

3.2.6.2 Comparación de la tasa de error de bits en las modulaciones

Para evaluar la tasa de error de bits en el sistema para la modulación y demodulación BPSK, se realiza el diagrama de la figura 92 en GNU Radio, con fuentes de archivos tipo byte los mismo que fueron guardados en el desarrollo de la práctica, tanto en el transmisor como el receptor, además se utiliza el bloque *Unpack K Bits* como se indicó anteriormente, el bloque es encargado de tomar una secuencia de bytes de entrada y extrae los datos en forma individual, se coloca el bloque *Error Rate* que permite calcular la tasa de error de bits de acuerdo con los datos enviados y el bloque *WX GUI Number Sink* para visualizar el resultado en forma de porcentaje.

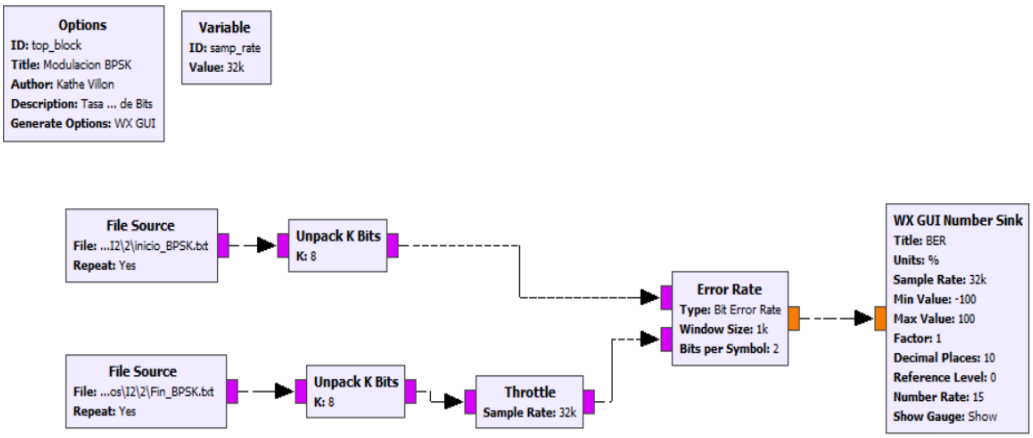


Figura 92. Diagrama para tasa de error de bits de modulación BPSK

Fuente: Elaborado por autor

Para evaluar la tasa de error de bits en el sistema para QPSK, se realiza el diagrama que se muestra en la figura 93, al igual que en BPSK las fuentes de archivos tipo byte guardados en el desarrollo de las prácticas y se coloca los mismos bloques para visualizar la tasa de error de bits en porcentaje.

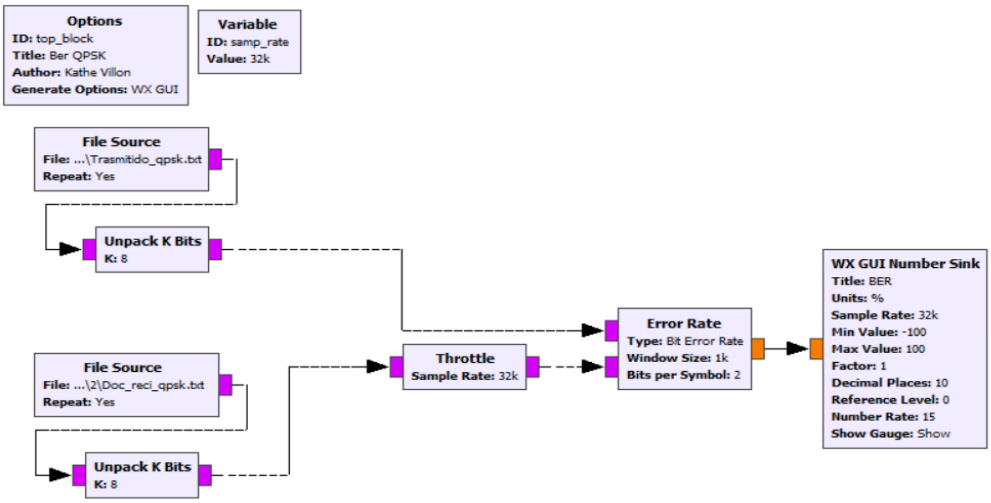


Figura 93. Diagrama para tasa de error de bits de modulación QPSK

Fuente: Elaborado por autor

3.2.6.3 Resultados de tasa de error de bits en GNU Radio

Para la evaluar la calidad de los sistemas de comunicaciones es importante la tasa de error de bits, en la figura 94 se observa la tasa de error de bits de la modulación BPSK con un 0.191% de margen de error el mismo que se calcula por cada 100 muestras, por lo que se puede interpretar que la probabilidad de error es baja es decir se está perdiendo de cada 100 muestra de datos el 0.19 de los datos transmitidos.

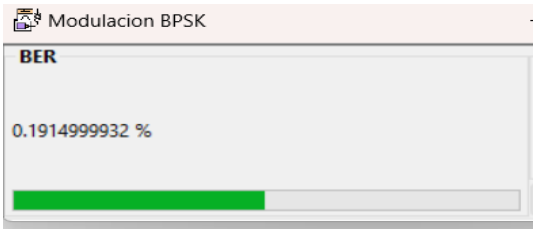


Figura 94. Resultado de la tasa error de bits en BPSK

Fuente: Elaborado por autor

En la figura 95 se observa la tasa de error de bits de la modulación QPSK con un 0.255% de margen de error como se indicó se calcula por cada 100 muestras, por lo que se puede interpretar que la probabilidad de error es baja, sin embargo, este resultado demostró más probabilidad de pérdidas

de un 0.5 que la modulación anterior debido a que se apreciar que por cada 100 datos se pierden el 0.25 de datos.

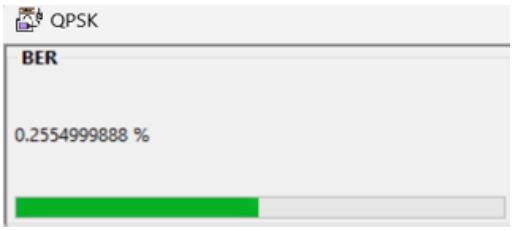


Figura 95. Resultado de la tasa error de bits en QPSK

Fuente: Elaborado por autor

3.2.6.4 Resultados en Matlab

Transmisor BPSK

En la figura 96 correspondiente a la modulación BPSK muestra las variaciones en fase características donde la fase cambia entre 0 y 180º para representar los bits binarios 0 y 1, debido que los datos son aleatorios en ambos software no coinciden con los de GNU Radio, sin embargo el comportamiento es el mismo la señal cambia de fase adecuadamente para cada bit garantizando la transmisión de información el diagrama de constelación refleja los estados correspondientes a la modulación BPSK demostrando excelente agrupación en los puntos y baja dispersión.

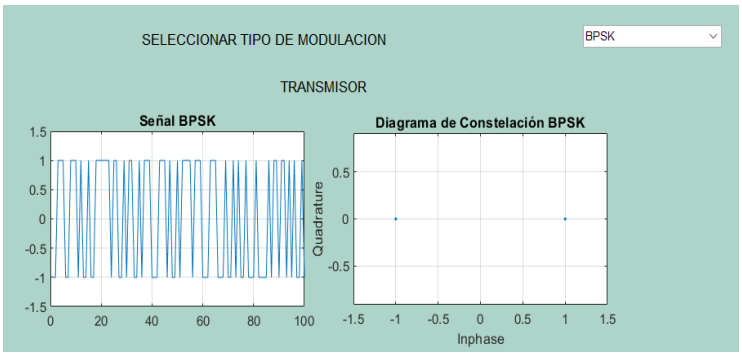


Figura 96. Resultados del transmisor de modulación BPSK en Matlab

Fuente: Elaborado por autor

Receptor BPSK

En la figura 97 muestra las transiciones de la fase correspondiente a los datos aleatorios de entrada, manteniendo la coherencia de los cambios de fase, en el diagrama de constelación la recepción se presenta los estados definidos, mostrando alineación precisa sin dispersión significativa.

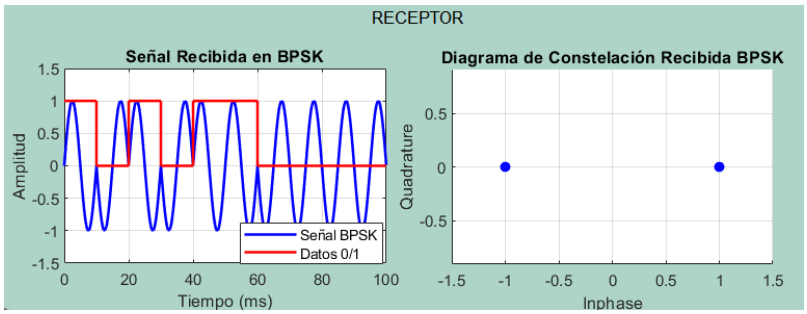


Figura 97. Resultados del receptor de modulación BPSK en Matlab

Fuente: Elaborado por autor

Transmisor QPSK

En la figura 98 se presenta la señal transmitida en Matlab, el lado izquierdo se observa la señal en el dominio del tiempo donde se presenta los cambios de fases acordes a los símbolos QPSK, similar a GNU Radio, la gráfica indica que la transmisión es continua, donde cada variación representa los bits transmitidos.

En el lado derecho se observa la constelación donde muestra cuatro puntos en las posiciones correspondientes de 45º, 135º, 255º y 315º, indicando una transmisión estable y sin distorsiones.

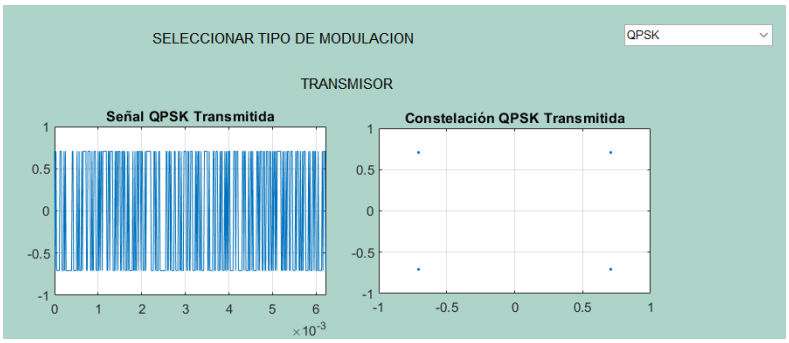


Figura 98. Resultados del transmisor de modulación QPSK en Matlab

Fuente: Elaborado por autor

Receptor QPSK

En la figura 99, el lado izquierdo muestra la señal recibida en el dominio del tiempo en Matlab, muestra variaciones en amplitud similares observadas en GNU Radio, indicando que la modulación se mantiene en las diferentes plataformas. Mientras en el lado derecho se observa el diagrama de constelación obtenida similar, presentando dispersión debido al ruido introducido para simular el canal de transmisión, con posibles interferencias.

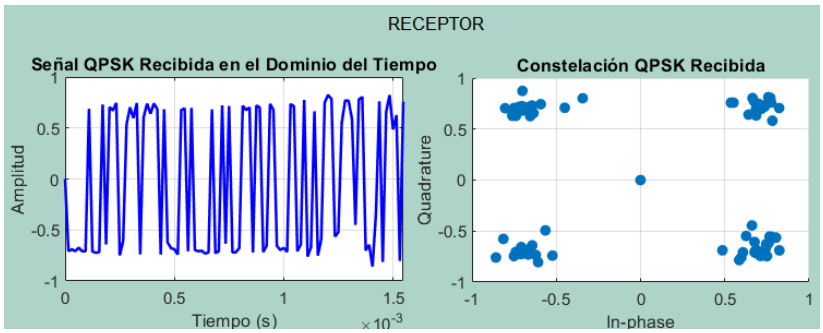


Figura 99. Resultados del receptor de modulación QPSK en Matlab

Fuente: Elaborado por autor

Hipótesis: La implementación de modulación digital en GNU Radio y su verificación en Matlab mejora la precisión de los esquemas de modulación digital.

3.2.7 Conclusión de la práctica

En esta práctica, se logró implementar y evaluar de manera eficiente los esquemas de modulación digital BPSK y QPSK, utilizando GNU Radio como Matlab, en las cuales se observó que la modulación BPSK, al utilizar dos fases, presenta menor tasa de error de bits al transmitir y recibir que la modulación QPSK, siendo más eficiente en condiciones de ruido, sin embargo, QPSK, al transmitir dos bits por símbolo, demuestra mayor eficiencia espectral.

El uso de GNU Radio facilito la simulación de los esquemas en un entorno controlado, permitiendo obtener resultados sin la necesidad de tener un hardware, luego fueron verificados en Matlab lo cual confirmó la precisión de los esquemas implementados evidenciando la interoperabilidad en ambas plataformas para el procesamiento de señal.

Los obtenidos en la simulación y la comparación permitieron validar los esquemas de modulación digital, observando comportamientos similares en las constelaciones y el dominio del tiempo de las señales transmitidas y recibidas, demostrando que los esquemas son efectivos pero cada uno presenta sus propias ventajas.

3.3 Práctica 3.- Desarrollo de un sistema de comunicación de Banda ancha

3.3.1 Descripción de la importancia de los sistemas de comunicación de banda ancha

Los sistemas de banda ancha permiten la transmisión de diferentes señales en un solo canal, facilitando servicios de voz, datos y video, teniendo una mejor experiencia en línea los usuarios y mayor eficiencia en el acceso de internet, a demás son esenciales para el crecimiento económico e innovación, impulsando el desarrollo de regiones rurales donde el uso de comunicación moderna ayuda a reducir la brecha digital[27].

Los sistemas de comunicación de banda ancha son más resilientes, garantizando confidencialidad en los servicios de comunicación y disponibilidad en momentos importantes como situaciones de emergencia, donde la conectividad pude ser vital[28].

Un sistema de comunicación de banda ancha es el estándar IEEE 802.11a, este sistema es clave para la transmisión de datos en entornos inalámbricos, alcanzando tasas de transferencia de 54 Mbps, resultando fundamental en aplicaciones que se requieren un alto ancho de banda, como la transmisión de videos en alta definición y juegos en líneas, también minimiza la interferencia provocada por dispositivos como microondas o teléfonos inalámbricos, teniendo conexiones más estables en área con bastantes dispositivos[29].

La modulación por división de frecuencia ortogonal es una característica importante del sistema 802.11a, debido que mejora la resistencia a interferencias y desvanecimiento de la señal, OFDM - *Orthogonal Frequency Division Multiplexing,* divide el canal de comunicación en múltiples portadoras, permitiendo una transmisión más segura y eficiente, maximizando el uso del espectro disponible al permitir transmitir múltiples flujos de datos en paralelo, resulta eficaz en un entorno donde el ancho de banda está en constante crecimiento[30].

Objetivo de la práctica: Desarrollar un sistema de comunicación para la transmisión de banda ancha utilizando HackRF One, GNU Radio y Matlab.

3.3.2 Revisión Teórica

Sistema de Comunicación Inalámbrica

Los sistemas de comunicación inalámbricos son aquellos que permite la transmisión de información mediante ondas electromagnéticas, que se propagan en un medio guiado o no guiado como el aire, estos sistemas son esenciales en actualmente porque facilitan la conexión y envió de datos sin necesidad de un medio cableado.

Los dispositivos que comúnmente se conectan a redes inalámbricas incluyen computadoras portátiles, de escritorio, asistentes personales digitales, teléfonos celulares, tables, aunque el funcionamiento de las redes inalámbricas es similar a las redes cableadas, es importante que las señales deben convertirse a un formato adecuado para la transmisión en el aire, estas redes son altamente versátiles y pueden utilizarse para sustituir redes cableadas o para transmitir datos desde ubicaciones remotas, ofreciendo mejora acceso a la información en diferentes entornos[31].

OFDM - *Orthogonal Frequency Division Multiplexing*

Existen diferentes esquemas de transmisión, clasificando en técnicas de portadora única y de múltiples portadoras OFDM, es un ejemplo de transmisión de múltiples portadoras, donde la diferencia principal de los esquemas de transmisión de portadora única y de múltiples portadoras es que la portadora única envía la información sobre una sola portadora, y OFDM disminuye la información en varias portadoras separadas de manera ortogonal[32].

El desarrollo de esta tecnología comenzó hace varios años, aunque su implementación inicial resulta compleja debido a que se necesitaba múltiples generadores sinusoidales para las subportadoras y un número correspondiente de demoduladores, esta configuración, hacia que esta práctica sea ineficiente y poco viable, para resolver esta problemática se realizó una reformulación teórica de OFDM basada en DFT *Discrete Fourier Transform*, que luego se optimizo con FFT *Fast Fourier Transform*, gracias a esto la OFDM puede llevar a cabo la transmisión en subportadoras paralelas, ayudando a disminuir ISI *Inter Symbol Interference* y ICI *Inter Carrier Interference*[33].

Hoy en día, algunas tecnologías inalámbricas y estándares, como es el estándar IEEE 802.11, LTE *long Term Evolution*, WiMAX *Worldwide Interoperability for Microwave Access* e ISDB – Tb *Integrated Service of Digital Broadcasting – Terrestrial*, emplean OFDM en su capa física, debido

a su capacidad para reducir interferencia, permitiendo una modulación adaptable y robustez a efectos del multi-trayecto[33].

Modelo General del Sistema OFDM

El sistema permite transmitir varios símbolos en paralelo, los cuales son modulados en conjunto específico de subportadoras que mantienen una separación mínima de frecuencia necesaria para conservar ortogonalidad, al realizar un sistema OFDM, es importante tener en cuenta puntos claves como el número de subportadoras, intervalo de guarda, duración del símbolo, el espacio entre subportadoras y el tipo de modulación[33].

