Agradecimientos

Nunca es tarea sencilla dar las gracias a tantas personas que han contribuido en mi beneficio. Aunque se han escrito muchas palabras de gratitud, en ocasiones el papel no puede plasmar a la perfección tanto afecto y admiración como siento en el término de tan fructífero proyecto.

Quiero mostrar mi agradecimiento a mi asesor externo el Dr. Eric Mario Silva Cruz, experto en el área de las telecomunicaciones, a mi asesor interno el M.C. Miguel Ángel Pérez Solano, por brindar sus conocimientos y apoyarme cuando lo he pedido. Asimismo, al Tecnológico Nacional de México Campus Oaxaca, que me ha abierto sus puertas y me ha permitido formar parte de un equipo de profesionales brillantes, como el Ing. Roberto Tamar Castellanos Baltazar y el M.E. Martin Vidal Reyes. Esta institución me ha enseñado a ser mejor estudiante y persona, a valorar cada uno de los esfuerzos que mis maestros del departamento de ingeniería electrónica, han hecho por mí y por la culminación de este proyecto.

Por su parte, me gustaría agradecer el apoyo incondicional de mi madre Marisa Elia Maldonado Nuñez, mi padre Salvador Sánchez Carrera y mi hermano Salvador Omar Sánchez Maldonado, quienes han puesto un antes y un después a lo largo de mi desarrollo académico y que me han ayudado a superar todos los obstáculos, grandes y pequeños, y me han animado a persistir. A su vez, he puesto mi empeño en este proyecto con la misma motivación y apoyo que he recibido tan generosamente de parte de mi amigo Carlos Ilescas Pérez, mi amiga Keila Reyes Colli, mi amigo Pablo Edgar López Meza y mi amigo German Reyes Osogobio, los cuales me han brindado sus conocimientos y acompañado en todo momento durante este bonito e inspirador viaje que es la Universidad. *El camino hacia la culminación de mi trabajo académico habría sido mucho más complicado y sinuosos sin el apoyo y la motivación que todos los individuos aquí nombrados.* Es para mí un honor haber contado con todos ustedes.

INSTITUTO TECNOLÓGICO DE OAXACA

DIVISIÓN DE ESTUDIOS PROFESIONALES

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

TESIS PROFESIONAL

DISEÑO E IMPLEMENTACIÓN DE UN MULTIPLEXOR SDR-OFDM PARA IOT Y 5G USANDO GNU RADIO Y TARJETAS DE SDR

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELECTRÓNICO

PRESENTA:

MIGUEL DE JESÚS SÁNCHEZ MALDONADO

ASESOR:

M.C. MIGUEL ÁNGEL PÉREZ SOLANO

COMISIÓN REVISORA:

DR. ERIC MARIO SILVA CRUZ
ING. ROBERTO TAMAR CASTELLANOS BALTAZAR
M.I. VICTOR MANUEL JIMENEZ RAMOS

OAXACA DE JUÁREZ, OAXACA **15 DE SEPTIEMBRE DE 2022**

Resumen

El objetivo de esta tesis es el diseño e implementación de un Multiplexor SDR-OFDM para IOT y 5G mediante la utilización de nuevas tecnologías como lo es el software GNU Radio y las tarjetas SDR (Software Defined Radio) USRP B200 y la HackRF One que por sus características permiten obtener los resultados deseados.

Actualmente estas tecnologías no se han tenido en cuenta por la falta de fuentes de información confiables y aquellas que lo son no se encuentran traducidas al español por el momento. Debido a eso, las fuentes de información utilizadas provienen del idioma inglés y el autor de esta tesis realiza las traducciones al español de la información útil para comprender el proceso de desarrollo.

Se ha recurrido a un diseño experimental que se aplica de manera longitudinal, considerando que el tema de investigación tiene un sustento teórico, se procede a realizar una investigación de tipo experimental con el fin de controlar deliberadamente las variables de las señales (frecuencia de transmisión, ganancia de recepción, ancho de banda y frecuencia de muestreo), y registrar los parámetros de la transmisión y recepción de los datos. La técnica de recolección de información utilizada es el “uso de documentos” ya que además de ayudar en la correcta estructuración de los métodos, permite conocer diversos aspectos históricos, contextuales y normativos necesarios.

Ahora bien, como resultados se consigue que mediante el uso de gráficos de flujo GRC se transmitan y reciban los datos en paquetes a través de la multiplexación por división de frecuencia ortogonal (OFDM). Además, se mostró el comportamiento de la señal OFDM transmitida con dos diferentes tipos de datos iniciales como lo es un archivo de audio (.wav) y un generador de números aleatorio. Finalmente, se comparó gráficamente la transmisión y recepción de la señal OFDM en el dominio del tiempo y en el dominio de la frecuencia.

La importancia de esta tesis es que aporta un medio de transmisión de una gran variedad de formatos de datos a través de softwares de licencia de código abierto y tarjetas de SDR por lo cual se reducen los gastos de tener que pagar licencias volviéndolo accesible para todos aquellos que estén interesados en replicar y mejorar el diseño del multiplexor. Además, cualquier modificación se realiza a través del mismo software por lo cual no es necesario comprar nuevos componentes de hardware como es el caso de los filtros electrónicos, codificadores y decodificadores digitales, multiplexores y demultiplexores, PLL externos, Relojes externos, etc., los cuales tienden a incrementar el costo de la implementación del proyecto.

Contenido

Tabla de contenido

Índice de cuadros, gráficas y figuras

Tabla de Figuras

Tabla de Tablas

1. Introducción

El avance de la tecnología requiere de la conectividad de dispositivos mediante nuevas tecnologías, tal es el caso de la Tecnología 5G, la cual permite a los sistemas de comunicaciones tener una gran variedad usos como lo es en el área del envío de datos y señales de sensores, como es el caso del Internet de las cosas (IOT) que ha tenido grandes aportaciones. De ahí que con las nuevas tecnologías basadas en 5G, se busque un aumento en el rendimiento y el desarrollo de una amplia gama de nuevas aplicaciones como lo es la Radio Cognitiva, Tecnologías Cloud para el acceso a redes de radio flexibles 5G, Conexión Dispositivo a Dispositivo, etc.

La Radio Cognitiva, es una tecnología de comunicaciones inalámbricas que con el pasar de los años se ha sometido a procesos de investigación, con el objetivo principal que a los dispositivos utilizados en telecomunicaciones se les pueda proporcionar cierto grado de inteligencia artificial con lo cual se puedan realizar tareas sencillas como la identificación de las secciones de espectro libre hasta detectar en su totalidad el espectro de telecomunicaciones. Además, mediante la asignación de tráfico a ciertos espacios del espectro electromagnético se puede garantizar una comunicación de calidad permitiendo a la vez la interconexión de una gran cantidad de dispositivos (López & Montejo, 2015).

Dicho lo anterior, en el departamento de Ingeniería Electrónica en colaboración con el departamento de ciencias básicas, se ha realizado la propuesta del proyecto de “Sismología y Riesgo Sísmico” para el Tecnológico Nacional de México, por lo cual dicho proyecto requiere la propuesta y diseño de una infraestructura de radiofrecuencia de múltiples sensores, derivado de esta necesidad el presente proyecto forma parte del área de diseño de instrumentación para solventar la necesidad de conectividad de datos en el proyecto antes mencionado.

1.1. Problemas a resolver

- Tener una infraestructura de radiofrecuencia con la capacidad de transmitir los datos obtenidos de múltiples sensores.
- Adquirir una infraestructura de radiofrecuencia con la capacidad de recibir en tiempo real los datos provenientes del transmisor.
- Establecer una compatibilidad de comunicación entre la tarjeta SDR HackRF One y la USRP B200.
- Distancia máxima de transmisión.
- Eficiencia espectral para mejorar el aprovechamiento de la banda de transmisión.
- Proporcionar una alta calidad en el servicio de transmisión y recepción.
- Reducir el consumo energético.
- Reducir el coste de instalación.
- Reducción de errores humanos.

1.2. Objetivos

1.2.1. Objetivo General

Diseñar e implementar un multiplexor SDR-OFDM para IOT y 5G usando el Software de código abierto GNU Radio y Tarjetas de SDR.

1.2.2. Objetivos particulares

- Identificar los elementos necesarios para llevar a cabo la modulación OFDM.
- Identificar los componentes necesarios para la transmisión OFDM utilizando múltiples portadoras.
- Identificar los componentes necesarios para llevar a cabo la recepción de la señal OFDM para múltiples portadoras.
- Programar el transmisor OFDM para múltiple portadora.
- Programar el receptor OFDM para múltiple portadora.
- Emplear la tarjeta de SDR USRP B200 como dispositivo transmisor.
- Emplear la tarjeta de SDR HackRF One como dispositivo receptor.
- Analizar los diferentes tipos de sincronización utilizadas para lograr la comunicación entre las tarjetas SDR HackRF One y USRP B200.
- Experimentar con las tarjetas de SDR para establecer la comunicación con la mayor calidad posible.
- Diseñar el sistema multiplexor SDR-OFDM con bajo consumo energético.
- Evaluar el funcionamiento del sistema multiplexor SDR-OFDM.

1.3. Justificación

Con el fin de contar con un sistema con características de transmisión y recepción a un bajo costo, se ha propuesto la opción de un sistema de transmisión RF que cumpla con las características necesarias para poder transmitir la información de los sensores sísmicos. De ahí que se ha optado como solución a esta problemática, el diseñar e implementar un Multiplexor SDR-OFDM para IOT y 5G usando GNU Radio y tarjetas de SDR, las cuales cuentan con las características necesarias para establecer un sistema de comunicación capaz de satisfacer la demanda de tráfico actual y que además proporcione una eficiencia energética, rendimiento, eficiencia espectral y calidad de servicio.

Para realizar este trabajo se ha recurrido a un diseño experimental que se aplicará de manera longitudinal, considerando que el tema de investigación tiene un sustento teórico, se ha decidido realizar una investigación de tipo experimental con el fin de controlar deliberadamente las variables de las señales (frecuencia de transmisión, ganancia de la antena transmisora y receptora, ancho de banda, ganancia de banda y frecuencia de muestreo), para así registrar los parámetros de transmisión y recepción de los datos. Ahora bien, la técnica de recolección de información apropiada a utilizar es el “uso de documentos” ya que es de ayuda para la correcta estructuración de los métodos y permite conocer diversos aspectos históricos, contextuales y normativos necesarios para llevar a cabo el diseño e implementación del proyecto.

Es así que, como resultado se presenta el diseño e implementación de un multiplexor SDR-OFDM para IOT en el entorno tecnológico de GNU Radio, utilizando el USRP B200 y la tarjeta SDR HackRF One como plataformas periféricas de radio de software universal, que proporcionan una cobertura de frecuencia de 70 MHz a 6 GHz y 1 MHz a 6 GHz respectivamente, permitiendo que se cumpla con los requisitos de rendimiento en tiempo real para su procesamiento adicional en el entorno de GNU Radio.

Personalmente, se ha optado por este proyecto proporcionado por el departamento de Ingeniería Electrónica, debido a que existen pocos trabajos en donde se realice la transmisión y recepción de datos a través de RF utilizando tarjetas de SDR y software de código abierto, como es en este caso con el uso de GNU Radio. Además, al existir poca información en el idioma español del uso de estas tarjetas de SDR con el software de GNU Radio para transmitir señales de OFDM, esta tesis tendrá mayor relevancia y será de ayuda para aquellos que deseen continuar con este proyecto.

Ahora bien, entre las ventajas podemos encontrar las facilidades para sistemas de comunicaciones y la versatilidad del software, permitiendo que se proporcionen nuevos y mejores servicios sin necesidad de cambiar terminales u otros equipos ya implementados ya que todo cambio ahora se realizara a nivel software. Asimismo, los servicios de telefonía serán de mayor calidad y más eficientes. Con respecto al punto de vista ecológico, se puede mencionar que los residuos industriales se verán reducidos debido a que SDR elimina en la mayoría de los casos el hardware.

2. Fundamento teórico

2.1. SDR

2.1.1. Definición

La radio definida por software (SDR), es un sistema de comunicación por radio donde los componentes que se han implementado tradicionalmente en hardware (mezcladores, filtros, amplificadores, moduladores/demoduladores, detectores, etc.) se implementan por medio de software en una computadora personal o sistema integrado (Montenegro et al., 2019).

Un sistema SDR básico puede consistir en una computadora personal equipada con una tarjeta de sonido u otro convertidor de analógico a digital, precediendo por alguna forma de extremo frontal de RF. Cantidades significativas de procesamiento de señales se entregan al procesador de propósito general, en lugar de realizarse en hardware de propósito especial (circuitos electrónicos). Tal diseño produce un radio que puede recibir y transmitir protocolos de radio muy diferentes basados únicamente en el software utilizado.

2.1.2. Ventajas

Las SDR tienen una utilidad significativa para los servicios militares y de telefonía celular, los cuales deben servir a una amplia variedad de protocolos de radio cambiantes en tiempo real. A largo plazo, los proponentes esperan que las radios definidas por software se conviertan en la tecnología dominante en las comunicaciones por radio.

2.1.3. Desventajas

El volumen de software descargado para algunos dispositivos reconfigurables llega a ser cada vez mayor y exige complejidad en los componentes, y como consecuencia el tiempo de descarga e instalación aumenta considerablemente.

2.2. GNU Radio

2.2.1. Definición

GNU Radio es un kit de herramientas de desarrollo de software libre y de código abierto que proporciona bloques de procesamiento de señales para implementar radios de software. Se puede utilizar con hardware de RF externo de bajo costo fácilmente disponible para crear radios definidos por software (SDR), o sin hardware en un entorno similar a la simulación. Es ampliamente utilizado en la investigación, la industria, la academia, el gobierno y los entornos de aficionados para apoyar tanto la investigación de comunicaciones inalámbricas como los sistemas de radio del mundo real ("GNU Radio", 2017).

2.2.2. Ventajas

- Como todos los sistemas de radio definido por software, la reconfigurabilidad es una característica clave.
- En vez de adquirir comercialmente varios tipos de radio, se puede adquirir una simple radio genérica la cual utiliza procesamientos de señal por software.
- Algunos transmisores y receptores pueden extender la cobertura a bandas entre 0 y 5.9 GHz.

2.2.3. Desventajas

- Es necesario descargar librerías externas para llevar a cabo diferentes procesos.
- La instalación de las librerías y paqueterías llegan a ser difíciles para aquellos que no están familiarizados con el entorno del programa.

2.3. Tarjeta SDR USRP B200 1X1

2.3.1. Descripción

El USRP B200 proporciona una plataforma periférica de radio de software universal totalmente integrada y de placa única con cobertura de frecuencia continua de 70 MHz a 6 GHz. Diseñado para la experimentación de bajo costo, combina un transceptor de conversión directa totalmente integrado que proporciona hasta 56 MHz de ancho de banda en tiempo real, un FPGA Sparta6 abierta y reprogramable. El soporte completo para el software UHD (USRP Hardware Driver) le permite comenzar inmediatamente a desarrollar con GNU Radio, crear prototipos de su propia estación base GMS con OpenBTS y realizar una transición perfecta del código B200 a plataforma USRP de mayor rendimiento y listas para la industria ("Ettus USRP B200: 1x1, 70MHz-6GHz SDR/Cognitive Radio", s.f.).

El soporte completo del software USRP Hardware Driver™ (UHD) permite la reutilización perfecta del código de los diseños existentes, la compatibilidad con aplicaciones de código abierto como HDSDR y OpenBTS, y una ruta de actualización a los sistemas USRP listos para la industria para cumplir con los requisitos de la aplicación. El frontend de RF integrado en el USRP B200 está diseñado con el dispositivo analógico AD9364, un transceptor de conversión directa de un solo chip y un procesador de banda base digital, capaz de transmitir hasta 56 MHz de ancho de banda de RF en tiempo real. El B200 utiliza una cadena de señal del AD9364, lo que le permite ser alimentado por bus y reduce la complejidad del diseño de software y hardware. El procesamiento y control de la señal a bordo del AD9364 se realiza mediante una FPGA Spartan-6 XC6SLX75 conectada a un PC host mediante SuperSpeed USB 3.0. El rendimiento del sistema en tiempo real USRP B200 se compara en cuadratura de 61.44MS/s, proporcionando los 56 MHz completos de ancho de banda de RF instantáneo al PC host para su procesamiento adicional utilizando el entorno de diseño GNU Radio SDR ("Ettus USRP B200: 1x1, 70MHz-6GHz SDR/Cognitive Radio", s.f.).

Tabla 1. Características de la tarjeta SDR USRP B200 1X1

Características	
Especificaciones de RF	• Xilinx Spartan-6 XC6SLX75 FPGA • Canales: 1 TX & 1 RX, Half or Full Duplex • Rango de frecuencia: 70 MHz a 6 GHz • Ancho de banda instantáneo: hasta 56 MHz • Supresión SSB/LO: .35/50 dBc ○ RMS de 3,5 GHz ○ RMS de 6 GHz • IIP3 (en NF típico): .20 dBm • Potencia de salida: >10 dBm
Rendimiento de conversión y relojes	• Frecuencia de muestreo ADC (máx.): 61.44 MS/s • Resolución ADC: 12 bits • ADC Wideband SFDR: 78 dBc • Frecuencia de muestreo DAC (máx.): 61.44 MS/s • Resolución DAC: 12 bits • Frecuencia de muestreo del host (16b): 61.44 MS/s • Precisión de frecuencia: ±2.0 ppm
Poder	• Alimentado por USB
Físico	• Orificios de montaje conectados a tierra • Interfaz USB 3.0 SuperSpeed • Conector USB 3.0 estándar • Dimensiones: 97 x 155 x 15 mm • Peso: 350 g
Ambiente	• Rango de temperatura de funcionamiento: 0 - 45 °C • API de GNU Radio, C++ y Python
Sincronización	• Referencia de reloj de 10 MHz • Referencia de tiempo PPS
Cumplimiento del producto	• HTC: 8471809000

Nota. Adaptado de *Ettus USRP B200: 1x1, 70MHz-6GHz SDR/Cognitive Radio*. Digilent. Consultado el 19 de enero de 2022, de https://digilent.com/shop/ettus-usrp-b200-1x1-70mhz-6ghz-sdr-cognitive-radio/.