Estructura de la trama de datos IEEE 802.11a

Según el estándar, los símbolos de OFDM consta de un total de 64 subportadoras, de las cuales 48 están dedicadas a datos, 4 son subportadoras piloto y las restantes nulas o virtuales, en la tabla 6 se muestra las características de varios estándares aplicados en OFDM, donde la práctica se centra en el primer estándar[34].

Estándar	N_{FFT}	N_{CP}	N_{FFT} / N_{CP}
IEEE 802.11 a/g	64	16	$\frac{1}{4}$
IEEE 802.11 n/ac	128	32, 16	$\frac{1}{4}, \frac{1}{8}$
DVB-T	2048, 4096	64, 128, 256, 512	$\frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}$

Tabla 6. Características de diferentes estándares aplicados en OFDM

Fuente: [33]

3.3.3 Diseño del Sistema

3.3.3.1 Configuración del flujo de transmisión y recepción de GNU Radio

El desarrollo del sistema se basa en comunicación básica de OFDM utilizando características de la capa física del estándar IEEE 802.11a en OFDM, con las siguientes características:

- Tamaño de N_FFT = 64
- Número de subportadoras = 52
- Subportadoras de datos = 48
- Subportadoras piloto = 4
- Longitud prefijo cíclico = ¼
- Modulación cabecera = BPSK
- Modulación carga útil = QPSK

Para el desarrollo del sistema de comunicaciones, es esencial definir los esquemas transmisor y receptor, donde cada uno de estos sistemas cuentan con variables y parámetros específicos que deben ser considerados para garantizar el rendimiento adecuado del sistema, las cuales se muestran en la figura 100.

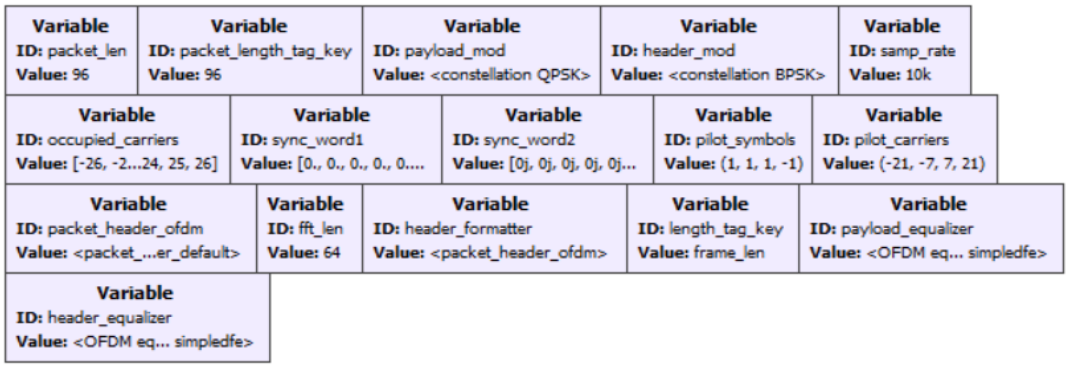


Figura 100. Variables de los esquemas trasmisor y receptor

Fuente: Elaborado por autor

3.3.3.1.1 Transmisor OFDM

En esta sección de presenta las configuraciones de los bloques para la implementación de un transmisor de un sistema básico de modulación OFDM, a continuación, se describe cada bloque utilizado en el sistema.

- **Random Source**

Este bloque genera una secuencia aleatoria de datos definiendo como mínimo 0y máximo de 255, simulando una fuente de datos binarios, el número de muestras para la transmisión está representado por el parámetro *Num samples*, el cual está definido en 100 para el desarrollo de la

práctica, permitiendo mejor visualización y en el parámetro *repeat* debe estar en modo yes para enviar más de una vez la señal como se muestra en la figura 101.

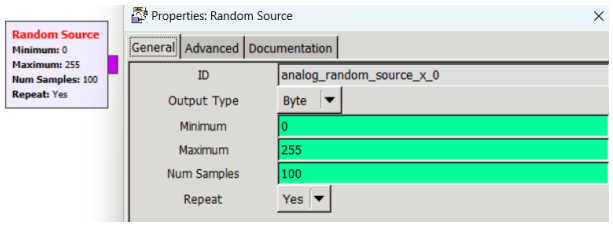


Figura 101. Propiedades del bloque Random Source

Fuente: Elaborado por autor

- **Throttle**

El bloque se encarga de procesar los datos, evitando la sobrecarga en el sistema, en la figura 102, se puede observar el valor establecido para el sistema.

Figura 102. Bloque Throttle

Fuente: Elaborado por autor

- **Stream to Tagget Stream**

Este bloque convierte el flujo continuo de datos en un flujo etiquetado con longitud definida para formar los paquetes, permitiendo segmentar al flujo de bits de manera simétrica la información para cada segmento, al realizar este proceso los bloques que continúan en el sistema pueden armar los paquetes con el tamaño adecuado, en la figura 103 se visualiza las configuraciones donde la longitud del paquete esta predefinida por la variable *packet_leng* con el valor de 96, debido a la cara útil o *payload* el cual tomara 2 bits por símbolos es decir será modulada en QPSK, la salida de la modulación estará etiquetado por 48 símbolos, que luego serán asignados en las subportadoras de datos.

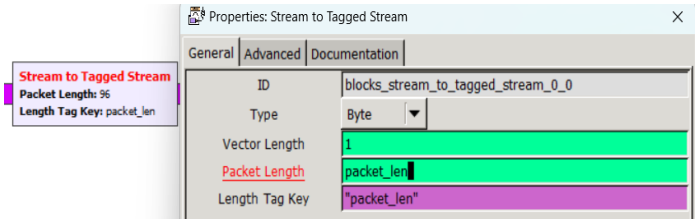


Figura 103. Propiedades del bloque Stream to Tagged Stream

Fuente: Elaborado por autor

- **Stream CRC32**

Este bloque genera una suma el cual permite verificar o detectar error dentro de las etiquetas de cada flujo en la detección de símbolos para la etapa de recepción, genera cabeceras que serán insertadas en paquetes OFDM, mientras los flujos son empaquetados, las cabeceras son generadas para la modulación, definiendo la longitud del paquete con la variable *packet_len* es decir 96 bits como se muestra en la figura 104.

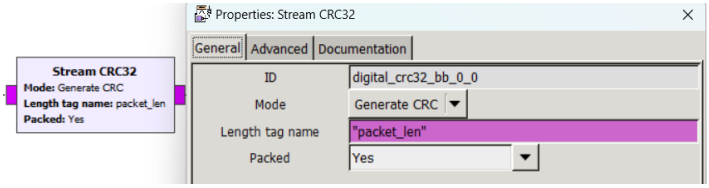


Figura 104. Propiedades del bloque Stream CRC32

Fuente: Elaborado por autor

- **Packet Header Generator**

Este bloque se encarga de generar cabecera que depende de la variable packet_len y su formato se genera de forma automática con la variable *packet_header_ofdm*, que está definida por *digital.packet_header_default()*, inmediatamente después de generar las cabeceras que se insertan en los paquetes los datos se demodulan.

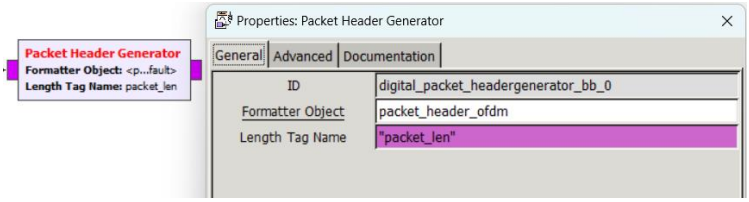


Figura 105. Propiedades del bloque Packet Header Generator

Fuente: Elaborado por autor

- **Repack bits**

Este bloque permite empaquetar los bits de carga útil, dependiendo el nivel de modulación que se va a utilizar, agrupando por bits para el desarrollo de la práctica se utiliza la modulación QPSK como se mencionó al inicio, empaquetando 4 símbolos por cada 8 bits, es decir empaqueta 2 bits por símbolo, el bloque encargado se muestra en la figura 106.

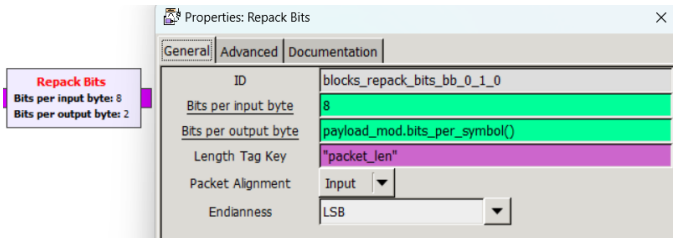


Figura 106. Propiedades del bloque Repack Bits

Fuente: Elaborado por autor

- **Chunks to Symbols**

Este bloque se encarga de asignar un flujo de índice de los símbolos desempaquetado a un flujo de puntos de constelación flotante o complejos, el parámetro encargado de mapear trozos a símbolos es Symbol Table, establecido por la constelación BPSK para la cabecera enviando dos niveles de modulación como se muestra en la figura 107.

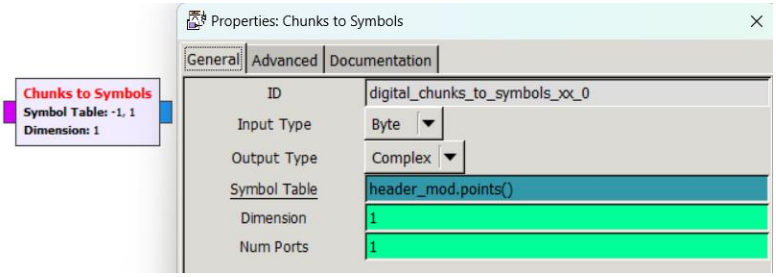


Figura 107. Propiedades del bloque Chunks to Symbols para cabecera

Fuente: Elaborada por autor

Mientras que para la carga útil se establece la constelación QPSK con 4 niveles y 2 bits por símbolos como se muestra en la figura 107.

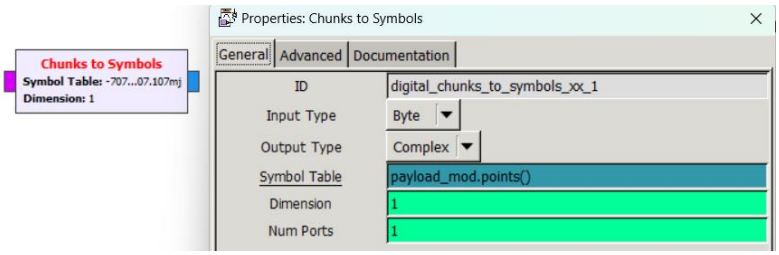


Figura 108. Propiedades del bloque Chunks to Symbols para carga útil

Fuente: Elaborado por autor

- **Tagged Streamms Mux**

El bloque es utilizado para unir la cabecera y la carga útil permitiendo multiplexar los flujos de información que fueron etiquetados en la cabecera por el bloque *Stream to Tagget Stream*, hasta esta sección pertenece a la etapa de Pre-OFDM lista para ingresar a los bloques encargados para la OFDM, en la figura 109 se muestra la respectiva configuración.

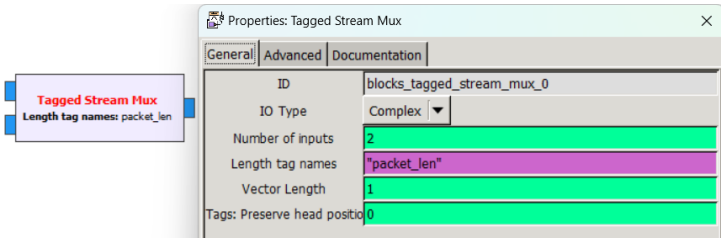


Figura 109. piedades del bloque Tagged Stream Mux

Fuente: Elaborado por autor

- **OFDM Carrier Allocator**

Este bloque es encargado de asignar la portadora, clasificando los complejos entrantes en la portadora, además se encarga de colocar los símbolos piloto en las posiciones correctas y palabras de sincronismo, el bloque depende se número total de subportadoras, definidos por la variable *fft_len* con el valor de 64, divididas de la siguiente manera

- *Occupied Carriers:* Establece la posición de todas las 48 subportadoras de datos, 5 portadoras van desde el -25 al -22, 13 portadoras desde el -20 hasta el -8, 6 portadoras desde el -6 hasta el -1, 6 portadoras desde el 1 al 6, 13 portadoras desde el 8 hasta el 20, 5 portadoras desde el 22 al 25.
- *Pilot Carriers:* Establece las 4 subportadoras dentro de los símbolos (-21, -7, 7, 21).

- *Pilot symbol*: Define los símbolos pilotos, en los datos a transmitir establecidos mediante el vector (1, 1, 1, -1).

Además, se añade las palabras de sincronismo se debe tener en cuenta que este bloque realiza la conversión de serie a paralelo enviando flujos en paralelo de la IFFT.

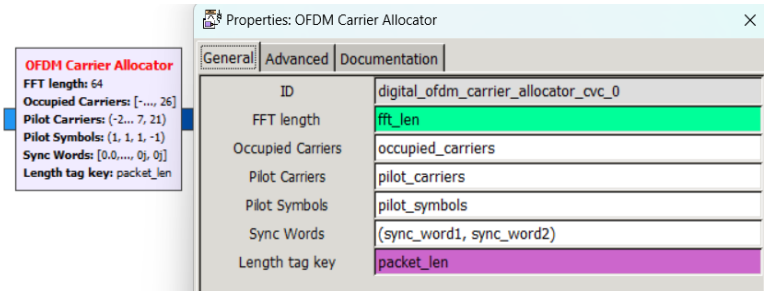


Figura 110. Propiedades del bloque OFDM Carrier Allocator

Fuente: Elaborado por autor

Las palabras de sincronismo contienen 64 símbolos y se establecen mediante las variables sync_word1 y sync_word2 encontradas en las figuras 111 y 112.

Sync_word1

En la primera palabra de sincronismo o símbolo de entrenamiento, se utiliza la secuencia de 0 y secuencia pseudoaleatoria o PN, ceros en las frecuencias impares y una secuencia pseudoaleatoria de ruido en frecuencias pares, multiplicando por $\sqrt{2}$, para mantener una energía de la señal constante.

Los colores representan las frecuencias para e impar y bandas guarda

- Azul: frecuencias pares
- Naranja: Frecuencias impares
- Verde: Bandas guarda

```
[0., 0., 0., 0., 0., 0., 0., 1.41421356, 0., -1.41421356, 0., 1.41421356, 0., -1.41421356,0., -1.41421356,
0., -1.41421356, 0., 1.41421356, 0., -1.41421356, 0., 1.41421356, 0., -1.41421356, 0., -1.41421356, 0.,
-1.41421356, 0., -1.41421356, 0., 1.41421356, 0., -1.41421356, 0., 1.41421356, 0., 1.41421356, 0.,
1.41421356, 0., -1.41421356, 0., 1.41421356, 0., 1.41421356, 0., 1.41421356, 0., -1.41421356, 0.,
1.41421356, 0., 1.41421356, 0., 1.41421356, 0., 0., 0., 0., 0., 0.]
```

Figura 111. Palabras de sincronización para sync_word1

Fuente: Elaborado por Autor

Sync_word2

El segundo símbolo o palabra de sincronismo contiene dos secuencias PN, una para frecuencias pares que ayuda a determinar el desplazamiento de la frecuencia y otra para frecuencias impares que ayuda a medir los subcanales.

Cada tipo de color representa las frecuencias

- Azul: frecuencias pares
- Naranja: Frecuencias impares
- Verde: Bandas guarda
- Rojo: Frecuencia central

```
[0j, 0j, 0j, 0j, 0j, 0j, (-1+0j), (-1+0j), (-1+0j), (-1+0j), (1+0j), (1+0j), (-1+0j), (-1+0j), (-1+0j), (1+0j), (-1+0j),
(1+0j), (1+0j), (1 +0j), (1+0j), (1+0j), (-1+0j), (-1+0j), (-1+0j), (-1+0j), (-1+0j), (1+0j), (-1+0j), (-1+0j), (1+0j),
(-1+0j), 0j, (1+0j), (-1+0j), (1+0j), (1+0j), (1+0j), (-1+0j), (1+0j), (1+0j), (1+0j), (-1+0j), (1+0j), (1+0j), (1+0j),
(1+0j), (-1+0j), (1+0j), (-1+0j), (-1+0j), (- 1+0j), (1+0j), (-1+0j), (1+0j), (-1+0j), (-1+0j), (-1+0j), (-1+0j), 0j, 0j,
0j, 0j, 0j]
```

Figura 112. Palabras de sincronización para sync_word2

Fuente: Elaborado por autor

- **FFT**

Este bloque se encarga de realizar el proceso de la transformada de Fourier, permitiendo configurar en reversa logrando la inversa de transformada de Fourier, el cual depende de la variable *fft_len*, representando la longitud de la IFFT en la figura 113 se visualiza la configuración de necesaria de bloque.

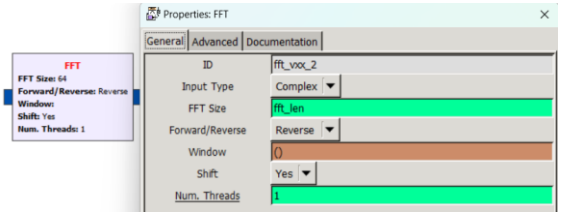


Figura 113. Propiedades del bloque FFT

Fuente: Elaborado por autor

- **OFDM Cyclic Prefixer**

El bloque se encarga de insertar el prefijo cíclico de la señal que también depende de la variable *fft_len* con el valor de 64, representado el número total de subportadoras, *CP Lenght* es encargado

de especificar el valor correspondiente al prefijo cíclico, que se obtiene al dividir el total de subportadoras para 4 como se muestran en el estándar 802.11a en la figura 114.