Figura 1. Ettus USRP B200: 1X1, 70 MHz-6 GHz SDR/Radio Cognitiva.

Nota. *Adaptado de Ettus USRP B200: 1x1, 70MHz-6GHz SDR/Cognitive Radio* [Imagen], por Digilent, 2011, (https://digilent.com/shop/ettus-usrp-b200-1x1-70mhz-6ghz-sdr-cognitive-radio/).

2.4. HackRF One

2.4.1. Descripción

HackRF One es la plataforma de hardware actual para el proyecto HackRF. Es un periférico de radio por software capaz de transmitir o recibir señales de radio de 1 MHz a 6 GHz. Diseñado para permitir la prueba y el desarrollo de tecnologías de radio modernas y de próxima generación, HackRF One es una plataforma de hardware de código abierto que se puede utilizar como periférico USB o programado para funcionamiento independiente ("HackRF Radio definida por Software One SDR, kit de placa base de 1MHz a 6GHz, placa de desarrollo| Tablero de demostración|", 2022).

Figura 2. HackRF One.

Nota. *Adaptado de Raspberry Pi台灣樹莓派* [Imagen], por Raspberry Pi, 2018, (https://www.raspberrypi.com.tw/21095/hackrf-one-software-defined-radio/).

2.4.2. Características

Tabla 2. Características de la tarjeta SDR HackRF One.

- frecuencia de funcionamiento: 1 MHz a 6 GHz
- frecuencias de muestreo admitidas: 2 Msps a 20 Msps (cuadratura)
- resolución: 8 bits
- interfaz: USB de alta velocidad (con conector USB Micro-B)
- potencia del puerto de antena controlada por software (máx. 50 mA a 3,3 V)

2.5. Antenas

La antena es un transductor entre una onda guiada (cable, alambre y guía de ondas) y el espacio libre (espacio vacío) o viceversa. Hay muchos, tipos infinitamente variados son posibles. Por ejemplo, dipolo helicoidal, micro tira (en placa de circuito impreso), parche, bocina, parábola, etc, (*Modern Digital Radio Communication Signals and Systems, 2021)*.

2.5.1. Directividad y Ganancia

Aunque las características de radiación de una antena involucran patrones tridimensionales, muchas características de radiación importantes se pueden expresar en términos de cantidades de valor único, como el ancho de haz, el área del lóbulo principal, la directividad, la ganancia de la antena, la apertura, etc. Estos se explicarán a medida que avancemos.

El área del haz en ángulo sólido (ΩA) es una extensión del ángulo de arco en un círculo y se resume en la siguiente figura.

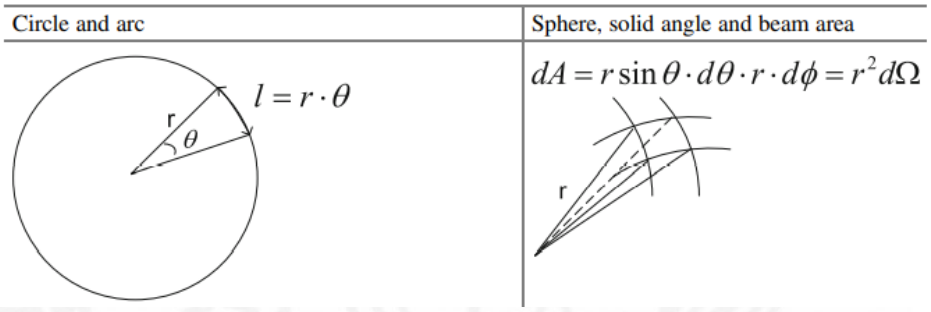

Figura 3. Extensión del ángulo de arco.

Nota. *Adaptado de Modern Digital Radio Communication Signals and Systems* (2ª ed., pp. 187-192), por Monn. S, 2021, Salmer.

2.5.2. Electrodepot 433 MHz Unity Gain Omni

Es una antena de alta calidad y duradera, omnidireccional de 15 cm de largo con cable coaxial premium de 60 pulgadas y conector SMA. Diseñada con una base magnética, puede colocarse en cualquier superficie de hierro horizontal o vertical.

Antena de transmisión o recepción inalámbrica que envía o recibe campos electromagnéticos y de radiofrecuencia (RF) en todas las direcciones horizontales en un plano geométrico, bidimensional.

Las antenas omnidireccionales se utilizan en la mayoría de los dispositivos inalámbricos RF de consumo, incluyendo teléfonos celulares y enrutadores inalámbricos.

2.5.2.1. Aplicaciones.

433 MHz es una banda de frecuencia comúnmente utilizada para todo tipo de equipos que requieren poca energía, como abridores de puertas de garaje, auriculares, algunos sistemas de comunicaciones, monitores de bebé y controles remotos.

Figura 4. Electrodepot 433 MHz Unity Gain Omni.

Nota. *Adpatado de Electrodepot 433 MHz Unity Gain Omni, antena de 6 pulgadas con base magnética y conector SMA macho – Impedancia 50 Ohmios* [Imagen], por Amazon, 2021 (https://www.amazon.com.mx/Electrodepot-Pulgadas-magn%C3%A9tica-Conector-impedancia).

2.5.3. Dioche BAOFENG Antena magnética de Radio

Hecho de material metálico de alta calidad, duradero en uso. Longitud del cable de hasta 3 metros. Alta sensibilidad y confiabilidad.

2.5.3.1. Características

- Material: Metal
- Modelo: UT-106UV
- Conector: SMA-M (macho)
- Frecuencia: UHF (430MHz-400 ~ 480MHz + VHF: 144MHz-136 ~ 174MHz)
- Ganancia: 3.0 dB/ 2.15 dB
- Impedancia: 50 ohmios
- Potencia máxima: 10W
- Longitud de la antena: 41 cm
- Peso: Aprox. 58 g/ 2 oz

Figura 5. Dioche Antena magnética de Radio.

Nota. https://www.amazon.com.mx/Dioche-magn%C3%A9tica-autom%C3%B3vil-frecuencia-dobleUT-106UV/dp/B07RXKZ291. (2021). [Imagen]. Consultado el 23 de enero de 2022, de https://www.amazon.com.mx/Dioche-magn%C3%A9tica-autom%C3%B3vil-frecuencia-dobleUT-106UV/dp/B07RXKZ291.

3. Marco Metodológico

3.1. Instalación del Software GNU Radio

Para este proyecto se utiliza la versión GR 3.8.2.0 la cual funciona con Python en su versión 3.9, compatible con el sistema operativo de Windows 10.

Ahora bien, para su instalación de forma binaria, se entra en la página http://www.gcndevelopment.com/gnuradio/index.htm en donde se pueden encontrar las versiones del software y se descarga.

Una vez terminada la descarga de la versión antes mencionada, solo es necesario ejecutarlo y dar clic en continuar. A continuación, se realiza la instalación del software GNU Radio y de Python en su versión 3.9, por lo cual es necesario dar clic en siguiente cuando se solicite, aceptar las condiciones del contrato o autorizaciones de Windows para que le permita hacer cambios en el equipo.

Por último, se hace clic en finalizar y ya tenemos el software GNU Radio instalado.

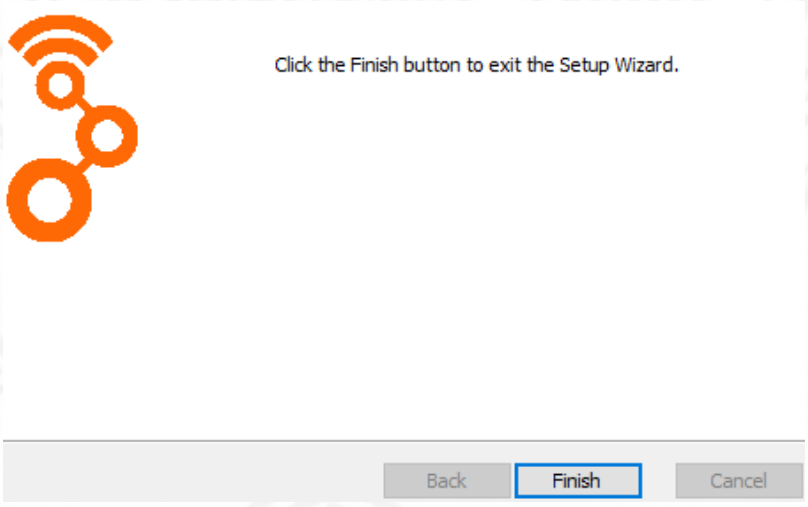

Figura 6. Instalación de GNU Radio 3.8 finalizada.

Nota. Elaboración propia en el Software GNU Radio.

3.1.1. Instalación de las Librerías de la Tarjeta USRP

3.1.1.1. Instalación de LibUSB.

Es un soporte de hardware basado en USB necesario para que la tarjeta USRP B200 sea detectada en el ordenador.

En primer lugar, se descarga la última versión (1.0.24) del siguiente enlace libusb - Explorar /libusb-1.0 en SourceForge.net.

Figura 7. Página en donde se puede descargar la última versión de LibUSB disponible.

Nota. libusb. (2008). *libusb Files* [Imagen]. Consultado el 23 de enero de 2022, de https://sourceforge.net/projects/libusb/files/libusb-1.0/.

A continuación, se descomprime el archivo. El directorio al que se extrae LibUSB no debe contener espacios. Es decir, eso causará problemas de compilación en el futuro, de ahí que, es recomendable utilizar la siguiente ubicación:

C:\local\libusb-1.0.24.

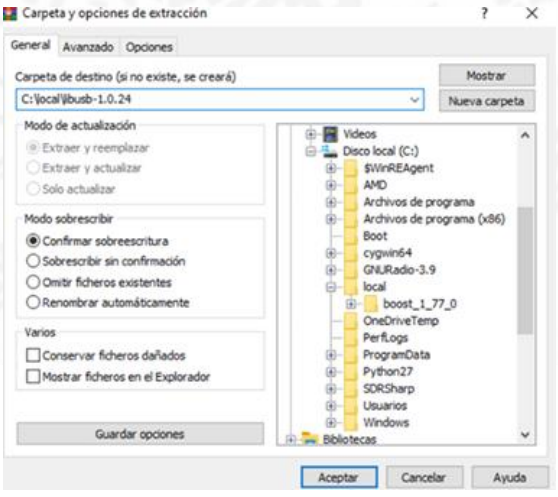

Figura 8. Directorio de extracción de LibUSB.

Nota. Elaboración Propia.

3.1.1.2. Instalación de UHD-4.1.0.4-Release.

Los paquetes de instalación se crean a partir de etiquetas de lanzamiento de la rama maint. Para esto es necesario ingresar a la página oficial de Ettus Research y descargar el paquete binario tal y como se muestra a continuación (figura 9).

Figura 9. Descarga del binario UHD en su versión 4.1.0.4.

Nota. Adaptado de *Ettus Research [Figura],* por Ettus Research, 2021, files.ettus.com (https://files.ettus.com/binaries/uhd/uhd_004.001.000.004-release/4.1.0.4/Windows-10-x64/).

A continuación, se ejecuta la aplicación como administrador.

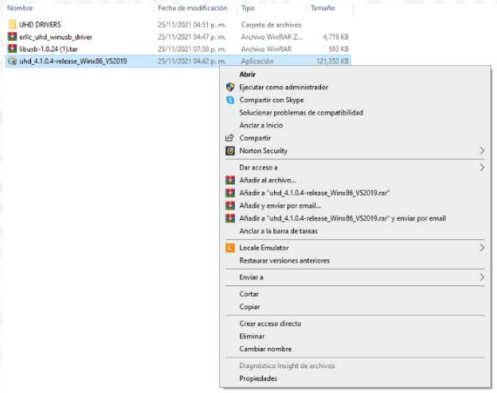

Figura 10. Ejecución del binario UHD.

Nota. Elaboración Propia.

Una vez se logre visualizar la ventana de instalación, se da clic en Next y se marca la opción “añadir UHD a la ruta del sistema para todos los usuarios” (add UHD to the system Path for all users) y se da clic en siguiente.

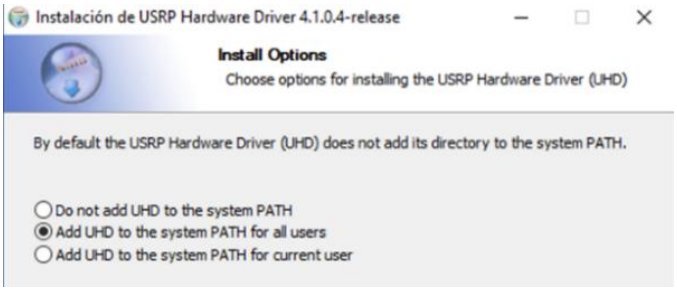

Figura 11. Instalación del binario UHD.

Nota. Elaboración Propia.

3.1.1.3. Instalación de los Controladores USB UHD.

Se debe descargar la última versión disponible a través del siguiente enlace http://files.ettus.com/binaries/misc/erllc_uhd_winusb_driver.zip.

Después, se extraen los archivos

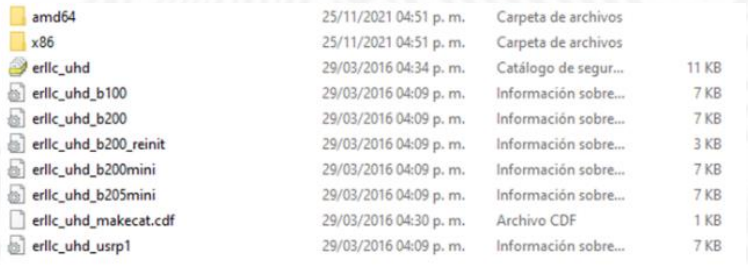

Figura 12. Extracción de los archivos de los controladores.

Nota. Elaboración Propia.

Se debe instalar el paquete de controladores para utilizar el producto basado en USB con software UHD tal y como se muestra a continuación.

- Abrir el administrador de dispositivos y conectar el dispositivo USRP

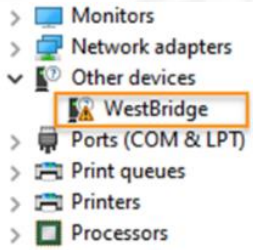

Figura 13. Controlador del dispositivo USRP.

Nota. Elaboración Propia.

- Hacer clic derecho en el dispositivo USB no reconocido y seleccionar actualizar/instalar controlador.
- En el asistente de instalación de controladores, seleccionar “buscar controlador” y seleccionar el archivo “directory.inf” dentro de la carpeta archivos de controlador.

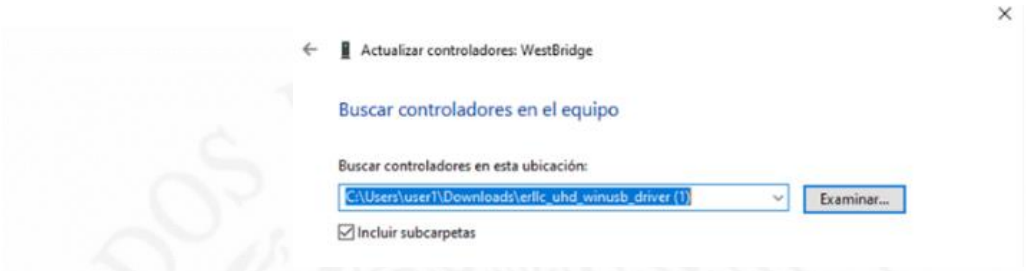

Figura 14. Búsqueda del controlador para el dispositivo USRP.

Nota. Elaboración Propia.

- Continuar por el asistente de instalación hasta que se el instalador este correctamente instalado.

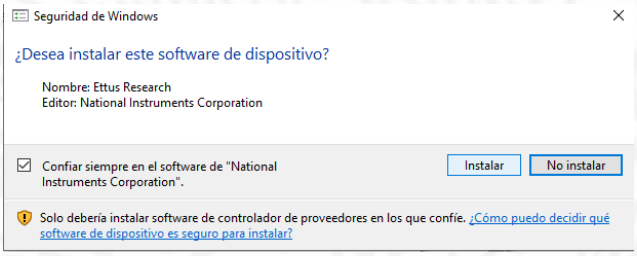

Figura 15. Instalación del controlador para el dispositivo USRP.

Nota. Elaboración Propia.

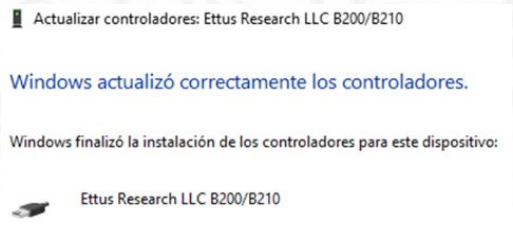

Figura 16. Controlador Ettus Research LLC B200/B210 instalado con éxito.

Nota. Elaboración Propia.

3.1.2. Instalación de la Paquetería de Radio Definido por Software

Esta paquetería debe ser instalada para que dentro el software GNU Radio pueda establecer la comunicación con la SDR HackRF One.

En primer lugar, se debe descargar el Paquete de radio definido por software de la página oficial de AIRSPY y descomprimir el archivo descargado sdrsharp-x86.

Figura 17. Descarga de la paquetería de radio definido por software.

Nota. Elaboración Propia.

Después, en la carpeta donde se extrajeron los archivos, se da clic en el archivo por lotes “install-rtlsdr”.