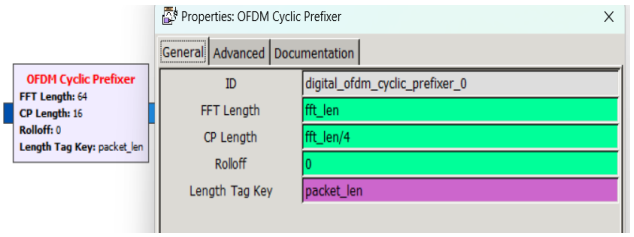


Figura 114. Propiedades del bloque OFDM Cyclic Prefixer

Fuente: Elaborado por autor

- **Multiply Const**

Este bloque se encarga de multiplicar la señal por una constante, en este caso es de 50m, permitiendo ajustar la amplitud de la señal como se muestra en la figura 115.

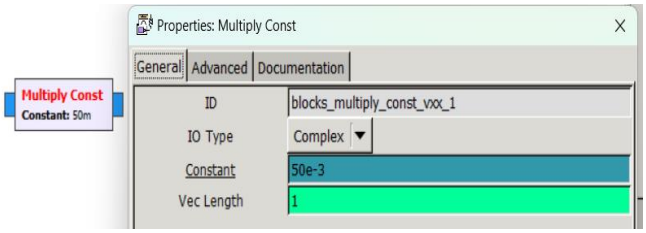


Figura 115. Propiedades del bloque Multiply Const

Fuente: Elaborado por autor

- **Tag Gate**

El bloque Tag Gate es encargado de controlar el flujo de etiquetas, permitiendo o bloqueando el paso de etiquetas en la figura 116 se muestra la configuración correspondiente del bloque.

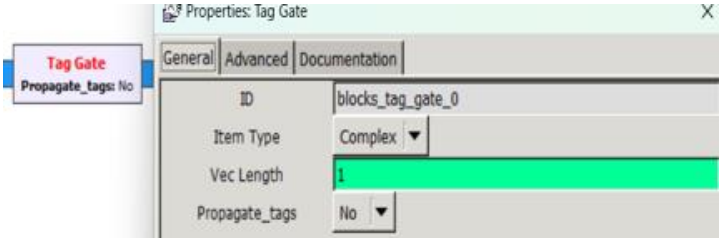


Figura 116. Propiedades del bloque Tag Gate

Fuente: Elaborado por autor

El diseño del sistema transmisor se divide en cuatro partes como se observa en la figura 117, la primera se encarga de generar paquetes, para ser moduladas y enviarlos al flujo de datos donde aparecen las portadoras y símbolos pilotos, pasando al dominio del tiempo mientras añade un prefijo cíclico, para finalmente enviar la información de datos a través de un medio hacia el receptor.

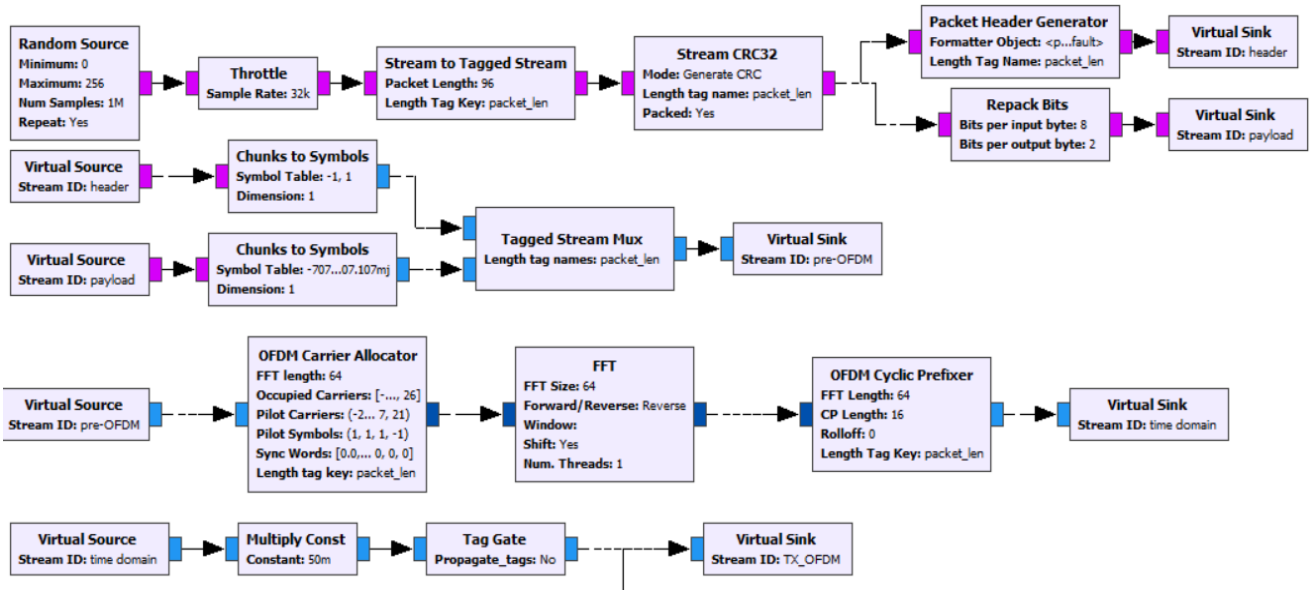


Figura 117. Esquema transmisor de OFDM

Fuente: Elaborado por autor

3.3.3.1.2 Receptor OFDM

En esta sección se presenta las configuraciones de los bloques para la implementación del receptor en un sistema tradicional con modulación OFDM las cuales se detallan a continuación los bloques correspondientes.

- **Schmidl & Cox OFDM Synch**

Este bloque se encarga de sincronizar la información en tiempo y en frecuencia, tiene dos salidas, una entrega el valor de desplazamiento, y la otra salida se encarga de la detección de palabras, mientras ayuda a delimitar los símbolos, cabeceras y carga útil, como se muestra en la figura 118.

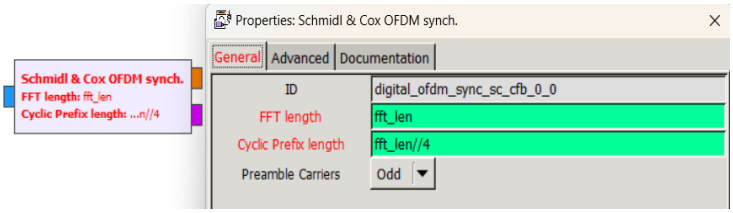


Figura 118. Propiedades del bloque Schmidl & Cox OFDM synch

Fuente: Elaborado por autor

- **Frequency Mod**

El bloque es encargado de entregar el valor al cual varia las muestras de frecuencia y del procedimiento, debido que el bloque anterior no entrega inmediatamente el valor de desviación de frecuencia, si no cuanto cambia la fase en función de la entrada, la desviación de frecuencia se obtiene utilizando un modulador analógico de frecuencia, en la figura 119 se muestra las configuraciones realiza en el bloque.

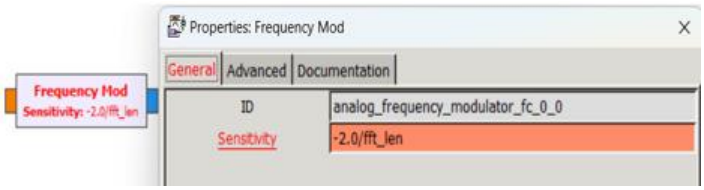


Figura 119. Propiedades del bloque Frecuencia Mod

Fuente: Elaborado por autor

- **Delay**

Este bloque es utilizado mientras se procesa un símbolo en el bloque *Schmidl & Cox OFDM Synch* sincronizando a la salida y este sea el mismo símbolo corregido, es decir que el símbolo que ingresa al sincronizador se retrasa a la duración de un símbolo, en la figura 120 se puede observar las configuraciones necesarias.

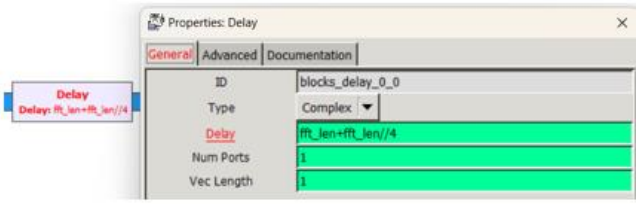


Figura 120. Propiedades del bloque Delay

Fuente: Elaborado por autor

- **Multiply**

Este bloque es encargado de multiplicar todos los flujos de la entrada del diagrama transmisor en la figura 121, se muestra el bloque con su respectiva configuración.

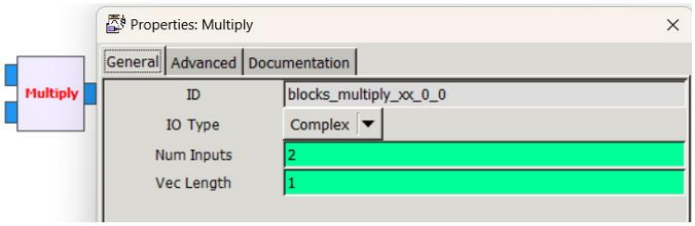


Figura 121. Propiedades del bloque Multiply

Fuente: Elaborado por autor

- **Header/Payload Demux**

Este bloque permite demultiplexación de cabeceras y de carga útil, el bloque contiene 3 entradas la primera entrada es de datos o información de carga útil, la segunda entrada de datos para el control del bloque de sincronismo y la tercera entrada corresponde el control de cabeceras después de ser procesadas, a la salida tiene la carga de cabecera y carga útil como se observa en la figura 122 con sus respectivas configuraciones.

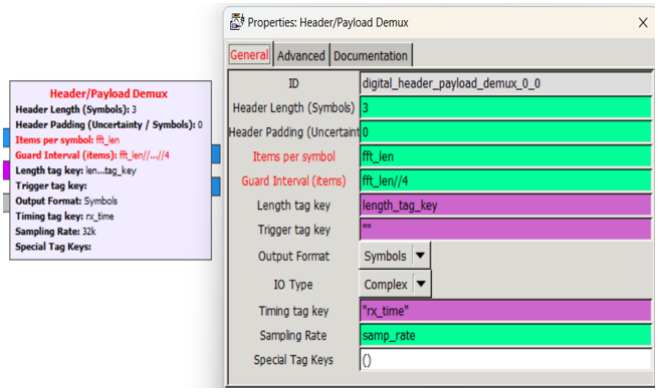


Figura 122. Propiedades del bloque Header/Payload demux

Fuente: Elaborado por autor

- **Packet Header Parser**

Este bloque es encargado de analizar la información de cabeceras y entregar de datos al bloque *Header/Payload demux*, con el fin de retroalimentar el procesamiento y aplicar correcciones de

estimación de canal además de la ecualización, en la figura 123 se observa las configuraciones respectivas.

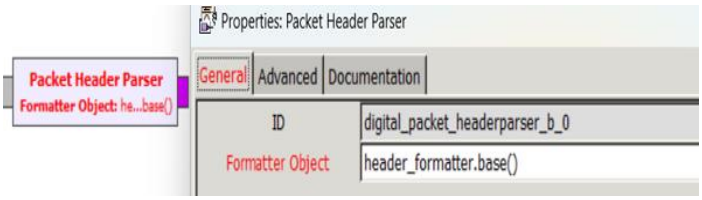


Figura 123. Propiedades del bloque Packet Header Parser

Fuente: Elaborado por autor

- **Constellation Decoder**

Para la demodulación de la cabecera se utiliza el bloque *Constellation Decoder*, debido que este bloque se encarga de realizar el proceso inverso al bloque que permite des - mapear los símbolos de acuerdo con la tabla de modulación que se utiliza en el transmisor, en la cabecera se utiliza la modulación BPSK como se observa en la figura 124.

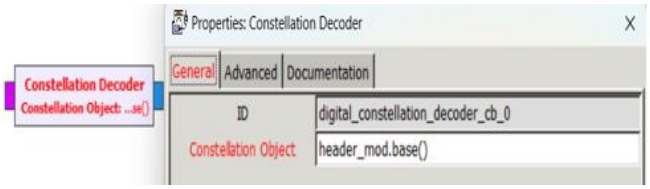


Figura 124. Propiedades del bloque Constellation Decoder BPSK

Fuente: Elaborado por autor

Asimismo, en la figura 125 se presenta las configuraciones para la modulación QPSK o de carga útil, a diferencia del bloque anterior este se realiza mediante la línea payload_mod.base().

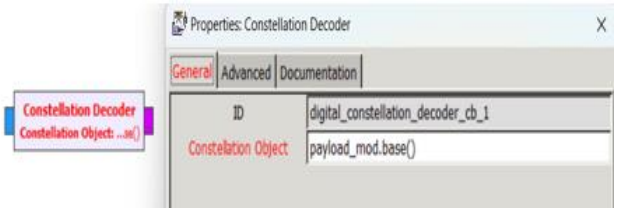


Figura 125. Propiedades del bloque Constellation Decoder QPSK

Fuente: Elaborado por autor

- **OFDM Serial**

El bloque *OFDM Serial*, se encarga de convertir los flujos de símbolos en un solo flujo de forma paralela tomando la información de cada subportadora entregando el flujo de símbolo en serie para ser demodulado, en la figura 126 se observa las configuraciones necesarias.

Figura 126. Propiedades del bloque OFDM Serializer BPSK

Fuente: Elaborado por autor

En la figura 127 se muestra la configuración del bloque, pero para la modulación QPSK o para la carga útil.

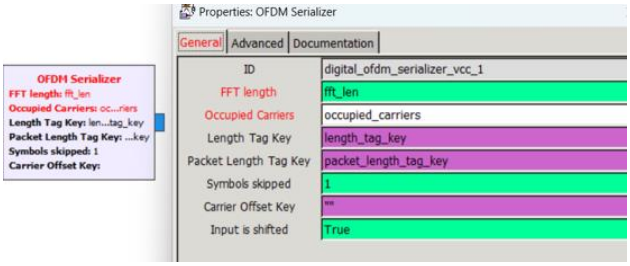


Figura 127. Propiedades del bloque OFDM Serializer de payload

Fuente: Elaborado por autor

- **OFDM Frame Equalizer**

Es el encargado de reducir fallos en el canal, se utiliza la información sobre la estimación del canal que se obtuvo en el procesamiento de la cabecera retroalimentado al demultiplexor, en la figura 128, se observa las configuraciones respectivas para la parte de *header_mod*.

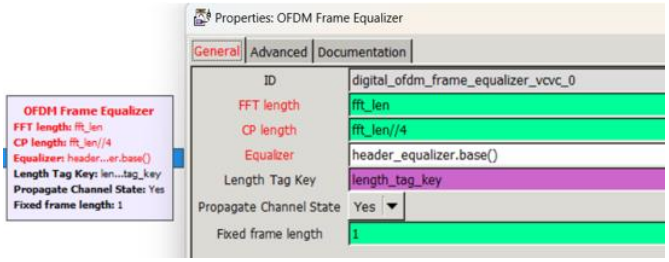


Figura 128. Propiedades del bloque OFDM Frame Equalizer BPSK

Fuente: Elaborado por autor

En la figura 129 se muestra la configuración del bloque para payload, respectivo para la modulación QSPK.

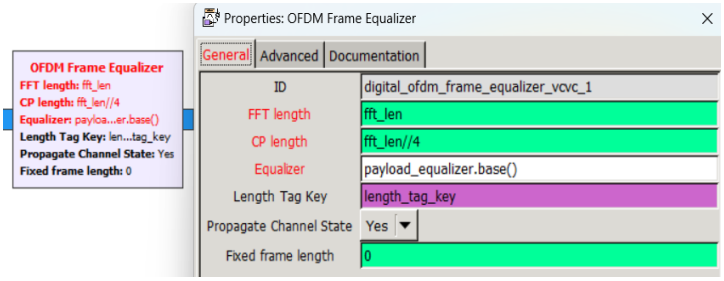


Figura 129. Propiedades del bloque OFDM Frame Equalizer payload

Fuente: Elaborado por autor

- **OFDM Channel Estimation**

El bloque tiene como entrada símbolos OFDM en el dominio de la frecuencia, donde se espera que los primeros símbolos sean símbolos de sincronización, que se utilizan para estimar la desviación de frecuencia y las derivaciones iniciales del ecualizador, en el bloque se utiliza las variables de *sync_word1* y *sync_word2*, como se observa en la figura 130.

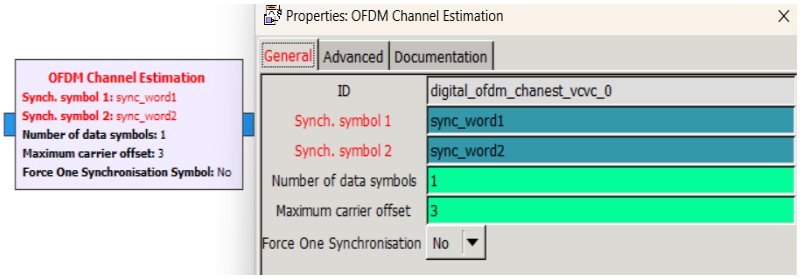


Figura 130. Propiedades del bloque OFDM Channel Estimation

Fuente: Elaborado por autor

- **FFT**

Este bloque es encargado de asignar la portadora para la cabecera, generando símbolos OFDM, convirtiéndole las señales en el domino del tiempo, antes de seguir en su proceso, razón por la que se canalizan el bloque FFT, en la figura 131 se muestra las configuraciones necesarias.