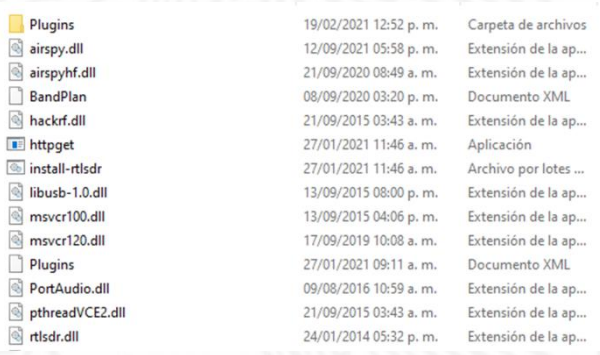

Figura 18. Ubicación del archivo por lotes “install-rtlsdr”.

Nota. Elaboración Propia.

A continuación, el archivo se ejecutará e instalará en el software GNU Radio la librería de OSMOCOM, la cual es utilizada para la transmisión, recepción y detección de la SDR HackRF One.

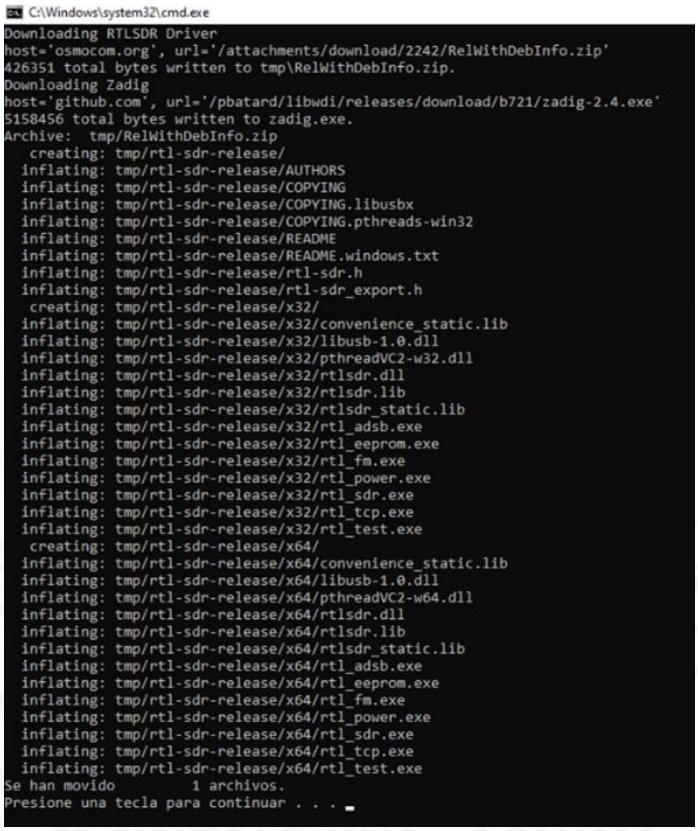

Figura 19. Instalación de la Librería OSMOCOM en GNU Radio.

Nota. Elaboración Propia.

3.1.2.1. Zadig

Zadig es una aplicación de Windows que instala controladores USB genéricos, como WinUSB, LibUSB-win32/LibUSB.sys o libusbk, para ayudarlo a acceder a dispositivos USB.

Puede se especialmente útil para los siguientes casos en que:

- Se necesite tener acceso a un dispositivo mediante una aplicación basada en libusb.
- Actualizar un controlador USB genérico.
- Tener acceso a un dispositivo mediante WinUSB.

Ahora bien, el programa puede descargarse en su página oficial.

Una vez descargado, es necesario ejecutar el programa como administrador. Al ser un programa portable no es necesario ninguna instalación para su uso.

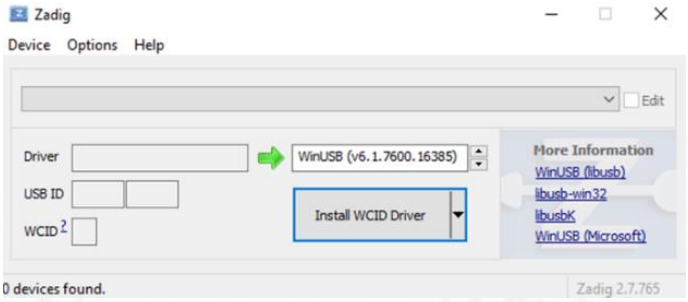

Figura 20. Software Zadig 2.7.

Nota. Elaboración Propia.

A continuación, se procede a instalar los drivers de la HackRF One de la siguiente manera.

- Conectar a través del puerto USB la HackRF One.

Figura 21. Tarjeta HackRF One.

Nota. Elaboración Propia.

- Dar clic en el apartado “Device” y después en “Create New Device”.

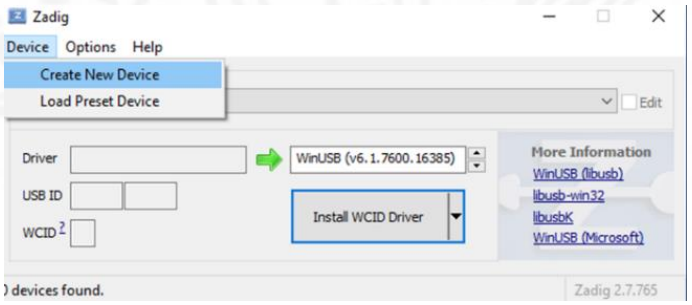

Figura 22. Detección del dispositivo HackRF One.

Nota. Elaboración Propia.

- Hacer clic en el apartado de “Options” y marcar “List All Devices”.

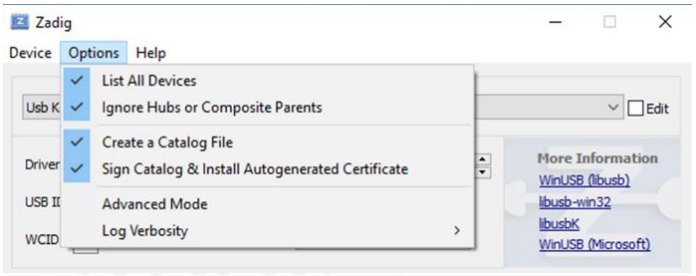

Figura 23. Búsqueda del dispositivo HackRF One.

Nota. Elaboración Propia.

- Desplegar el listado de los dispositivos conectados en la computadora y seleccionar HackRF One.

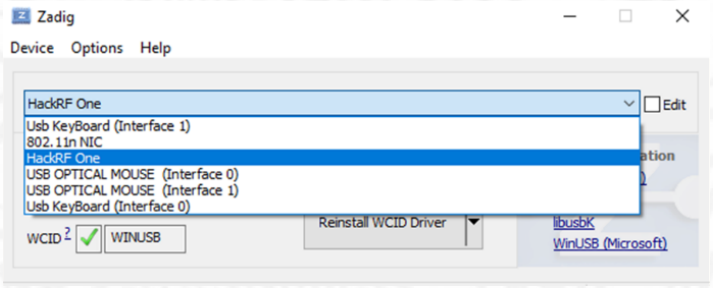

Figura 24. Selección del dispositivo HackRF One.

Nota. Elaboración Propia.

- Seleccionar como Driver “WinUSB” y dar clic en instalar WCID Driver.

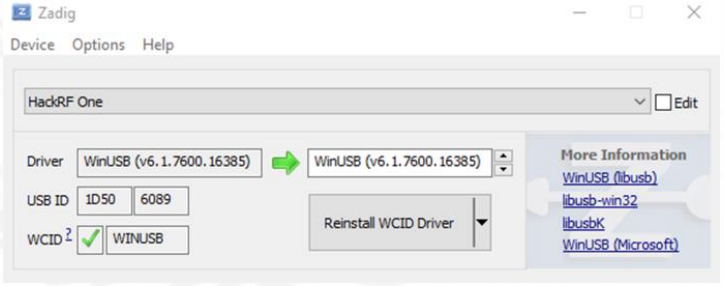

Figura 25. Instalación del driver WinUSB.

Nota. Elaboración Propia.

3.2. Normas IFT

3.2.1. Disposiciones Generales

Cada Estación de radiocomunicación o fuente emisora durante su despliegue y operación debe observar de manera obligatoria los límites de referencia de exposición máxima para seres humanos a radiaciones electromagnéticas de radiofrecuencia no ionizantes establecidos en la Tabla 3 considerando la Distancia de cumplimiento (Instituto Federal de Telecomunicaciones, 2018).

Tabla 3. Límites de referencia de exposición máxima.