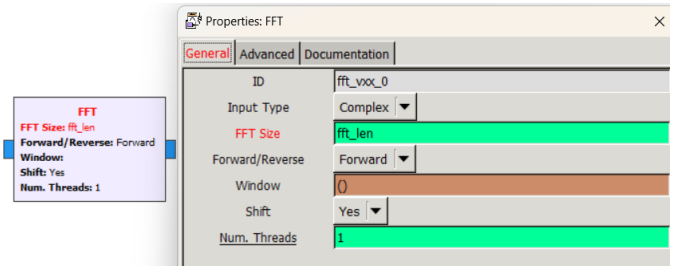


Figura 131. Propiedades del bloque FFT para header

Fuente: Elaborado por autor

En la figura 132 se presenta la configuración para la parte de payload o respectivamente la modulación QPSK, el cual tiene la misma configuración que el bloque anterior.

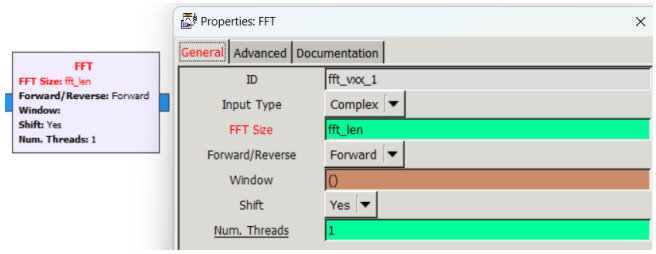


Figura 132. Propiedades del bloque FFT para payload

Fuente: Elaborado por autor

- **Repack Bits**

Es el encargado de volver a empaquetar los bits k del flujo de la entrada de bits 1 y del flujo de salida, aquí no se debe perder ningún bit, se permite cualquier valor para k y l, dentro de un rango de [1, 8], en cada nuevo byte de entrada, comienza a leer en el LSB, y comienza a copiar en el LSB también.

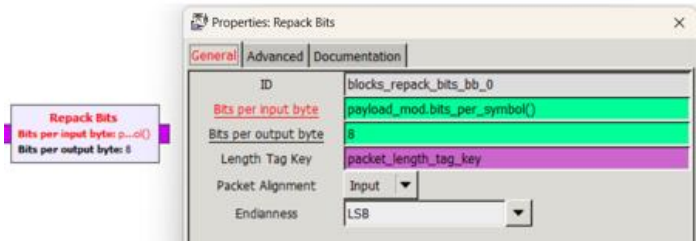


Figura 133. Propiedades del bloque Repeack Bits

Fuente: Elaborado por autor.

- **Stream CRC32**

Este bloque permite realizar la verificación de las etiquetas insertadas en la trasmisión antes de presentar la información ya recuperada, en la figura 134, se muestra las configuraciones respectivas.

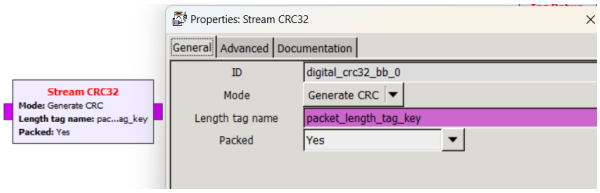


Figura 134, Propiedades del bloque Stream CRC32

Fuente: Elaborado por autor

- **Tag Debug**

El bloque se encarga de recoger todas las etiquetas que se envían en todos los puertos de entradas y las muestra en la ventana de comando, en la figura 135 se observa las configuraciones del bloque

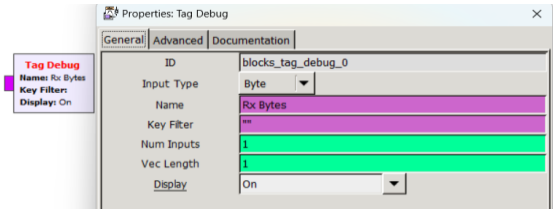


Figura 135. Propiedades del bloque Tag Debug

Fuente: Elaborado por autor

La etapa del receptor también consta de cuatro etapas al igual que el transmisor, la primera es encargada de la detección de símbolos OFDM, luego se envía a la ecualización y detección de cabecera encargada de detectar si el paquete tiene una correcta demultiplexación para seguir en la

siguiente etapa de recuperación o reconstrucción de los datos enviados, como se muestra en la figura 136, se muestra en mi cuarto.

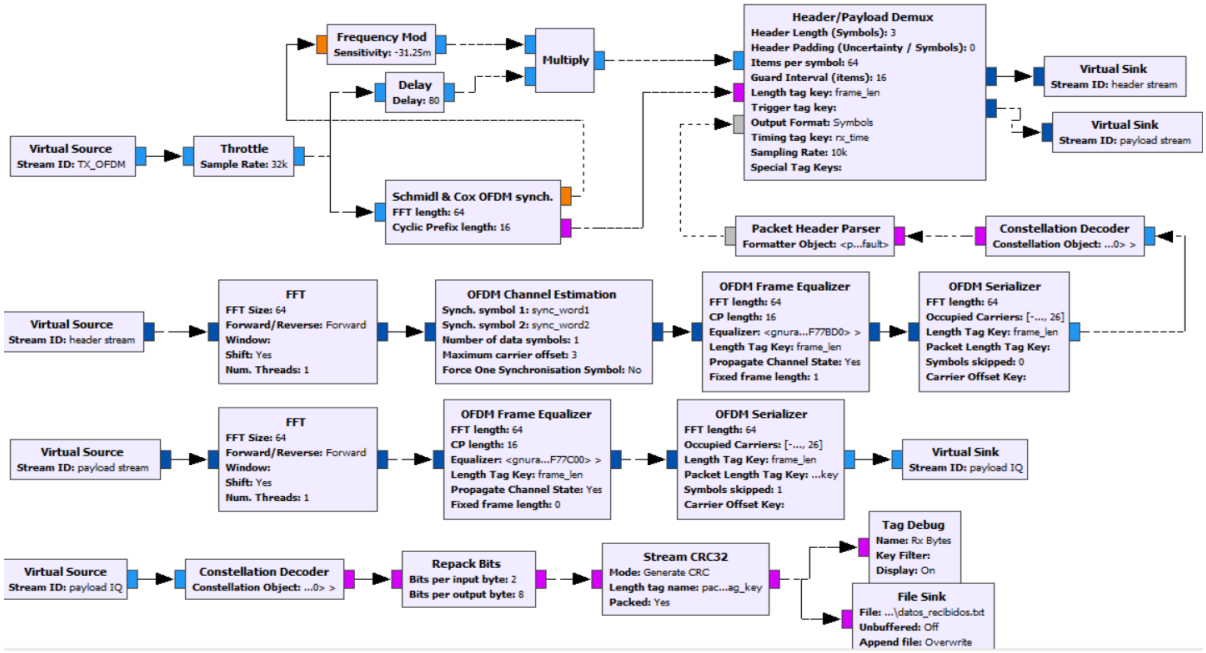


Figura 136. Esquema receptor para OFDM

Fuente: Elaborado por autor

3.3.3 Implementación

Los bloques que permiten la configuración entre los equipos y el programa del transmisor y receptor los bloques encargados para ejecutar la práctica son los que se muestran en la figura 137.

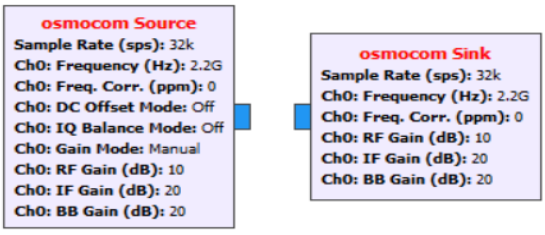


Figura 137. Bloques encargados de trasmitir y recibir datos

Fuente: Elaborado por autor

Para la transmisión se utiliza el bloque *osmocom sink,* configurado frecuencia central de 2.2GHz, ganancia de la antena, frecuencia de muestreo, como se muestra en la figura 138.

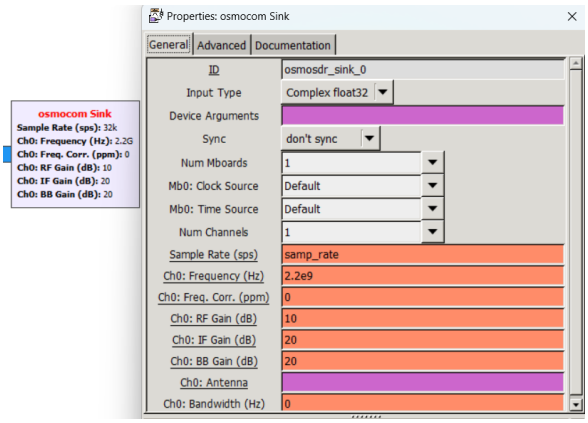


Figura 138. Propiedades del bloque Osmocom Sink

Fuente: Elaborado por autor

Para la recepción se utiliza el bloque *osmocom source*, configurado frecuencia central de 2.2GHz, ganancia de la antena, frecuencia de muestreo, como se muestra en la figura 139.

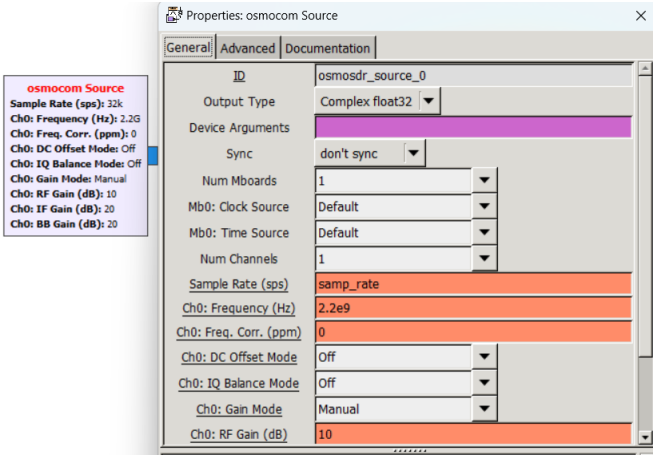


Figura 139. Propiedades del bloque Osmocom Source

Fuente: Elaborado por autor

3.3.4 Simulación y Verificación

Uso de Matlab para simular y verificar el sistema

3.3.4.1 Configuración del flujo de transmisión y recepción en Matlab

3.3.4.1.1 Transmisor OFDM

Para realizar la transmisión y verificación de datos en la simulación de comunicación de banda ancha utilizando OFDM, se desarrollaron las líneas de código en Matlab, las cuales se encuentran

en el anexo 7, a continuación, se presenta las equivalencias correspondientes a los bloques GNU Radio.

- En GNU Radio *Random Source* por la línea *fileID = fopen('tramiison.txt', 'r');text = fread(fileID, '*char')';fclose(fileID);bits = reshape(de2bi(uint8(text), 8, 'left-msb')', 1, [])* en Matlab

Esta línea se encarga de leer el archivo de texto y convertirlo en bits, equivalente a una fuente aleatoria que genera datos a transmitir.

- En GNU Radio *Stream to Tagged Stream* por la línea *packet_len = length(bits_crc*) en Matlab

Se encarga de calcular la longitud del paquete después de añadir el CRC32

- En GNU Radio *Stream CRC32* por la línea *function bits_crc = append_crc32(bits) crc = randi ([0 1], 1, 32); bits_crc = [bits, crc]; end bits_crc = append_crc32(bits)* en Matlab

Esta línea simula la adicción de un CRC32 a los bits de datos y similar al bloque CRC32 en GNU Radio.

- En GNU Radio *Packet Header Generator* por la línea *header_bits = randi([0 1], 1, 96); header_symbols = pskmod(header_bits.', mod_order_header, pi)* en Matlab

Es encargada de generar y modular la cabecera BPSK e identificar el inicio de cada paquete

- En GNU Radio *Repack Bits* por la *línea payload_symbols = qammod(payload_bits.', mod_order_payload, 'InputType', 'bit', 'UnitAveragePower', true)* en Matlab

Se encarga de empaquetar los bits de carga útil y los modula en QPSK, igual que en proceso de empaquetamiento y modulación en OFDM.

- En GNU Radio *Tagged Stream Mux* por la línea *tx_symbols = [header_symbols; payload_symbols]* en Matlab

Combina los símbolos de la cabecera y carga útil en una sola secuencia

- En GNU Radio *Chunks to Symbols* por la línea *header_symbols = pskmod(header_bits.', mod_order_header, pi)* en Matlab

Es encargada de modular la cabecera, equivalente al mapeo de bits en símbolos.

- En GNU Radio *OFDM Carrier Allocator* por la línea *ofdm_symbols(occupied_carriers + FFT_length/2 + 1, :) = reshape(tx_symbols, num_symbols, [])* en Matlab

Asigna los símbolos modulados en las portadoras correspondientes del esquema OFDM.

- En GNU Radio *FFT* por la línea *tx_signal = ifft (ofdm_symbols, FFT_length)* en Matlab

Realiza la inversa de la transformada de Fourier logrando pasar los datos al domino del tiempo, generando señal OFDM.

- En GNU Radio *OFDM Cyclic Prefixer* por la línea *tx_signal_cp = [tx_signal(end-CP_length+1:end, :); tx_signal]* en Matlab

Añade al prefijo cíclico en el dominio del tiempo a la señal OFDM

- En GNU Radio *Multiply Const* por la línea *tx_signal_cp = tx_signal_cp * const_mult* en Matlab

Se encarga de ajustar la amplitud de la señal con un multiplicador constante

- En GNU Radio *Virtual Sink TX_OFDM* por la línea *save ('datos_transmision.mat', 'tx_signal_cp', 'header_bits')* en Matlab

Guarda la señal transmitida y la cabecera en un archivo para el receptor

- En GNU Radio *Virtual Sink time domain* por la línea *load ('datos_transmision.mat', 'tx_signal_cp'; tx_signal_cp_flat=tx_signal_cp(:)* en Matlab

Es la encargada de graficar la señal OFDM en el dominio del tiempo

- En GNU Radio *QT GUI Frequency Sink* por la línea *sample_rate = 10e6; carrier_frequency = 5e9; N = length(tx_signal_cp_flat); tx_spectrum = fft(tx_signal_cp_flat, N)* en Matlab

Se encarga de calcular y graficar la señal OFDM transmitida

3.3.4.1.2 Receptor OFDM

Para realizar la recepción y verificación de datos en la simulación de comunicación de banda ancha utilizando OFDM, se desarrollaron las líneas de código en Matlab, las cuales se encuentran en el anexo 8, a continuación, se presenta las equivalencias correspondientes a los bloques GNU Radio.

- En GNU Radio *Virtual Source TX_OFDM* por la línea *load ('datos_transmision.mat', 'tx_signal_cp')* en Matlab

Carga la señal transmitida con el prefijo cíclico desde un archivo previamente guardado

- En GNU Radio *Throttle* por la línea *rx_signal = tx_signal_cp* en Matlab

Pasa la señal transmitida directamente al fujo de entrada hacia el receptor

- En GNU Radio *Schmidl & Cox OFDM synch* por la línea *rx_signal_sync = schmidl_cox_synchronization (rx_signal, FFT_length, CP_length)* en Matlab

Realiza la sincronización de la señal recibida usando método Schmidl & Cox

- En GNU Radio *Header/Payload Demux* por la línea [*header_signal, payload_signal] = header_payload_demux (rx_signal_sync, CP_length, FFT_length)* en Matlab

Se encarga de dividir la señal sincronizada en dos partes como es cabecera (header) y carga útil (payload)

- En GNU Radio *FFT (header stream)* por la línea *header_freq = fftshift(fft(header_signal (CP_length+1:CP_length+FFT_length), FFT_length))* en Matlab

Convierte la cabecera de la señal del dominio de tiempo al dominio de la frecuencia mediante FFT

- En GNU Radio *OFDM Channel Estimation* por la línea *header_data = ofdm_channel_estimation (header_freq, sync_word_1, sync_word_2)* en Matlab

Estima las características del canal usando palabras de sincronización, logrando compensar distorsiones de la señal

- En GNU Radio *OFDM Frame Equalizer* por la línea *header_equalized = ofdm_frame_equalizer(header_data, occupied_carriers)* en Matlab

Ecualiza las portadoras ocupadas en la cabera, eliminando el efecto en la señal

- En GNU Radio *Constellation Decoder (BPSK)* por la línea *header_bits = constellation_decoder (header_equalized, 'BPSK')* en Matlab

Decodifica la cabera usando modulación BPSK, convirtiendo los símbolos de la constelación en bits

- En GNU Radio Packet Header Parser por la línea packet_length = packet_header_parser(header_bits) en Matlab

Es el encargado de extraer la longitud del paquete de los primeros bits de la cabecera, interpretándolos como longitud payload, para determinar el tamaño del paquete.

- En GNU Radio *FFT (payload stream)* por la línea *payload_freq = fftshift (fft(payload_signal (CP_length+1:CP_length+FFT_length), FFT_length))* en Matlab

Convierte la carga útil del dominio del tiempo al domino de la frecuencia mediante FFT

- En GNU Radio *OFDM Frame Equalizer* por la línea *payload_equalized = ofdm_frame_equalizer (payload_freq, occupied_carriers)* en Matlab

Ecualiza las portadoras ocupadas en la carga útil, eliminado el efecto del canal en la señal.

- En GNU Radio *Constellation Decoder (QPSK)* por la línea *payload_bits = constellation_decoder (payload_equalized, 'QPSK')* en Matlab

Decodifica la carga útil usando modulación QPSK, convirtiendo los símbolos de la constelación en bits.

- En GNU Radio *Repack Bits* por la línea repacked_bits = repack_bits(payload_bits, 8, 2) en Matlab

Reorganiza los bits de la carga útil en los bloques específicos, ajustando la longitud de los paquetes para verificar CRC.

- En GNU Radio *Stream CRC32* por la línea [crc_valid, received_data] = stream_crc32(repacked_bits) en Matlab

Verifica los datos en la carga útil mediante CRC32 y extrae los datos sin el CRC

- En GNU Radio *File Sink* por la línea *fid = fopen ('documento_recibido.txt', 'w'); fwrite(fid, received_data, 'char'); fclose(fid)* en Matlab

Es encargado de guardar el documento recibido

- En GNU Radio *Constellation Decoder (payload IQ)* por la línea *freq_spectrum = (fft (rx_signal_sync, nfft))* en Matlab

Visualiza la señal recibida en el dominio del tiempo y el espectro de la frecuencia de dicha señal

3.3.5 Evaluación y Análisis

3.3.5.1 Resultados en GNU Radio

Transmisión en GNU Radio

En el proceso de transmisión, se realizó primero el etiquetado de los datos. después se integró el bloque CRC32, que no solo agrega etiquetas, sino que también ajusta su longitud de 96 a 100 bytes, se continua el bloque generador de la cabecera, mientras prepara la carga útil para la modulación. La cabecera, está compuesta por una secuencia de bits 0 y 1, se puede visualizar en la figura 140.

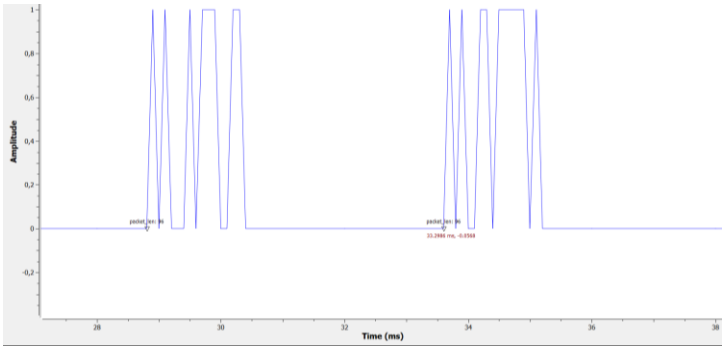


Figura 140. Cabecera generada

Fuente: Elaborado por autor

El proceso de empaquetado de carga útil convierte la señal en 4 niveles, como se muestra en la figura 141, este proceso es fundamental para preparar la señal antes de su transmisión, asegurando que tengan las características adecuadas para la modulación.

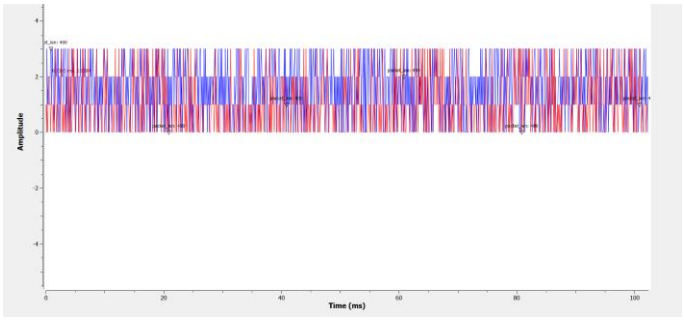


Figura 141. Empaquetado de carga útil

Fuente: Elaborado por autor

Una vez completado el empaquetamiento, la señal tanto de la cabecera como de la carga útil, se procesan para su modulación la parte de la cabecera corresponde a la modulación BPSK mientras para la carga útil la modulación QPSK, después de realizar las modulaciones las señales se multiplexan, generando un flujo de símbolos, como se muestra en la figura 142.

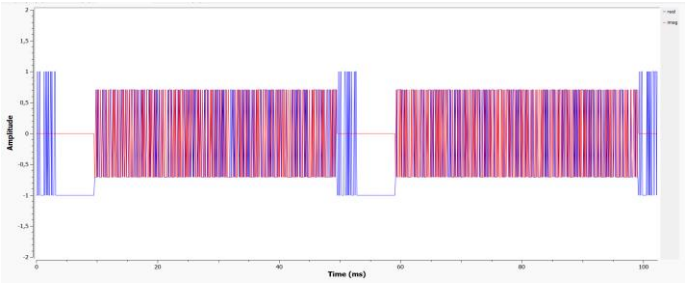


Figura 142. Flujo de símbolos en serie de la multiplexación de cabecera y carga útil

Fuente: Elaborado por autor

En la figura 143 se muestra el espectro de la frecuencia correspondiente a la señal OFDM a la salida de IFFT, donde se puede observar el ancho de banda ocupado es de 8 KHz y dos de banda guardia ubicados a los extremos ayudan a minimizar la interferencia y completando los 10KHz

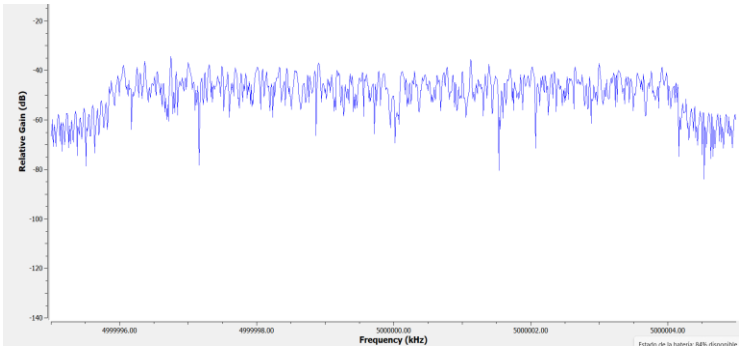


Figura 143. Espectro de la señal a transmitir

Fuente: Elaborado por autor

Mientras que la señal en el dominio del tiempo ya procesada y lista para la transmisión se encuentra en la figura 144.

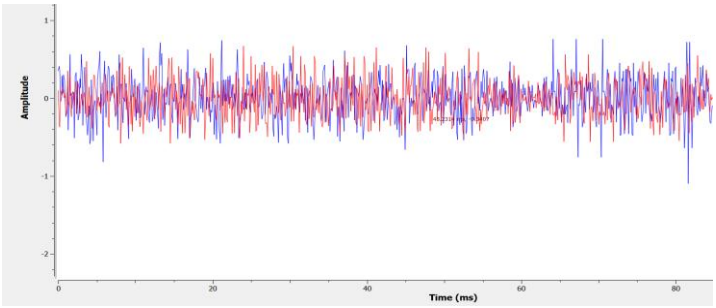


Figura 144. Señal trasmitida en el dominio del tiempo

Fuente: Elaborada por autor

Receptor en GNU Radio

La señal en el dominio del tiempo que ingresa al receptor se presenta en la figura 145, donde se puede apreciar que la señal ha sido atenuada durante su transmisión, afectando la calidad y capacidad del receptor en realizar una correcta demodulación.

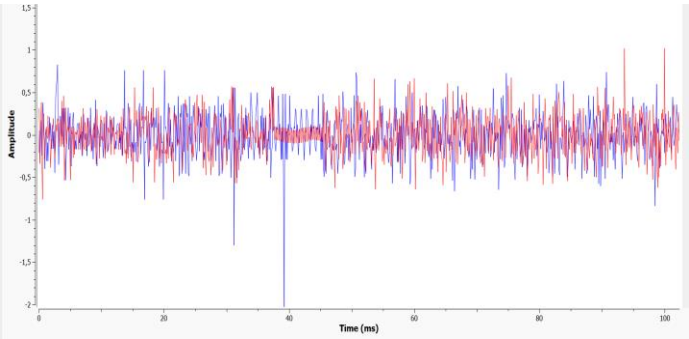


Figura 145. Señal recibida en el dominio del tiempo

Fuente: Elaborado por autor

A pasear de la posible atenuación durante la transmisión, es posible visualiza el espectro de la señal recibida, como se observa en la figura 146.

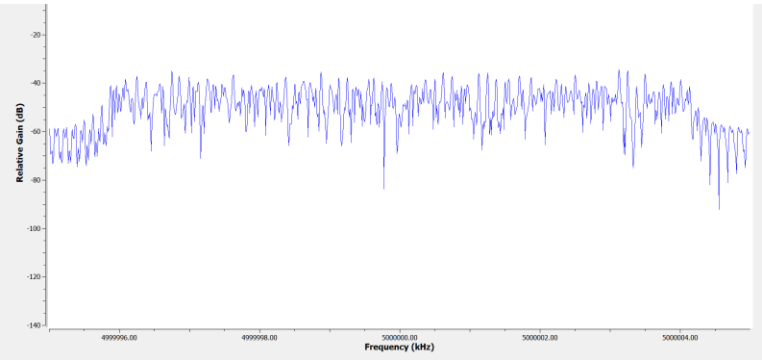


Figura 146. Espectro de señal recibida

Fuente: Elaborada por autor

3.3.5.2 Documento Transmitido y Recibido en GNU Radio

Para el desarrollo de la práctica, se envió un documento con un texto que se puede observar en el lado izquierdo de la figura 147. En el lado derecho, se presenta el documento recuperado en la recepción, donde se aprecian ciertos espacios y saltos de línea que no estaban presentes en el documento original, estas irregularidades pueden ser el resultado de la atenuación de la señal durante la transmisión, interferencias en el canal o errores en el proceso de demodulación. A pesar de estas alteraciones, el contenido principal del documento ha sido recuperado, lo que indica que el sistema de transmisión es, funcional. Pero se debe tener en cuenta, que es necesario mejorar la calidad de la señal para optimizar la integridad de los datos recibidos.

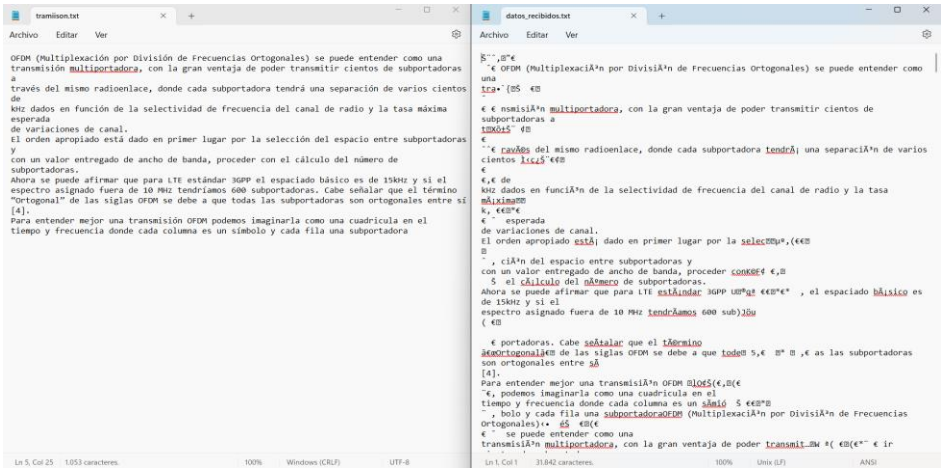


Figura 147. Documento trasmitido lado izquierdo y recibido lado derecho en GNU Radio

Fuente: Elaborado por autor

3.3.5.3 Resultados en Matlab

En la figura 148, en primera gráfica de la izquierda, se muestra la señal OFDM en el dominio del tiempo, donde la señal modulada en amplitud y fase presenta una apariencia característica con variaciones rápidas, resultado de la suma de múltiples subportadoras con diferentes fases y amplitudes, donde la combinación de las subportadoras en el dominio del tiempo genera una señal con cambios significativas debido a la superposición de ondas moduladas.

En la segunda gráfica de la derecha, se representa el espectro de frecuencia, donde se puede observar el ancho de banda ocupado por la señal y la distribución de potencia en cada subportadora. Las bandas de guarda en los extremos permiten minimizar la interferencia entre subportadoras, asegurando que cada una de ellas pueda ser demodulada adecuadamente en el receptor sin interferencias.

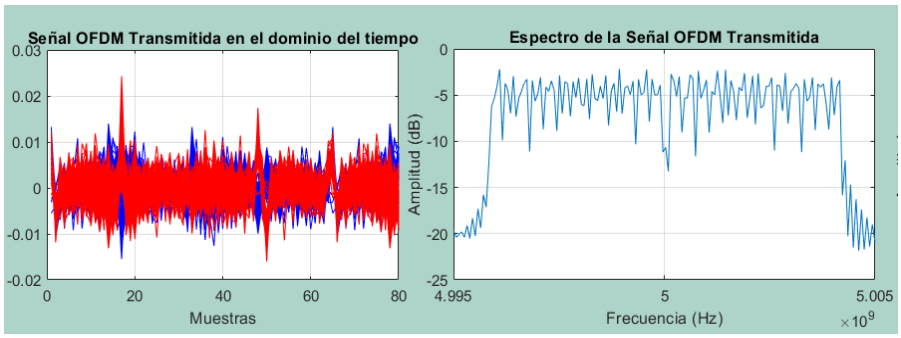


Figura 148. Señal trasmitida OFDM en el dominio del tiempo y espectro de la señal

Fuente: Elaborado por autor

En la figura 149, la gráfica de la izquierda corresponde a la señal final recibida en el dominio del tiempo muestra variaciones en amplitud, permitiendo visualizar la forma de la señal recibida, se observa posibles distorsiones o interferencias que pudieron afectar la transmisión.

La gráfica de la derecha correspondiente del espectro de frecuencia muestra la frecuencia de la señal recibida, esta información permite verificar si la señal recibió interferencias o si hubo distorsión en las frecuencias asignadas a las subportadoras, donde se observa que el espectro presenta distintas alteraciones indicando de problemas en la sincronización de la señal o en la configuración del receptor, afectando la decodificación correcta de los datos.

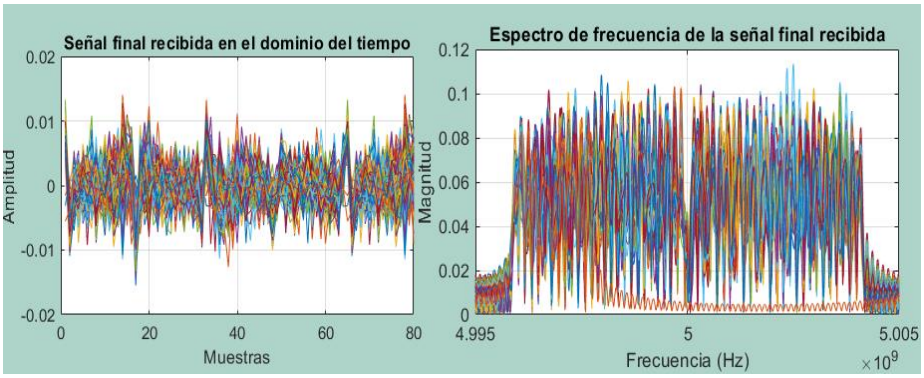


Figura 149. Señal recibida OFDM en el dominio del tiempo y espectro de la señal

Fuente: Elaborado por autor

3.3.5.4 Documento Transmitido y Recibido en Matlab

Al finalizar el proceso de recepción y decodificación, se intentó recuperar el texto transmitido, como se observa en la figura 150, sin embargo, el contenido recibido no se recuperó correctamente, esto puedo deberse a diferentes factores tales como errores en el sistema de corrección de errores, problemas de sincronización en cabecera, o dificultades en separación de subportadoras, dado que, Matlab emplea funciones específicas para su equipo SDR, en este código se intentaron replicar dichas funciones, pero el resultado no fue exitoso.

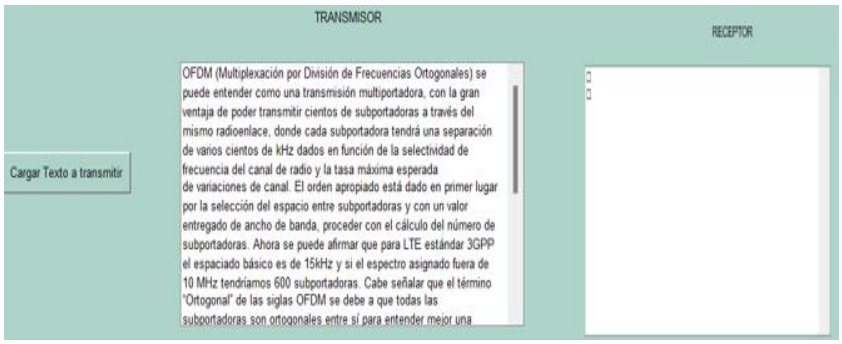


Figura 150. Documento trasmitido lado izquierdo y recibido lado derecho en Matlab

Fuente: Elaborado por autor

Hipótesis: La utilización de técnicas de multiplexación y la simulación en Matlab permiten optimizar la eficiencia de los sistemas de comunicación de banda ancha.

3.3.6 Conclusión de la práctica

Esta práctica demostró la importancia y viabilidad de estas tecnologías para la transmisión de datos, aprovechando los beneficios de la modulación OFDM, frente a interferencias y eficiencia espectral.

La implementación en GNU Radio, mostró resultados exitosos, logrando el propósito de transmitir y recibir un documento, a pesar de obtener pequeños errores, el documento se logró visualiza y cargar en la computadora, por otro lado, en Matlab, aunque se logró construir un sistema similar, la recuperación del texto no fue exitoso debido a que solo presentaba caracteres, aunque existió limitaciones en este software podrían resolverse con mejoras en la configuración de sincronización y en la implementación de la corrección de errores.

La práctica confirmó la importancia de los sistemas de banda ancha y su potencial en la transmisión de datos de manera segura y eficiente, aunque también los desafíos en la implementación y ajuste de estos sistemas para optimizar la integridad de los datos transmitidos y recibidos es fundamental para tener un mejor resultado.