Tipo de exposición	Intervalo de frecuencias	Intensidad de campo eléctrico (E) [V/m] (valor eficaz)	Intensidad de campo magnético (H) [A/m] (valor eficaz)	Densidad de potencia de onda plana equivalente (S_{eq}) [W/m^2]
Público en general	100 kHz-150 kHz	87	5	–
	0.15 MHz-1 MHz	87	$0.73 / f$	–
	1 MHz-10 MHz	$87 / f^{\frac{1}{2}}$	$0.73 / f$	–
	10 MHz-400 MHz	28	0.073	2
	400 MHz-2 000 MHz	$1.375\, f^{\frac{1}{2}}$	$0.0037\, f^{\frac{1}{2}}$	$f / 200$
	2 GHz-300 GHz	61	0.16	10

Notas:

1. f es la frecuencia expresada en las unidades indicadas en la columna de intervalo de frecuencias.
2. Para frecuencias entre 100 kHz y 10 GHz, los valores de E^2, H^2 y de la Densidad de potencia equivalente de onda plana (S_{eq}) deben ser promediados sobre cualquier periodo de 6 minutos.
3. Todos los valores de la tabla son valores RMS.
4. Para frecuencias de 100 kHz, los valores pico permitidos son los que resultan de multiplicar los valores RMS que aparecen en la tabla por $\sqrt{2}$ (≈ 1.414).
5. Para frecuencias mayores a 10 GHz, los valores de E^2, H^2 y de la Densidad de potencia equivalente de onda plana (S_{eq}) deben ser promediados sobre cualquier periodo de $68 / f^{1.05}$ min. Con f en GHz.

Nota. Adaptado de *INSTITUTO FEDERAL DE TELECOMUNICACIONES [Tabla]*, 2019, IFT.

3.2.2. Estaciones de Radiocomunicación o Fuentes Emisoras Inherentemente Conformes.

I. Las Estaciones de radiocomunicación o fuentes emisoras que tengan una PIRE de 2 Watts o menor se consideran inherentemente conformes9. Este tipo de Estaciones de radiocomunicación o fuentes emisoras producen campos que cumplen con los límites máximos de exposición a unos centímetros de la Antena; por lo tanto, no se requieren precauciones particulares.

II. Las Estaciones de radiocomunicación o fuentes emisoras inherentemente conformes deberán ser registradas en la base de datos que para tal efecto disponga la Unidad de Concesiones y Servicios del Instituto.

3.2.3. Normativa

De conformidad con lo dispuesto en el Artículo 54 de la Ley Federal de Telecomunicaciones y Radiodifusión (LFTyR), corresponde al Instituto la administración del espectro radioeléctrico en beneficio de los usuarios, siguiendo las recomendaciones de la Unión Internacional de Telecomunicaciones y otros organismos internacionales.

Por su parte, la fracción II del artículo 55 de la LFTyR, establece lo siguiente:

"Artículo 55. Las bandas de frecuencia del espectro radioeléctrico se clasificarán de acuerdo con lo siguiente: (…)

II. Espectro libre: Son aquellas bandas de frecuencia de acceso libre, que pueden ser utilizadas por el público en general, bajo los lineamientos o especificaciones que establezca el Instituto, sin necesidad de concesión o autorización; (…)"

De lo anterior se advierte que el espectro libre puede ser empleado por cualquier usuario, sin necesidad de contar particularmente con un instrumento habilitante para el uso del espectro, siempre y cuando se acaten las condiciones de operación que se establecen a través de los instrumentos regulatorios correspondientes (Instituto Federal de Telecomunicaciones, 2019).

3.2.4. Notas Nacionales

MX134. La banda de 410-430 MHz se tiene prevista para la provisión del servicio móvil de radiocomunicación especializado de flotillas. El segmento 410 – 415/420 – 425 MHz se destina a operaciones de uso comercial, mientras que el segmento 415 – 240/425 - 430 MHz se destina a operaciones de uso público (Instituto Federal de Telecomunicaciones, 2019).

3.3. OFDM

3.3.1. Introducción a OFDM

3.3.1.1. Transmisión de una sola portadora.

3.3.1.1.1. Transmisión de Banda en una Sola Portadora: Modelo de Sistema.

La Figura 26 muestra una configuración típica de un extremo a otro para un sistema de comunicación de una sola portadora.

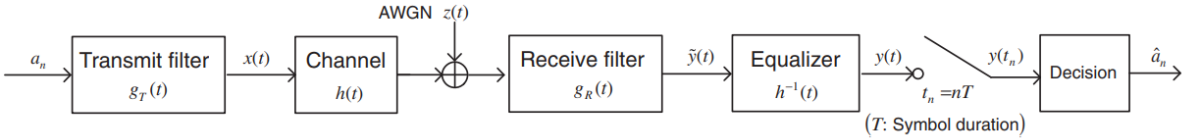

Figura 26. Modelo de sistema de comunicación de banda base de portadora única.

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Figura], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 130). John Wiley & Sons.

Considere un canal de banda limitada h (t) con un ancho de banda de W. Los símbolos de transmisión $\{a_n\}$, cada uno con un período de símbolo de T segundos, es decir, una tasa de datos de $R = 1/T$, tienen una forma de pulso por un filtro transmisor $gT\ (t)$ en el transmisor. Después de recibirlos a través del canal, son procesados con el filtro de recepción, el ecualizador y el detector en el receptor. Dejando a $gT\ (t)$, $gR\ (t)$ y $h^{-1}(t)$, denotar la respuesta de impulso del filtro de transmisión, filtro de recepción y ecualizador, respectivamente (*MIMO-OFDM Wireless Communications with MATLAB*, 2010). La salida del ecualizador se puede expresar como:

$$y(t) = \sum_{m=-\infty}^{\infty} a_m g(t - mT) + z(t) \qquad (1)$$

Donde z(t) es una ruido aditivo y g(t) es la respuesta al impulso del sistema general de extremo a extremo dado como

$$g(t) = g_T * h(t) * g_R(t) * h^{-1}(t) \qquad (2)$$

El ecualizador está diseñado para compensar el efecto de canal. En esta sección, simplemente asumimos que el efecto del canal está perfectamente compensado por el ecualizador como se indica en la Ecuación (2). Por lo tanto, la respuesta general al impulso está sujeta a filtros de transmisión y recepción únicamente. Cuando se ignora el término de ruido, la señal de salida muestreada del ecualizador se puede expresar como:

$$y(t_n) = \sum_{m=-\infty}^{\infty} a_m g((n-m)T) \quad donde \quad t_n = nT \qquad (3)$$

Aislando la enésima muestra para detectar a_n, la ecuación (3) se puede escribir como

$$y(t_n) = a_n g(0) + \sum_{m=-\infty, m\neq n}^{\infty} a_m g((n-m)T) \qquad (4)$$

3.3.1.1.2. ISI y Criterio de Nyquist.

ISI: Interferencia entre símbolos (Inter-symbol interference)

En la ecuación (4), ISI puede ser completamente eliminado mediante el cumplimiento de la siguiente condición en el dominio del tiempo en la respuesta de impulso general:

$$g(nT) = \delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \qquad (5)$$

Tenga en cuenta que la condición de la ecuación (5) es equivalente a la siguiente condición en el dominio de la frecuencia:

$$\sum_{i=-\infty}^{\infty} G\left(f - \frac{i}{T}\right) = T \qquad (6)$$

Donde G (f) es la transformada de Fourier de g (t), que representa la respuesta general en frecuencia. La condición Ecuación (5) o Ecuación (6) se conoce como el criterio de Nyquist, que garantiza una comunicación libre de ISI (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

Tenga en cuenta que los filtros que satisfacen el criterio de Nyquist se denominan filtros de Nyquist. Un filtro Nyquist obvio es un LPF (filtro pasa bajas) ideal, que tiene un tipo de función sinc de respuesta de impulso o, de manera equivalente, un tipo de respuesta de frecuencia de pulso rectangular (o pared de ladrillo) como se describe por:

$$G_I(f) = \frac{1}{2W} rect\left(\frac{f}{2W}\right) = \begin{cases} T, & |f| \le \frac{1}{2T} \\ 0, & |f| > \frac{1}{2T} \end{cases} \qquad (7)$$

Donde $W = \frac{R}{2} = 1/(2T)$. En la ecuación (7), R y W corresponden a la tasa de Nyquist y Ancho de banda de Nyquist, respectivamente. Tenga en cuenta que el ancho de banda de Nyquist W es el ancho de banda mínimo posible que se requiere para realizar la tasa de datos R sin ISI. Sin embargo, el filtro ideal en la ecuación (7) no se puede lograr físicamente porque su respuesta al impulso no es casual.

Otro filtro Nyquist bien conocido, ahora físicamente realizable, es el filtro de coseno alto, que se especifica mediante la siguiente respuesta en frecuencia:

$$G_{RC}(f) = \begin{cases} T, & |f| \le \frac{1-r}{2T} \\ \frac{T}{2}\left\{1 + \cos\frac{\pi T}{r}\left(|f| - \frac{1-r}{2T}\right)\right\}, & \frac{1-r}{2T} < |f| \le \frac{1+r}{2T} \\ 0, & |f| > \frac{1+r}{2T} \end{cases} \qquad (8)$$

Donde r es el factor de caída que adapta el ancho de banda total $0 \le r \le 1$. Está claro que la ecuación (8) satisface la condición libre de ISI de la ecuación (6), pero no es tan nítida como la respuesta de frecuencia de un LPF ideal. Tenga en cuenta que la respuesta de frecuencia del coseno elevado en la ecuación (8) ocupa un rango de frecuencia más amplio que el ancho de banda de Nyquist. El ancho de banda real se rige por el factor de caída r (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

Las figuras 27 (a1) y (a2) muestran las respuestas de impulso y frecuencia de los filtros de coseno elevado con los factores de caída de r = 0, 0.5 y 1, respectivamente.