CONCLUSIONES Y RECOMNEDACIONES

CONCLUSIONES

El uso de SDR permite vincular los conceptos teóricos con la práctica, demostrando la posibilidad de simular prototipos para transmisión en comunicaciones básicas a comunicaciones complejas como fue simular las características del estándar 802.11a con modulación OFDM utilizado en comunicaciones de banda ancha.

El desarrollo de los esquemas transmisor y receptor en Matlab, utilizando GUIDE así mismo como scripts y GNU Radio, mediante bloques, permitió comparar en ambos entornos en el procesamiento de señal, ambos Software demostraron ser eficaz en modulación y demodulación de las señales en las tres prácticas realizadas, teniendo en cuenta que cada uno tuvo ciertas ventajas sobre otro.

Se alcanzaron resultados satisfactorios en los sistemas implementados en las diferentes prácticas, sin embargo, en la tercera práctica la implementación en Matlab, resulto ser más compleja debido que el software cuenta con librerías específicas para su dispositivo SDR compatible, a pesar de que se intentó replicar estas librerías se obtuvo resultados erróneos en la recepción de datos.

RECOMENDACIONES

- Es recomendable utilizar los nombres correspondientes en gráficas a visualizar y así evitar conflictos en los resultados obtenidos.
- Tener en cuenta que se debe utilizar el bloque Throttle, debido a que permite no exceder en muestras y así evitar el consumo de recursos innecesarios del computador, mostrando lentitud o se quede sin responder el software de GNU radio.
- Tener en cuenta que se debe utilizar dos SDR uno como transmisor y el otro como receptor en la misma frecuencia de operación e identificando cada uno con su respectivo nombre para evitar confusiones en la trasmisión.
- Se recomienda tener todos los archivos necesarios en la misma carpeta caso contrario los scripts en Matlab, como el código en bloques de GNU Radio arrojaran errores en la simulación.

GUÍAS PRÁCTICAS

Práctica 1: Sistema de Comunicación Básica Modulación ASK

1.- Objetivos

Objetivo general

Comprender con la modulación ASK mediante la implementación de un transmisor y receptor usando el dispositivo SDR HackRF One.

Objetivos específicos

- Investigar los principios de la modulación ASK y sus aplicaciones en comunicaciones digitales.
- Implementar y modificar el diagrama de bloques proporcionado en GNU Radio para la transmisión y recepción de señales ASK.
- Utilizar Matlab para simular y validar los resultados obtenidos en GNU Radio comparando las señales obtenidas.

2.- Materiales e insumos

Componentes y dispositivos

- SDR HackRF One
- Dos computadoras con sistema operativo Linux o Windows.
- Antena compatible con el HackRF One

Software

- GNU Radio (versión 3.7.11 o superior).
- MATLAB

3. – Procedimiento

Parte A: Investigación y análisis previo

Investiga los principios de la modulación ASK, y responde las siguientes preguntas:

¿Qué es modulación ASK y cómo se diferencia de otras modulaciones?

¿Cómo se modulan las señales ASK, y que características definen su forma de onda?

Investiga cómo influye la frecuencia portadora y la amplitud en la transmisión de datos ASK.

Parte B. Implementación de la transmisión y recepción ASK en GNU Radio

1. Descargar archivos de código: Los archivos necesarios están en el siguiente enlace:

https://drive.google.com/drive/folders/15H5IY5KLJ5Izftj2wp01iaDTnGIzxeZt?q=sharedwith:public%20parent:15H5IY5KLJ5Izftj2wp01iaDTnGIzxeZt

2. Transmisión en GNU Radio:

Abra el archivo 'TRANSMISOR_ASK.grc' en GNU Radio

- Configure la fuente de datos para enviar una secuencia binaria de bits.
- Visualice la señal transmitida utilizando el bloque Scope Sink

Analiza y describe las características de la señal en frecuencia, amplitud y fase.

3. Recepción en GNU Radio:

Cargue 'RECEPTOR_ASK.grc' en GNU Radio, ajuste los parámetros de recepción para que coincida con la configuración de la trasmisión, asegurando una correcta decodificación de la señal.

Prueba el código proporcionado, y realiza un análisis de la señal recibida y compárala con la señal transmitida.

Parte C. Pruebas de modificación

Cambie la fuente de datos de un vector fijo (Vector Source) por datos de números aleatorios (Random Source), configure los siguientes parámetros:

- Minimum: 0
- Maximum: 256
- Num.Samples: 1K
- Repeat: yes

Después de realizar las modificaciones, vuelve a realizar las pruebas y compara los resultados obtenidos.

¿Cómo cambia la forma de onda y la amplitud de la señal en comparación con la secuencia fija?

Parte D. Simulación en MATLAB

- Ejecute el guide principal y seleccione práctica 1 en MATLAB.
- Coloque los números binarios que desea trasmitir.
- Visualice la señal transmitida y demodulada.
- Compare las señales generadas en Matlab con las de GNU Radio

Responda

¿Existen diferencias significativas en las formas de onda en ambos softwares? Justifique

Práctica 2: Modulaciones Digitales BPSK Y QPSK

1.- Objetivo General

Aplicar los principios de modulación BPSK y QPSK utilizando GNU Radio para la implementación del sistema y MATLAB para la verificación de resultados.

2. Objetivos Específicos

- Entender los conceptos importantes de la modulación digital.
- Implementar y modificar el diagrama de bloques proporcionado en GNU Radio para la transmisión y recepción.
- Utilizar MATLAB para simular y validar los resultados obtenidos de GNU Radio.
- Analizar y comparar el rendimiento de ambas modulaciones en términos de tasa de error de bit (BER) y constelaciones.

3.- Materiales e insumos

Componentes y dispositivos

- SDR HackRF One
- Dos computadoras con sistema operativo Linux o Windows.
- Antena compatible con el HackRF One

Software

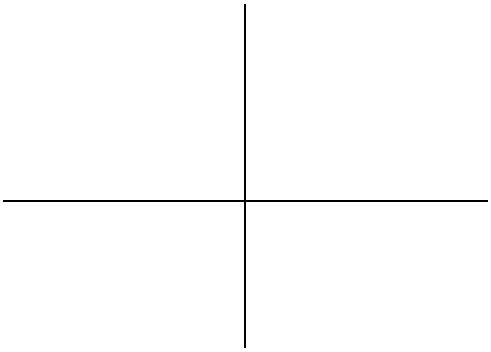
- GNU Radio (versión 3.7.11 o superior).
- MATLAB

4.- Procedimiento

Parte A: Investigación y análisis previo

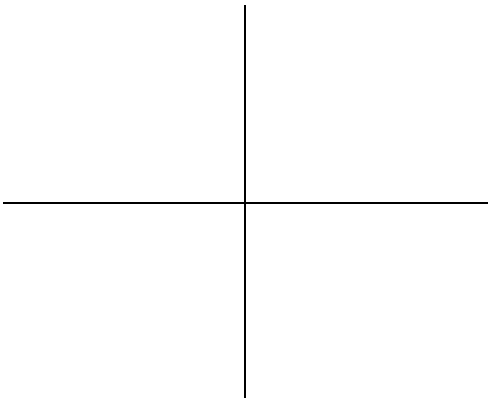
¿Qué es la modulación BPSK y cómo se representa?

Dibuje el diagrama de constelación BPSK



¿Qué es la modulación QPSK y cómo se representa?

Dibuje el diagrama de constelación QPSK



¿Qué es el Bit Error Rate (BER) y por qué es importante en los sistemas de comunicación?

Parte B: Implementación de la transmisión y recepción BPSK en GNU Radio

1. Descargar archivos de código: Accede a los archivos necesarios en el siguiente enlace:

https://drive.google.com/drive/folders/15H5IY5KLJ5Izftj2wp01iaDTnGIzxeZt?q=sharedwith:public%20parent:15H5IY5KLJ5Izftj2wp01iaDTnGIzxeZt

2. Transmisión BPSK en GNU Radio:

Paso 1: Abre el archivo 'TRANSMISOR_BPSK.grc' en GNU Radio.

Paso 2: Configura la fuente de datos:

- Utiliza un *Random Uniform Source* para generar una secuencia binaria.
- Establece el rango de bits mediante el vector [-1, 1] y luego define el número de muestras deseadas.

Paso 3: Verifica la configuración de los bloques de modulación BPSK, asegúrese de que la salida esté conectada a un *File Sink* para guardar la señal transmitida.

3. Recepción BPSK en GNU Radio:

Paso 1: Abre el archivo 'RECEPTOR_BPSK.grc'.

Paso 2: Carga el archivo generado por el transmisor en el bloque *File Source.*

- Asegúrate de que el formato y la tasa de muestreo coincidan con los del transmisor.

Paso 3: Visualiza la señal recibida utilizando el bloque *QT GUI Sink* para observar la señal en el dominio del tiempo y su constelación.

4. Tasa de error de bit (BER) en GNU Radio:

Paso 1: Abre el archivo 'Error_Bits_BPSK.grc'.

Paso 2: Carga el archivo generado por el transmisor y receptor en el bloque *File Source*

- Verifica que el formato coincida con los del transmisor y receptor, y que estén en la misma carpeta.

Analiza los resultados y en caso de tener perdidas de datos calcula cuantos datos se pierden

Parte C. Implementación de la transmisión y recepción QPSK en GNU Radio

1. Descargar archivos de código: Accede a los archivos necesarios en el siguiente enlace:

https://drive.google.com/drive/folders/15H5IY5KLJ5Izftj2wp01iaDTnGIzxeZt?q=sharedwith:public%20parent:15H5IY5KLJ5Izftj2wp01iaDTnGIzxeZt

2. Transmisión QPSK en GNU Radio:

Paso 1: Abre el archivo 'TRANSMISOR_QPSK.grc' en GNU Radio.

Paso 2: Configura la fuente de datos:

- Utiliza un Random Source con las siguientes configuraciones:
  - ✓ Mínimo: 0
  - ✓ Máximo: 256
  - ✓ Número de muestras: 10 K
  - ✓ Repetir: Sí

Paso 3: Verifica la configuración de los bloques de modulación QPSK, asegurándote de que la salida esté conectada a un *File Sink* para guardar la señal transmitida.

3. Recepción QPSK en GNU Radio:

Paso 1: Abre el archivo 'RECEPTOR_QPSK.grc'.

Paso 2: Carga el archivo generado por el transmisor en el bloque File Source:

- Asegúrate de que el formato y la tasa de muestreo coincidan con el transmisor.

Paso 3: Visualiza la señal recibida mediante el bloque *QT GUI Sink* para observar la señal en el dominio del tiempo y su constelación.

4.Tasa de error de bit (BER) en GNU Radio:

Paso 1: Abre el archivo 'Error_Bits_QPSK.grc'.

Paso 2: Carga el archivo generado por el transmisor y receptor en el bloque File Source:

- Asegúrate de que el formato coincida con los del transmisor y receptor, y que estén en la misma carpeta.

Analiza los resultados y en caso de tener perdidas de datos calcula cuantos datos se pierden

Parte D. Pruebas de modificación en BPSK

- Para bloque *packed to unpacked* cambia bits *per chunk* a 2

Investiga para que sirve el bloque y describe las modificaciones encontradas en la señal

Parte E. Pruebas de modificación en QPSK

- Sustituye el bloque *Random Source* por un *File Source*, asegurándote de que el documento tenga formato .txt.
- En lugar de utilizar el objeto de constelación, emplea una variable con la configuración *digital.constellation_qpsk().base()*

Analiza el comportamiento de esta señal y describe las modificaciones encontradas

Parte F. Simulación en MATLAB

- Ejecuta la guía principal y selecciona la práctica 2 en MATLAB.
- Elige el tipo de modulación BPSK o QPSK.
- Visualiza la señal transmitida y demodulada.
- Compara los resultados con los de GNU Radio en términos de:
  - ✓ Señal en el dominio del tiempo
  - ✓ Diagrama de constelación

Al comparar los resultados de GNU Radio y Matlab, ¿Qué similitudes y diferencias encuentras en la representación de las señales en ambas modulaciones?

Práctica 3: Sistema de comunicación de banda ancha basado en OFDM

1.- Objetivos

Objetivo general

Desarrollar transmisiones de banda ancha basadas en modulación OFDM, mediante la implementación de un transmisor y un receptor utilizando dispositivos de radio definida por software SDR.

Objetivos específicos

- Implementar un sistema de transmisión y recepción OFDM
- Evaluar el rendimiento del sistema mediante pruebas de modificación con diferentes fuentes de datos.

- Visualizar la señal transmitida y recibida en MATLAB, analizando su comportamiento en el dominio del tiempo y la frecuencia.

2.- Materiales e insumos

Componentes y dispositivos

- SDR HackRF One
- Dos computadoras con sistema operativo Linux o Windows.
- Antena compatible con el HackRF One

Software

- GNU Radio (versión 3.7.11 o superior).
- MATLAB

3. – Procedimiento

Parte A: Investigación y análisis previo

¿Qué es OFDM?

Investiga sobre la estructura de una trama de datos en sistemas cómo IEEE 802.11a y cómo se utiliza OFDM en estos estándares.

Parte B: Implementación de la transmisión y recepción OFDM en GNU Radio

1. Descargar archivos de código: Los archivos necesarios están en el siguiente enlace:

https://drive.google.com/drive/folders/15H5IY5KLJ5Izftj2wp01iaDTnGIzxeZt?q=sharedwith:public%20parent:15H5IY5KLJ5Izftj2wp01iaDTnGIzxeZt

2. Transmisión en GNU Radio:

El sistema se basa en una comunicación básica OFDM, utilizando características del estándar IEEE 802.11a, con las siguientes especificaciones:

-Tamaño de FFT: 64
- Número de subportadoras: 52 (48 para datos, 4 para pilotos)
- Longitud del prefijo cíclico: ¼
- Modulación de cabecera: BPSK
- Modulación de carga útil: QPSK

- Abre el archivo 'TRANSMISOR_OFDM.grc' en GNU Radio.
- Configura el bloque de *File Source* para enviar un archivo con los datos que deseas enviar. Asegúrate de que el archivo esté en la misma carpeta que el proyecto.

En el sistema OFDM propuesto, se utilizan 52 subportadoras, ¿Por qué se asigna 4 subportadoras para pilotos y qué importancia tiene en la trasmisión?

3. Recepción en GNU Radio:

- Carga el archivo 'OFDM_receptor.grc' en GNU Radio.
- Verifica que el formato y la tasa de muestreo en el receptor coincidan con los del transmisor para asegurar una correcta sincronización y procesamiento de la señal.

Observa y analiza la señal recibida en el dominio del tiempo y el espectro de la frecuencia y compara con la señal transmitida.

¿Se recibió el documento transmitido de manera correcta? Justifiqué.

Parte C: Pruebas de modificación

Enviar datos aleatorios sustituye el bloque *File Source* por *Random Source* con las siguientes configuraciones:

- ✓ Mínimo: 0
- ✓ Máximo: 255
- ✓ Número de muestras: 100
- ✓ Repetir: Sí

Indaga que impacto tiene la repetición de las muestras en el flujo de datos

Parte D: Simulación en MATLAB

- Ejecute el guide principal y seleccione práctica 3 en MATLAB.
- Seleccione el texto que desea trasmitir
- Visualice la señal en el dominio del tiempo y espectro de la frecuencia

¿Qué diferencias puedes notar entre las representaciones de Matlab y GNU Radio en las representaciones gráficas del dominio del tiempo y el espectro de frecuencia?

¿Qué sucede en la recepción de documento en Matlab, y cuáles son las posibles causas relacionadas en este efecto?

BIBLIOGRAFÍA

[1] Á. Pendás Recondo, “Diseño e implementación de un sistema de comunicaciones digitales adaptativas sobre dispositivos USRP,” jul. 2021, Accessed: Aug. 28, 2024. [Online]. Available: http://hdl.handle.net/10651/60640

[2] K. Navarro, F. Canto, and H. Poveda, “La Radio Definida por Software como Herramienta de Aprendizaje Educativa en Comunicaciones Inalámbricas,” in *Proceedings of the LACCEI international Multi-conference for Engineering, Education and Technology*, Latin American and Caribbean Consortium of Engineering Institutions, 2018. doi: 10.18687/LACCEI2018.1.1.115.

[3] N. E. Arevalo, “Transmisión y recepción de imagen con modulación QPSK usando radio definido por software,” Apr. 2017, Accessed: Sep. 11, 2024. [Online]. Available: http://hdl.handle.net/10654/16606

[4] I. Pinar Domínguez, J. José, and M. Fuentes, “Laboratorio de Comunicaciones Digitales Radio Definida por Software,” 2011. Accessed: Sep. 11, 2024. [Online]. Available: https://personal.us.es/murillo/librosdr/#:~:text=El%20resultado%20es%20un%20manual,%20que%20esperamos

[5] Botella-Mascarell *et al.*, “Evaluación del impacto del uso de dispositivos de radio definida por software como herramienta docente en la materia de comunicaciones digitales,” 2020. Accessed: Sep. 12, 2024. [Online]. Available: https://hdl.handle.net/10550/80120

[6] SDR En español, “Conceptos básicos.” Accessed: Sep. 24, 2024. [Online]. Available: https://sdr-es.com/conceptos-basicos/

[7] C. Parra, M. C. Paredes, G. Arévalo, and C. Tipantuña, “Prototipo de red GSM basada en SDR,” *Revista de Investigación en Tecnologías de la Información*, vol. 10, no. 21, pp. 24–39, Sep. 2022, doi: 10.36825/riti.10.21.004.

[8] B. Riyanto Trilaksono *et al.*, “Software Architecture of Software-Defined Radio (SDR).” Accessed: Sep. 22, 2024. [Online]. Available: https://www.researchgate.net/publication/268378153

[9] Jaime Rodríguez Calderón, “Investigación bibliográfica: ¿qué es y cómo hacerla?” Accessed: nov. 02, 2024. [Online]. Available: https://www.jaimerodriguez.mx/trabajos-academicos/investigacion-bibliográfica-que-es-y-como-hacerla/

[10] Lifeder, “Investigación experimental.” Accessed: nov. 01, 2024. [Online]. Available: https://www.lifeder.com/investigacion-experimental/

[11] “GNU Radio,” https://wiki.gnuradio.org/index.php/Main_Page.

[12] J. Pablo and M. Hidalgo, “Implementación de un sistema de comunicaciones basado en Software Radio,” Jan. 2014. Accessed: Sep. 24, 2024. [Online]. Available: http://hdl.handle.net/10486/660544

[13] Á. I. Monteros Túquerres, “Diseño y elaboración de prácticas de laboratorio para la materia de fundamentos de comunicaciones usando radio definida por software,” Quito, Mar. 2019. Accessed: Sep. 19, 2024. [Online]. Available: http://bibdigital.epn.edu.ec/handle/15000/20182

[14] Orientanet, “¿Qué es y para qué sirve el MATLAB?” Accessed: Sep. 20, 2024. [Online]. Available: https://www.orientanet.es/que-es-y-para-que-sirve-el-matlab/

[15] MATLAB & Simulink, “Introducción a Radio definida por software (SDR).” Accessed: Sep. 20, 2024. [Online]. Available: https://la.mathworks.com/discovery/sdr.html#:~:text=Con%20SDR%20(radio%20definida%20por%20software),%20MATLAB%20y%20Simulink%20%C2%AE

[16] “Great Scott Gadgets HackRF One.” Accessed: Sep. 21, 2024. [Online]. Available: https://greatscottgadgets.com/hackrf/one/

[17] E. Millard, C. Rene, D. Nayade, and P. Fidel, “Análisis de los parámetros teóricos de las señales de RF con el uso de SDR (HackRF One),” *Revista Ibérica de Sistemas e Tecnologías de Informação*, vol. 25, pp. 1–15, 2020, Accessed: oct. 02, 2024. [Online]. Available: https://www.researchgate.net/profile/Andres-Morocho-2/publication/340316114_Prediccion_de_demanda_de_energia_electrica_mediante_redes_neuronales_artificiales/links/5e83cf4b4585150839b2bf0d/Prediccion-de-demanda-de-energia-electrica-mediante-redes-neuronales-artificiales.pdf#page=14

[18] “Capítulo 3. Sistemas de Comunicaciones SISTEMAS DE COMUNICACIONES.” Accessed: Sep. 04, 2024. [Online]. Available: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/861/A6.pdf?sequence=6

[19] C. N. Chávez Gutiérrez, “Comunicación inalámbrica por radiofrecuencia, para control de iluminación de una residencia,” 2022. Accessed: Sep. 03, 2024. [Online]. Available: http://repositorio.umsa.bo/xmlui/handle/123456789/32871

[20] J. David, B. Lizcano, M. S. Alex, and A. M. Salcedo, “Desarrollo implementación de laboratorios de Radio Definido por Software RDS-para el laboratorio de telecomunicaciones de la UPB,” 2022. Accessed: oct. 03, 2024. [Online]. Available: https://repository.upb.edu.co/bitstream/handle/20.500.11912/10456/TG_RDS_EntregaFinal.pdf?sequence=1

[21] J. A. Vacacela Morales, “Implementación de un sistema SDR para modulación analógica y digital de señales AM, FM y M-PSK utilizando Matlab,” Guayaquil, Aug. 2018. Accessed: oct. 07, 2024. [Online]. Available: http://repositorio.ucsg.edu.ec/handle/3317/10952

[22] Jorge Arturo Alvarado Sánchez, “Estudio de las principales técnicas de modulación para el canal de radio,” México, Jan. 2005. Accessed: Sep. 05, 2024. [Online]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/alvarado_s_ja/capitulo2.pdf

[23] D. F. Chávez Cartagena, “Implementación de los esquemas de modulación BPSK y QPSK en Matlab/Simulink: estudio comparativo.,” 2017. Accessed: Sep. 03, 2024. [Online]. Available: http://repositorio.ucsg.edu.ec/handle/3317/7691

[24] J. A. Argüellos D., “Simulación del diagrama funcional de transmisión del sistema ISDB-T, para el estudio de su estructuración,” jun. 2017, Accessed: Sep. 03, 2024. [Online]. Available: http://hdl.handle.net/10872/14043

[25] V. A. Balseca Acuña, “Sistema de comunicaciones empleando SDR para prácticas multidisciplinarias de la Carrera de Ingeniería en Electrónica y Comunicaciones de la Universidad Técnica de Ambato,” Universidad Técnica de Ambato, 2021. Accessed: Oct. 11, 2024. [Online]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/32718

[26] Á. Pendás Recondo, “Diseño e implementación de un sistema de comunicaciones digitales adaptativas sobre dispositivos USRP,” jul. 2021. Accessed: oct. 11, 2024. [Online]. Available: http://hdl.handle.net/10651/60640

[27] Marco Suing, “Avances en Redes de Acceso de Banda Ancha,” Mar. 27, 2024. Accessed: oct. 19, 2024. [Online]. Available: https://telecomunicaciones.edu.ec/repositorio/articulos-blog/redes/avances-en-redes-de-acceso-de-banda-ancha

[28] “Llevar banda ancha de emergencia a cualquier lugar,” *Noticias De La Ciencia Y La Tecnología (Amazings® / NCYT®)*, nov. 2023, Accessed: oct. 19, 2024. [Online]. Available: https://noticiasdelaciencia.com/art/48672/llevar-banda-ancha-de-emergencia-a-cualquier-lugar

[29] “Familia IEEE 802.11.” Accessed: oct. 19, 2024. [Online]. Available: https://biblus.us.es/bibing/proyectos/abreproy/11579/fichero/f.+Cap%C3%ADtulo+2+-+Familia+IEEE+802.11.pdf+

[30] M. Cabrera and F. Tarrés, “Multiplexación por división en frecuencias ortogonales (OFDM).” Accessed: oct. 19, 2024. [Online]. Available: https://openaccess.uoc.edu/bitstream/10609/63345/2/Teor%C3%ADa%20de%20la%20codificaci%C3%B3n%20y%20modulaciones%20avanzadas_M%C3%B3dulo%205_Multiplexaci%C3%B3n%20por%20divisi%C3%B3n%20en%20frecuencias%20ortogonales(OFDM).pdf

[31] J. Salazar, “REDES INALÁMBRICAS.” Accessed: oct. 21, 2024. [Online]. Available: http://www.techpedia.eu

[32] R. E. Erreyes Cedeño and M. N. López Núñez, “Implementación de un prototipo de sistema MIMO-OFDM 2x2 basado en SDR mediante GNU Radio,” Quito, oct. 2019. Accessed: oct. 25, 2024. [Online]. Available: http://bibdigital.epn.edu.ec/handle/15000/20512

[33] F. V. Lincango Oña, “Implementación de un Prototipo de Sistema de Comunicación Inalámbrico OFDM en SDR,” Quito, oct. 2018. Accessed: oct. 26, 2024. [Online]. Available: http://bibdigital.epn.edu.ec/handle/15000/19827

[34] H. R. Cristian David and V. C. Luis David, “Prácticas de transmisión de datos en radiofrecuencia empleando dispositivos de radio definida por software,” 2023. Accessed: oct. 26, 2024. [Online]. Available: https://repository.udistrital.edu.co/server/api/core/bitstreams/c1feec3e-3b9d-42eb-907f-fd33c796a213/content

ANEXOS

Anexo 1 _ Transmisor ASK

```
% EL SIGUIENTE CODIGO CORRESPONDE AL TRASMISOR ASK, EL MISMO QUE SE DETALLO QUE LINEAS DE CODIGO
% QUE SE REMPLAZO POR CADA BLOQUE EN GNU RADIO PARA REALIZAR LA COMPARATIVA CORRESPONDIENTE

data = [1 0 1 1 0 1 0 0];  % Datos binarios a transmitir (vector random)

% Parámetros de la señal portadora (Signal Source)
samp_rate = 32000;   % Tasa de muestreo (Hz)
f_carrier = 1e6;     % Frecuencia de la portadora (Hz)

% Repetir los bits para interpolar (Repeat Interpolation)
interpolation_factor = 10;   % Factor de interpolación (repetición de bits)
t = (0:(length(data)*interpolation_factor)-1) / samp_rate;  % Tiempo
% ======== TRANSMISIÓN (Modulación ASK) =======:

% Interpolación de datos (repetir bits)
data_repeated = repelem(data, interpolation_factor);

% Convertir los datos a flotante
data_float = double(data_repeated);

% Generar señal portadora (senoidal)
carrier = sin(2 * pi * f_carrier * t);

% Modulación ASK: Multiplicar los datos por la portadora (Multiply)
modulated_signal = data_float .* carrier;

% Guardar la señal modulada en un archivo (File Sink)
fileID = fopen('modulated_ask_data.txt', 'w');
fprintf(fileID, '%f\n', modulated_signal);
fclose(fileID);

% Visualización de la transmisión
figure;
% Datos binarios transmitidos (representación de la señal en 1 y 0)
subplot(2, 1, 1);
stairs(t * 1e3, data_repeated, 'LineWidth', 2); % Representación en binario
ylim([-0.2 1.2]);
title('Datos Binarios Transmitidos');
xlabel('Tiempo (ms)');
ylabel('Valor Binario');
grid on;

% Señal modulada (Transmisión)
subplot(2, 1, 2);
plot(t * 1e3, modulated_signal);  % Tiempo en ms
title('Transmisión ASK');
xlabel('Tiempo (ms)');
ylabel('Amplitud');
grid on;
```

Anexo 2 _ Receptor ASK

```
% EL SIGUIENTE CODIGO CORRESPONDE AL RECEPTOR ASK, EL MISMO QUE SE DETALLO
% QUE LINEAS DE CODIGO SE REMPLAZO POR CADA BLOQUE EN GNU RADIO PARA REALIZAR
% LA COMPARATIVA CORRESPONDIENTE

%Cargar los datos modulados desde el archivo (File Source)
load('modulated_ask_data.txt');  % Cargar archivo con los datos modulados

% ======== RECEPCIÓN (Demodulación asK) ========
% Simular recepción de la señal (igual a la modulada en este caso)
received_signal = modulated_signal;

% Convertir a magnitud (absoluto) para detectar la señal (complex to mag)
received_magnitude = abs(received_signal);

% Aplicar un umbral simple para detectar los bits recibidos (threshold)
threshold = max(received_magnitude) / 2;
demodulated_data = received_magnitude > threshold;

% Decodificación de los bits recibidos
bits_received = demodulated_data(1:interpolation_factor:end);

% Subgráfica 3: Señal recibida y datos demodulados
subplot(1, 1, 1);
plot(t * 1e3, received_signal);  % Tiempo en ms
title('Recepción ASK');
xlabel('Tiempo (ms)');
ylabel('Amplitud');
grid on;
```

Anexo 3 _ Transmisor BPSK

```
% EL SIGUIENTE CODIGO CORRESPONDE AL TRASMISOR BPSK, EL MISMO QUE SE DETALLO
% QUE LINEAS DE CODIGO QUE SE REMPLAZO POR CADA BLOQUE EN GNU RADIO PARA
% REALIZAR LA COMPARATIVA CORRESPONDIENTE

% Parámetros
N = 100; % Número de bits a transmitir
sample_rate = 2.048e6; % Tasa de muestreo

% 1. Generar una secuencia de bits aleatorios
bits = randi([0 1], 1, N); % Genera bits aleatorios entre 0 y 1

% 2. Convertir los bits a flotantes (Short to Float)
signal_float = double(bits);

% 3. Modulación BPSK (Packed to Unpacked)
mod_signal = 2*signal_float - 1; % Convertir 0->-1, 1->1

% 4. Guardar los datos en un archivo (File Sink)
fileID = fopen('BPSK_output.txt','w');
fprintf(fileID, '%f\n', mod_signal);
fclose(fileID);

% 5. Visualización de la señal (QT GUI Time Sink)
figure;
plot(mod_signal);
title('Señal BPSK');

% 6. Diagrama de constelación (QT GUI Constellation Sink)
figure;
scatter(real(mod_signal), imag(mod_signal));
title('Diagrama de Constelación BPSK');
```

Anexo 4 _ Receptor BPSK

```
% EL SIGUIENTE CODIGO CORRESPONDE AL RECEPTOR BPSK, EL MISMO QUE SE DETALLO
% QUE LINEAS DE CODIGO QUE SE REMPLAZO POR CADA BLOQUE EN GNU RADIO PARA
% REALIZAR LA COMPARATIVA CORRESPONDIENTE

% Parámetros
fs = 5000;  % Frecuencia de muestreo (Throttle en GNU)
t = 0:1/fs:1;  % Vector de tiempo (1 segundo de duración)

% File Source - Cargar datos binarios desde el archivo
data_binary = load('BPSK_output.txt'); % Esto carga los datos en un vector
data_binary = data_binary(:)'; % Convierte a fila si es necesario

% Delay - Ajuste del número de datos
n_data_points = min(10, length(data_binary)); % Asegúrate de no exceder el número de datos
data_binary = data_binary(1:n_data_points); % Selecciona los primeros n datos

% Ajustar el tiempo y las muestras por bit
n_data_points = length(data_binary);
samples_per_bit = floor(length(t) / n_data_points);
t = t(1:n_data_points * samples_per_bit);

% Repetir cada valor de data_binary para coincidir con el tiempo
data_binary_repeated = repelem(data_binary, samples_per_bit);

% Complex to Mag (si la señal fuera compleja) y Threshold
signal_bpsk = data_binary_repeated;  % La señal ya está en [-1, 1]

% Modulación BPSK
fc = 10;  % Frecuencia de la portadora
bpsk_modulated = signal_bpsk .* sin(2 * pi * fc * t);  % Señal BPSK modulada

% Scope Sink - Visualización
figure;
subplot(2, 1, 1);
plot(t * 1000, bpsk_modulated, 'b', 'LineWidth', 1.5);  % Señal BPSK modulada
hold on;
data_binary_display = (signal_bpsk + 1) / 2;  % Convertir de [-1, 1] a [0, 1]
stairs(t * 1000, data_binary_display, 'r', 'LineWidth', 1.5);  % Datos binarios
title('Señal Recibida en BPSK');
xlabel('Tiempo (ms)');
ylabel('Amplitud');
ylim([-1.5 1.5]);
legend('Señal BPSK', 'Datos 0/1');
grid on;

% Constelación
subplot(2, 1, 2);
scatter(real(signal_bpsk), imag(signal_bpsk), 'bo');  % Puntos de la constelación
title('Constelación BPSK');
xlabel('Eje Real');
ylabel('Eje Imaginario');
xlim([-1.5, 1.5]);
ylim([-1.5, 1.5]);
grid on;
```

Anexo 5 _ Transmisor QPSK

```
% EL SIGUIENTE CODIGO CORRESPONDE AL TRANSMISOR QPSK, EL MISMO QUE SE DETALLO
% QUE LINEAS DE CODIGO QUE SE REMPLAZO POR CADA BLOQUE EN GNU RADIO PARA
% REALIZAR LA COMPARATIVA CORRESPONDIENTE

% Parámetros
M = 4; % Modulación QPSK (4 símbolos)
k = log2(M); % Bits por símbolo
nSamples = 100; % Número de muestras (10k)
minVal = 0; % Valor mínimo del random source
maxVal = 255; % Valor máximo del random source

% Generación de datos aleatorios (equivalente a Random Source en GNU Radio)
data = randi([minVal maxVal], nSamples, 1); % Secuencia aleatoria de 0 a 255

% Conversión de enteros a bits
data_bits = de2bi(data, 8, 'left-msb'); % Convertimos a bits (8 bits por símbolo)
data_bits = data_bits'; % Transponemos para tener una secuencia continua de bits
data_bits = data_bits(:); % Convertimos la matriz en vector de bits

% Empaquetado de bits (Unpack K Bits)
% Agrupamos los bits de 8 en 8
unpacked_bits = reshape(data_bits, k, []); % Agrupamos en bloques de 2 bits para QPSK

% Modulación QPSK (equivalente a Constellation Modulator en GNU Radio)
% Usamos la función qammod de MATLAB para la modulación
symbols = bi2de(unpacked_bits', 'left-msb'); % Convertimos los bits a símbolos
mod_signal = pskmod(symbols, M, pi/M); % Modulación QPSK (con desfase pi/M para simbolos)

% Guardar datos para el receptor
save('qpsk_transmitter_data.mat', 'mod_signal', 'data_bits', 'symbols', 'data');

% Gráfica de la constelación transmitida
scatterplot(mod_signal);
title('Constelación QPSK Transmitida');
grid on;

% Throttle - Controlamos la tasa de muestreo en MATLAB
Fs = 64e3; % Frecuencia de muestreo
t = (0:length(mod_signal)-1)/Fs; % Tiempo para la señal

% Scope - Graficar la señal en el dominio del tiempo
figure;
plot(t, real(mod_signal));
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal QPSK Transmitida en el Dominio del Tiempo');
grid on;
```