Tenga en cuenta que el filtro de coseno elevado con r = 0 resulta ser idéntico al LPF ideal, y el filtro de coseno elevado con r = 1 ocupa el doble del ancho de banda de Nyquist (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

Desde $G_R(f) = G^*{}_T(f)$, $G_{RC}(f) = |G_T(f)|^2$ *o* $G_T(f) = \sqrt{G_{RC}(f)}$ y, por lo tanto, debe tener la siguiente respuesta en frecuencia, que se conoce como filtros de coseno elevado de raíz cuadrada:

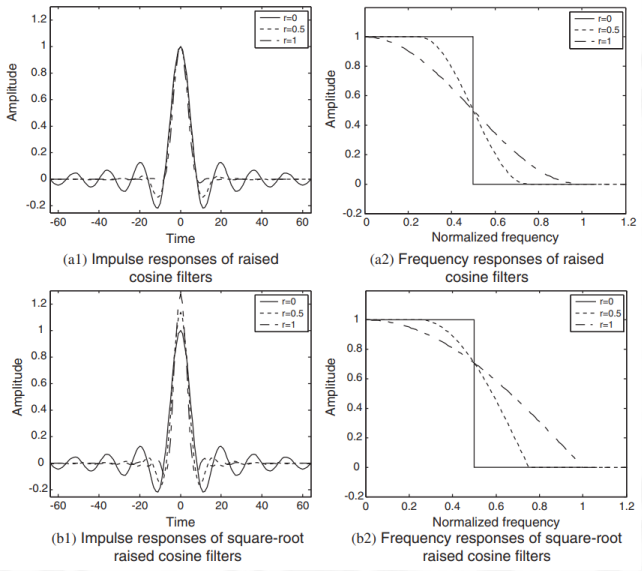

Figura 27. Filtros de coseno elevado y coseno elevado de raíz cuadrada

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Figura], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 114). John Wiley & Sons.

Las Figuras 27 (b1) y (b2) muestran las respuestas de impulso y frecuencia de los filtros de coseno alto de raíz cuadrada con los factores de desintegración de r = 0, 0.5 y 1, respectivamente. Tenga en cuenta que, si se utilizan dos filtros de coseno elevado de raíz cuadrada idénticos como filtros de transmisión y recepción, respectivamente, el último desempeña el papel de un filtro personalizado para el primero y, por lo tanto, la respuesta de frecuencia combinada satisface el criterio de Nyquist, aunque ninguno de ellos lo haga.

3.3.1.1.3. Limitaciones de la Transmisión de una Sola Portadora para Alta Velocidad de Datos.

Hasta ahora, se ha supuesto que el canal está perfectamente compensado por el ecualizador. Sin embargo, a medida que aumenta la tasa de símbolos, el ancho de banda de la señal aumenta. Cuando el ancho de banda de la señal se vuelve mayor que el ancho de banda de coherencia en el canal inalámbrico, el enlace sufre un desvanecimiento de múltiples rutas, incurriendo en la interferencia entre símbolos (ISI). En general, los ecualizadores adaptativos se emplean para tratar el ISI incurrido por el canal de desvanecimiento de múltiples rutas que varía en el tiempo.

Además, la complejidad de un ecualizador aumenta con la tasa de datos. Más específicamente, los ecualizadores adaptativos se implementan mediante filtros de respuesta de impulso finito (FIR) con los coeficientes de derivación adaptativos que se ajustan para minimizar el efecto de ISI. De hecho, se requieren más toques de ecualizador a medida que el ISI se vuelve significativo, por ejemplo, cuando aumenta la velocidad de datos (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

El detector óptimo para el canal de desvanecimiento multitrayecto es un detector de secuencia de máxima verosimilitud (MLSD), que basa sus decisiones en la observación de una secuencia de símbolos recibidos en intervalos de símbolo sucesivos, a favor de maximizar la probabilidad a posteriori. Tenga en cuenta que su complejidad depende del orden de modulación y el número de rutas múltiples. Sea M y L el número de posibles puntos de señal para cada símbolo de modulación y el intervalo de ISI incurrido en el canal de desvanecimiento de trayectos múltiples, respectivamente (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

3.3.1.2. Transmisión de Múltiple Portadora.

3.3.1.2.1. Estructura Base de un Esquema de Transmisión de Múltiple Portadora.

Para superar la selectividad de frecuencia del canal de banda ancha experimentado por la transmisión de una sola portadora, se pueden utilizar múltiples portadoras para la transmisión de datos de alta velocidad. La Figura 28 (a) muestra la estructura básica y el concepto de un sistema de transmisión multiportadora. Aquí, se analiza una señal de banda ancha (a través de múltiples filtros de banda estrecha $H_k(f)'s$) en varias señales de banda estrecha en el transmisor y se sintetiza (a través de múltiples filtros de banda estrecha $G_k\ (f)'s$, cada uno de los cuales se adapta a $H_k\ (f)$) en el receptor de modo que el canal de banda ancha selectiva en frecuencia se pueda aproximar por múltiples frecuencias planas de canales de banda estrecha como se muestra en la Figura 28 (b).

Tenga en cuenta que la no selectividad de frecuencia de los canales de banda estrecha reduce la complejidad del ecualizador para cada subcanal. Mientras se mantenga la ortogonalidad entre los subcanales, la ICI (interferencia entre portadoras) puede suprimirse, lo que conduce a una transmisión sin distorsiones. En un sistema multicanal, la banda ancha se divide en N subcanales de banda estrecha, que tienen la frecuencia de la subportadora desactivada f_k, $k\ =\ 0, 1, 2, \dots, N-1$ (*MIMO-OFDM Wireless Communications with MATLAB*, 2010). La figura 29 (a) muestra la estructura básica de un esquema de comunicación multiportadora, que es una forma específica del sistema multicanal, donde los diferentes símbolos se transmiten con subcanales ortogonales en forma paralela.

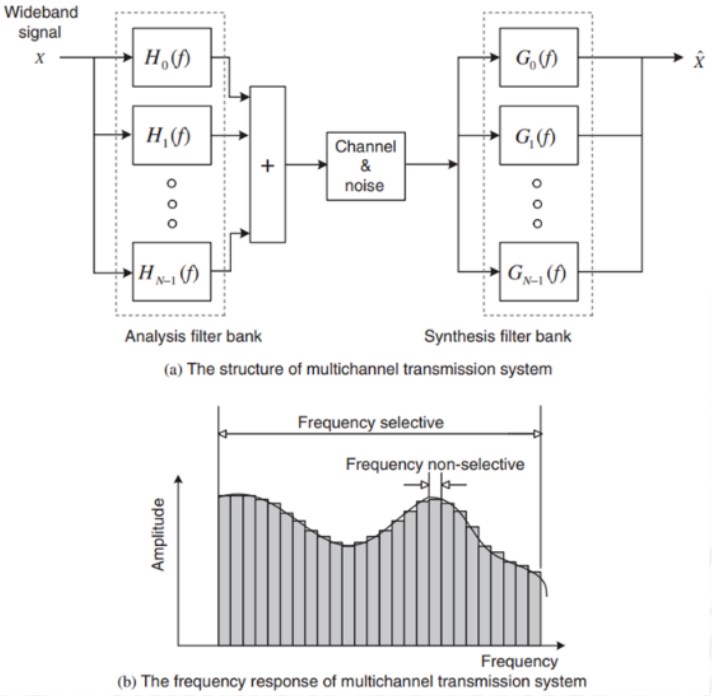

Figura 28. Estructura y características de frecuencia del sistema de transmisión multicanal.

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Figura], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 116). John Wiley & Sons.

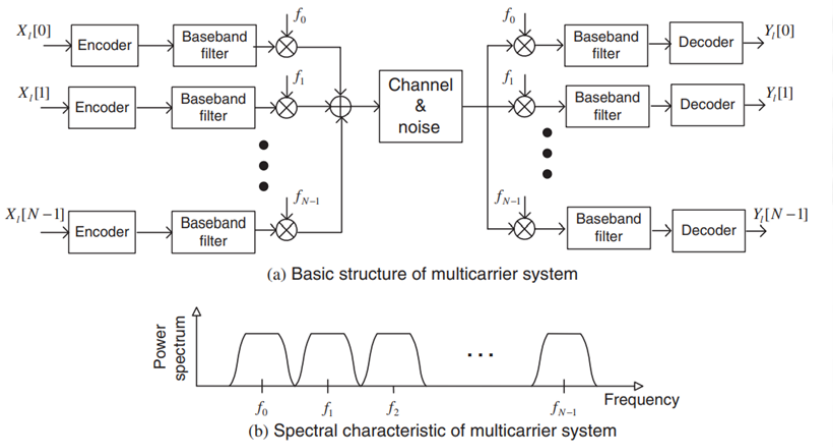

Figura 29. Estructura y característica espectral del sistema de transmisión multiportadora.