Anexo 6 _ Receptor QPSK

```
% EL SIGUIENTE CODIGO CORRESPONDE AL R QPSK, EL MISMO QUE SE DETALLO
% QUE LINEAS DE CODIGO QUE SE REMPLAZO POR CADA BLOQUE EN GNU RADIO PARA
% REALIZAR LA COMPARATIVA CORRESPONDIENTE

% Cargar datos del transmisor
load('qpsk_transmitter_data.mat', 'mod_signal');  % Señal modulada transmitida
% Parámetros
M = 4;  % Modulación QPSK (4 símbolos)
k = log2(M);  % Bits por símbolo
Fs = 64e3;  % Frecuencia de muestreo
numTaps = 15;  % Número de taps del ecualizador
% Throttle (Control de la tasa de muestreo en MATLAB)
nSamples = length(mod_signal);
t = (0:nSamples-1)/Fs;

% Polyphase Clock Sync (Sincronización de reloj)
sync = comm.SymbolSynchronizer('SamplesPerSymbol', 4, ...
                               'DampingFactor', 1, ...
                               'NormalizedLoopBandwidth', 0.01, ...
                               'TimingErrorDetector', 'Early-Late (non-data-aided)');

% CMA Equalizer (Ecualizador CMA) usando comm.LinearEqualizer
eq = comm.LinearEqualizer('Algorithm', 'CMA', 'NumTaps', numTaps, ...
                          'StepSize', 0.01, 'ReferenceTap', 1);

% Costas Loop (Recuperación de portadora)
carrierSync = comm.CarrierSynchronizer('Modulation', 'QPSK', ...
                                       'SamplesPerSymbol', 4, ...
                                       'DampingFactor', 0.707, ...
                                       'NormalizedLoopBandwidth', 0.01);

% Differential Decoder (Decodificador diferencial)
diffDecoder = comm.DifferentialDecoder;  % Eliminamos 'ModulationOrder'
```

```
% Recepción y procesamiento de la señal

% Sincronización de símbolos
synced_signal = sync(mod_signal);
% Aplicación del ecualizador
equalized_signal = eq(synced_signal);
% Recuperación de la portadora con el lazo de Costas
recovered_signal = carrierSync(equalized_signal);
% Actualizar el vector de tiempo para que coincida con recovered_signal
t_recovered = (0:length(recovered_signal)-1) / Fs;
% Demodulación QPSK
demodulated_symbols = pskdemod(recovered_signal, M, pi/M);
% Decodificación diferencial
decoded_data = diffDecoder(demodulated_symbols);

% Conversión de símbolos a bits
received_bits = de2bi(decoded_data, k, 'left-msb');
received_bits = received_bits';
received_bits = received_bits(:);  % Convertimos la matriz a un vector de bits

% Guardar los bits recibidos en un archivo txt
dlmwrite('received_qpsk_data.txt', received_bits, 'delimiter', '\t');

% Gráfica de la constelación recibida
scatterplot(recovered_signal);
title('Constelación QPSK Recibida');
grid on;

% Gráfica de la señal en el dominio del tiempo (parte real)
figure;
plot(t_recovered, real(recovered_signal));
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal QPSK Recibida en el Dominio del Tiempo (Parte Real)');
grid on;
```

Anexo 7 _ Transmisor OFDM

```
% Parámetros
FFT_length = 64;  % Longitud de la FFT utilizada en el sistema OFDM
occupied_carriers = [-26:-1, 1:26];  % Índices de las portadoras ocupadas en el esquema OFDM
pilot_carriers = [-21, -7, 7, 21];  % Índices de las portadoras de piloto
pilot_symbols = [1, 1, 1, -1];  % Símbolos de los pilotos
CP_length = 16;  % Longitud del prefijo cíclico
mod_order_header = 2;  % Orden de modulación para la cabecera (BPSK)
mod_order_payload = 4;  % Orden de modulación para la carga útil (QPSK)
const_mult = 50e-3;  % Multiplicador constante para ajustar la amplitud de la señal

% 1. Leer el archivo de texto
fileID = fopen('tramiison.txt', 'r');  % Abre el archivo de texto en modo lectura
text = fread(fileID, '*char')';  % Lee el contenido del archivo como caracteres
fclose(fileID);  % Cierra el archivo

% Convertir texto en bits
bits = reshape(de2bi(uint8(text), 8, 'left-msb')', 1, []);  % Convierte el texto en una secuencia de bits

function bits_crc = append_crc32(bits)
    % Agrega un CRC32 simulado (esta es una versión simplificada)
    crc = randi([0 1], 1, 32);  % Genera 32 bits de CRC aleatorios
    bits_crc = [bits, crc];  % Añade el CRC al final de los bits
end

% 2. Añadir CRC32 y ajustar longitud
bits_crc = append_crc32(bits);  % Llama a la función para simular el CRC32 y almacenar los bits con CRC
packet_len = length(bits_crc);  % Calcula la longitud del paquete después de añadir el CRC

% 3. Generar la cabecera y la carga útil
% Cabecera BPSK
header_bits = randi([0 1], 1, 96);  % Genera 96 bits aleatorios para la cabecera
header_symbols = pskmod(header_bits.', mod_order_header, pi);  % Modula la cabecera usando BPSK

% Empaquetamiento de la carga útil en QPSK
payload_bits = bits_crc;  % La carga útil son los bits con CRC
payload_symbols = qammod(payload_bits.', mod_order_payload, 'InputType', 'bit', 'UnitAveragePower', true);  % Modula la carga útil usando QPSK

% 4. Multiplexar cabecera y carga útil
tx_symbols = [header_symbols; payload_symbols];  % Combina los símbolos de la cabecera y la carga útil

% Relleno de tx_symbols para que su longitud sea un múltiplo de occupied_carriers
num_symbols = length(occupied_carriers);  % Número de símbolos ocupados
remainder = mod(length(tx_symbols), num_symbols);  % Calcula el residuo de la longitud de tx_symbols
if remainder ~= 0  % Si el residuo no es cero, se añade relleno
    tx_symbols = [tx_symbols; zeros(num_symbols - remainder, 1)];  % Rellena con ceros hasta el siguiente múltiplo
end

% 5. Asignación de portadoras y FFT
ofdm_symbols = zeros(FFT_length, ceil(length(tx_symbols)/num_symbols));  % Inicializa la matriz de símbolos OFDM
ofdm_symbols(occupied_carriers + FFT_length/2 + 1, :) = reshape(tx_symbols, num_symbols, []);  % Asigna los símbolos OFDM a las portadoras ocupadas

% Aplicar IFFT
tx_signal = ifft(ofdm_symbols, FFT_length);  % Realiza la IFFT para obtener la señal OFDM en el dominio del tiempo
```

```
% 6. Añadir prefijo cíclico
tx_signal_cp = [tx_signal(end-CP_length+1:end, :); tx_signal];  % Añade el prefijo cíclico a la señal

% 7. Multiplicador constante y guardar datos
tx_signal_cp = tx_signal_cp * const_mult;  % Ajusta la amplitud de la señal con el multiplicador constante

% Guardar la señal transmitida y la cabecera para el receptor
save('datos_transmision.mat', 'tx_signal_cp', 'header_bits');  % Guarda la señal y la cabecera en un archivo MAT
% Cargar la señal desde el archivo MAT
load('datos_transmision.mat', 'tx_signal_cp');  % Carga la señal transmitida desde el archivo

% Aplanar la señal para graficarla
tx_signal_cp_flat = tx_signal_cp(:);  % Convierte la señal en un vector unidimensional para graficar

% Graficar la señal en el dominio del tiempo
figure;
plot(real(tx_signal_cp_flat), 'b'); hold on;  % Grafica la parte real de la señal en azul
plot(imag(tx_signal_cp_flat), 'r');  % Grafica la parte imaginaria de la señal en rojo
title('Señal OFDM Transmitida con Prefijo Cíclico');  % Título de la gráfica
xlabel('Muestras');  % Etiqueta del eje x
ylabel('Amplitud');  % Etiqueta del eje y
legend('Parte Real', 'Parte Imaginaria');  % Leyenda de la gráfica
grid on;  % Activa la cuadrícula en la gráfica

sample_rate = 10e6;  % Tasa de muestreo ajustada para un ancho de banda de 10 MHz
carrier_frequency = 5e9;  % Frecuencia de portadora para la señal

% Calcular la FFT y convertir a dB
N = length(tx_signal_cp_flat);  % Longitud de la señal a ser transformada
tx_spectrum = (fft(tx_signal_cp_flat, N));  % Calcula la FFT de la señal transmitida
freq_axis = linspace(-sample_rate/2, sample_rate/2, N) + carrier_frequency;  % Ajusta el eje de frecuencia para centrar la portadora

% Graficar el espectro en dB
figure;
plot(freq_axis, 20*log10(abs(tx_spectrum)/max(abs(tx_spectrum))));  % Grafica el espectro en dB
title('Espectro de la Señal OFDM Transmitida');  % Título de la gráfica del espectro
xlabel('Frecuencia');  % Etiqueta del eje x
ylabel('Amplitud (dB)');  % Etiqueta del eje y
grid on;  % Activa la cuadrícula en la gráfica
```

Anexo 8 _ Receptor OFDM

```
% Cargar la señal transmitida desde el archivo guardado
load('datos_transmision.mat', 'tx_signal_cp'); % Se carga la señal transmitida (con prefijo cíclico) desde un archivo .mat.
rx_signal = tx_signal_cp;  % Usamos la señal transmitida como la entrada del receptor

% Parámetros de OFDM
FFT_length = 64; % Longitud de la FFT
CP_length = 16;  % Longitud del prefijo cíclico
occupied_carriers = -26:26;  % Portadoras ocupadas (índices de las portadoras en la FFT)
pilot_carriers = [-21, -7, 7, 21]; % Portadoras para los símbolos piloto
pilot_symbols = [1 1 1 -1]; % Símbolos piloto (valores a usar en las portadoras de piloto)
sync_word_1 = ones(1, FFT_length); % Palabra de sincronización 1 (todos los valores son 1)
sync_word_2 = -ones(1, FFT_length); % Palabra de sincronización 2 (todos los valores son -1)

% Sincronización (Schmidl & Cox)
rx_signal_sync = schmidl_cox_synchronization(rx_signal, FFT_length, CP_length);

% Función de sincronización Schmidl & Cox
function sync_signal = schmidl_cox_synchronization(rx_signal, FFT_length, CP_length)
    sync_signal = rx_signal; % Para esta demostración, solo se pasa la señal tal cual
end

% Demultiplexación de Cabecera/Payload
[header_signal, payload_signal] = header_payload_demux(rx_signal_sync, CP_length, FFT_length);

% Función de demultiplexación de cabecera y carga útil
function [header_signal, payload_signal] = header_payload_demux(rx_signal, CP_length, FFT_length)
    % Asegura que la cabecera tenga CP_length + FFT_length muestras
    header_signal = rx_signal(1:CP_length + FFT_length);  % Cabecera completa
    payload_signal = rx_signal(CP_length + FFT_length + 1:end);  % Resto como carga útil
end

% Procesamiento de la Cabecera
header_freq = fftshift(fft(header_signal(CP_length+1:CP_length+FFT_length), FFT_length));
header_data = ofdm_channel_estimation(header_freq, sync_word_1, sync_word_2);
header_equalized = ofdm_frame_equalizer(header_data, occupied_carriers);

% Función de estimación de canal
function est_channel = ofdm_channel_estimation(header_freq, sync_word_1, sync_word_2)
    % Estimación de canal usando palabras de sincronización
    est_channel = header_freq ./ (sync_word_1 + sync_word_2);
end

% Función de ecualización de trama OFDM
function equalized_data = ofdm_frame_equalizer(data, occupied_carriers)
    % Extrae y ecualiza las portadoras ocupadas
    equalized_data = data(occupied_carriers + length(data) / 2 + 1);
end

% Decodificación de la Cabecera (Modulación BPSK)
header_bits = constellation_decoder(header_equalized, 'BPSK');

% Función de decodificación de constelación
function bits = constellation_decoder(data, modulation)
    if strcmp(modulation, 'BPSK')
        bits = real(data) > 0; % Decodificación BPSK: 0 si la parte real <= 0, 1 si > 0
    elseif strcmp(modulation, 'QPSK')
```

```
        bits = reshape([real(data) > 0; imag(data) > 0], [], 1); % Decodificación QPSK
    else
        error('Modulación no soportada'); % Manejo de errores para otras modulaciones
    end
end

% Análisis de la Cabecera para obtener el tamaño del paquete
packet_length = packet_header_parser(header_bits);

% Función de análisis de cabecera
function packet_length = packet_header_parser(header_bits)
    % Interpreta los primeros bits como longitud del paquete
    packet_length = bi2de(header_bits(1:8), 'left-msb'); % Conversión de los primeros 8 bits a un número entero
end

% Procesamiento de la Carga Útil (Payload)
payload_freq = fftshift(fft(payload_signal(CP_length+1:CP_length+FFT_length), FFT_length));
payload_equalized = ofdm_frame_equalizer(payload_freq, occupied_carriers);

% Decodificación de la Carga Útil (Modulación QPSK)
payload_bits = constellation_decoder(payload_equalized, 'QPSK');

% Función para reempaquetar los bits
function repacked_bits = repack_bits(bits, in_bits, out_bits)
    % Convierte los bits a paquetes de longitud específica
    repacked_bits = reshape(bits(1:floor(length(bits)/out_bits)*out_bits), out_bits, []).'; % Reorganiza los bits en paquetes
end
```

```
% Función para verificación CRC
function [crc_valid, data] = stream_crc32(bits)
    % Verificación CRC32 simulada
    crc_valid = true; % Asumimos que es válido para esta demostración
    data = bits; % Devuelve los datos sin el CRC
end

% Desempaquetar y Verificación CRC
repacked_bits = repack_bits(payload_bits, 8, 2); % Reempaquetar los bits de la carga útil
[crc_valid, received_data] = stream_crc32(repacked_bits); % Verificación CRC

% Guardar el documento recibido
if crc_valid
    fid = fopen('documento_recibido.txt', 'w'); % Abre un archivo para escritura
    fwrite(fid, received_data, 'char'); % Escribe los datos recibidos en el archivo
    fclose(fid); % Cierra el archivo
    disp('Documento recibido y guardado.'); % Mensaje de confirmación
else
    disp('Error en la recepción del documento: CRC no válido.'); % Mensaje de error
end

% Graficar la señal final recibida en el dominio del tiempo
figure;
subplot(2, 1, 1);
plot(real(rx_signal_sync)); % Gráfico de la parte real de la señal sincronizada
title('Señal final recibida en el dominio del tiempo');
xlabel('Muestras');
ylabel('Amplitud');
grid on;

sample_rate = 10e6;    % Tasa de muestreo ajustada para 10 MHz de ancho de banda
carrier_frequency = 5e9;  % Frecuencia de portadora en 5 GHz

nfft = 1024; % Número de puntos para la FFT
subplot(2, 1, 2);
freq_spectrum = (fft(rx_signal_sync, nfft)); % Uso de fftshift para centrar el espectro en cero
freq_axis = linspace(-sample_rate/2, sample_rate/2, nfft) + carrier_frequency;  % Eje de frecuencia corregido con nfft
plot(freq_axis, abs(freq_spectrum)); % Gráfico del espectro de frecuencia
title('Espectro de frecuencia de la señal final recibida');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud');
grid on;
```

Anexo 9 _ Reporte Anti-plagio

CERTIFICADO DE ANÁLISIS
magister

Tesina_Katherine_Villon

7% Textos sospechosos

- 2% Similitudes
  - < 1% similitudes entre comillas
  - < 1% entre las fuentes mencionadas
- 3% Idiomas no reconocidos
- 3% Textos potencialmente generados por la IA

Nombre del documento: Tesina_Katherine_Villon.docx
ID del documento: 3e89df1c0ee9b99dbc677de7198bcc6065c265e6
Tamaño del documento original: 12 MB
Autores: []

Depositante: DANIEL ARMANDO JARAMILLO CHAMBA
Fecha de depósito: 20/11/2024
Tipo de carga: interface
fecha de fin de análisis: 21/11/2024

Número de palabras: 25.192
Número de caracteres: 165.068

Ubicación de las similitudes en el documento:

Fuentes principales detectadas

Nº	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	Documento de otro usuario #873dcf El documento proviene de otro grupo 7 fuentes similares	< 1%		Palabras idénticas: < 1% (133 palabras)
2	Documento de otro usuario #851c29 El documento proviene de otro grupo 5 fuentes similares	< 1%		Palabras idénticas: < 1% (110 palabras)
3	repository.udistrital.edu.co https://repository.udistrital.edu.co/bitstreams/c66f2852-aed6-44a1-af44-1a50235aae90/download	< 1%		Palabras idénticas: < 1% (24 palabras)

Fuentes con similitudes fortuitas

Nº	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	repository.udistrital.edu.co https://repository.udistrital.edu.co/bitstreams/1024bd7b-9736-46ac-a73d-3ecff990c23c/download	< 1%		Palabras idénticas: < 1% (33 palabras)
2	hdl.handle.net | Diseño e implementación de un sistema de comunicaciones digital... http://hdl.handle.net/10651/60640	< 1%		Palabras idénticas: < 1% (22 palabras)
3	repositorio.uta.edu.ec https://repositorio.uta.edu.ec/bitstreams/2ce39b57-0105-49a1-ba73-09c43867e0b4/download	< 1%		Palabras idénticas: < 1% (27 palabras)
4	Documento de otro usuario #9f23dc El documento proviene de otro grupo	< 1%		Palabras idénticas: < 1% (22 palabras)
5	hdl.handle.net | Evaluación del impacto del uso de dispositivos de radio definida po... https://hdl.handle.net/10550/80120	< 1%		Palabras idénticas: < 1% (19 palabras)

Fuentes mencionadas (sin similitudes detectadas)

Estas fuentes han sido citadas en el documento sin encontrar similitudes.

1. http://hdl.handle.net/10654/16606
2. https://sdr-es.com/conceptos-basicos/
3. https://www.researchgate.net/publication/268378153
4. https://www.lifeder.com/investigacion-experimental/
5. https://wiki.gnuradio.org/index.php/Main_Page