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Figura], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 116). John Wiley & Sons.

Sean $X_l[k]$ y $Y_l[k]$ las señales transmitidas y recibidas transportadas a la frecuencia portadora f_k en el l-enésimo intervalo de símbolo, respectivamente.

Implica que la transmisión multiportadora puede considerarse como una especie de método FDMA (acceso múltiple por división de frecuencia).

La Figura 29 (b) ilustra un espectro de señal transmitida en el sistema de transmisión multiportadora, que ocupa múltiples sub-bandas de igual ancho de banda, cada una centrada en la frecuencia portadora diferente. Si cada subcanal tiene banda limitada como se muestra en la Figura 29 (b), se convierte en una transmisión FMT (Filtrado Multi-Tono).

Si bien un sistema de transmisión multiportadora de tipo FMT puede hacer frente a la selectividad de frecuencia de un canal de banda ancha, su implementación se vuelve compleja ya que involucra más codificadores / decodificadores y osciladores, y filtros de mayor calidad a medida que aumenta el número de subportadoras (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

3.3.1.2.2. Esquema de Transmisión OFDM.

El esquema de transmisión de multiplexación por división de frecuencia ortogonal (OFDM) es otro tipo de sistema multicanal, que es similar al esquema de transmisión FMT en el sentido de que emplea múltiples subportadoras. Como se muestra en la Figura 30 (a), no utiliza filtros y osciladores de banda limitada individuales para cada subcanal y, además, los espectros de las subportadoras se superponen para la eficiencia del ancho de banda, a diferencia del esquema FMT donde la banda ancha está completamente dividida en N subcanales ortogonales de banda estrecha (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

Las múltiples señales de subportadoras ortogonales, que se superponen en el espectro, pueden producirse generalizando el criterio de Nyquist de una sola portadora en la Ecuación (6) en el criterio de múltiples portadoras.

En la práctica, los procesos de transformada discreta de Fourier (DFT) y DFT inversa (IDFT) son útiles para implementar estas señales ortogonales.

Tenga en cuenta que DFT e IDFT se pueden implementar de manera eficiente utilizando la transformada rápida de Fourier (FFT) y la transformada rápida inversa de Fourier (IFFT), respectivamente. En el sistema de transmisión OFDM, se toma IFFT de N puntos para los símbolos transmitidos (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

Como todas las subportadoras son de duración finita T, el espectro de la señal OFDM se puede considerar como la suma de las funciones sinc con desplazamiento de frecuencia en el dominio de la frecuencia, como se ilustra en la Figura 30 (c), donde las funciones sinc vecinas superpuestas están espaciadas por 1 / T.

El esquema de multitono discreto (DMT) utilizado en ADSL (Línea de abonado digital asimétrica) y VDSL (Línea de abonado digital de datos de muy alta velocidad) basado en cremallera también tiene la misma estructura que OFDM.

Dado que cada señal de subportadora tiene un límite de tiempo para cada símbolo (es decir, no está limitado por banda), una señal OFDM puede incurrir en radiación fuera de banda, lo que provoca una interferencia de canal adyacente (ACI) no despreciable. Se ve claramente en la Figura 30 (d) que el primer lóbulo lateral no es tan pequeño en comparación con el lóbulo principal en los espectros. Por lo tanto, el esquema OFDM coloca una banda de guarda en las subportadoras externas, llamadas portadoras virtuales (VC), alrededor de la banda de frecuencia para reducir la radiación fuera de banda.

El esquema OFDM también inserta un intervalo de guarda en el dominio del tiempo, llamado prefijo cíclico (CP), que mitiga la interferencia entre símbolos (ISI) y entre símbolos OFDM (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

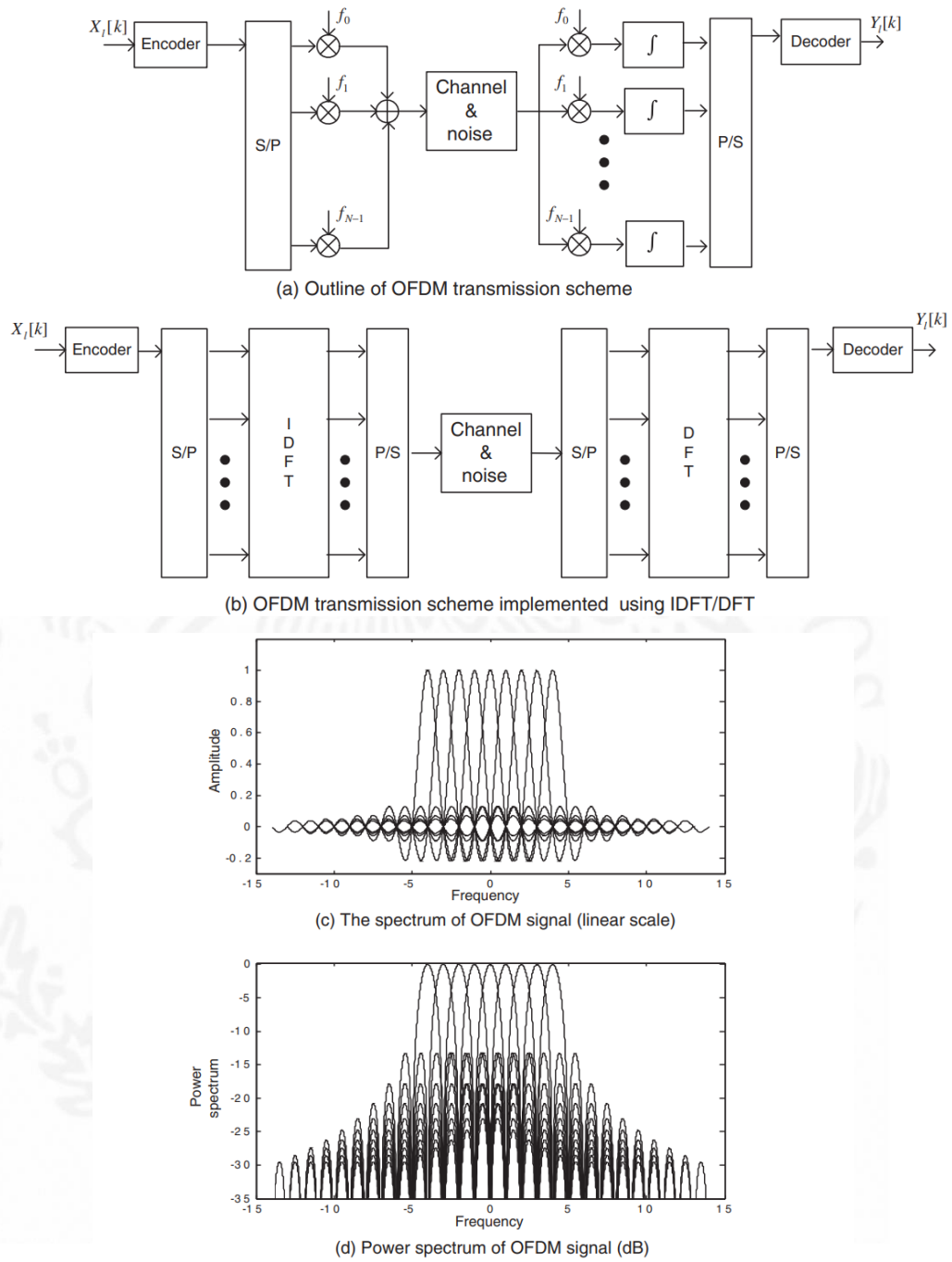

Figura 30. Estructura y característica espectral del esquema de transmisión OFDM.

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Figura], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 118). John Wiley & Sons.

3.3.1.3. Transmisión de Portadora Única Contra Múltiples Portadoras.

Está claro que cada uno de estos esquemas tiene sus propias ventajas y desventajas. El esquema de portadora única puede no ser útil para una transmisión inalámbrica de alta velocidad, simplemente porque requiere un ecualizador de alta complejidad para resolver el problema de la interferencia entre símbolos en el canal de desvanecimiento de trayectos múltiples o, de manera equivalente, en el canal de desvanecimiento selectivo en frecuencia.

Mientras tanto, el esquema de múltiples portadoras es útil para una transmisión inalámbrica de alta velocidad, que no implica la complejidad de la ecualización de canales. Estos esquemas se diferencian entre sí en la forma de dividir la banda de frecuencia en sub-bandas. OFDM no necesita filtros para separar las sub-bandas ya que la ortogonalidad se mantiene entre las subportadoras, pero requiere una banda de protección como VC (Virtual Carriers) para combatir el ACI. En contraste, FMT usa filtros para separar las sub-bandas para reducir el ACI sacrificando la eficiencia espectral, pero no necesita la banda de guarda (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

Desde el punto de vista de la eficiencia espectral, FMT es conocido por ser ventajoso sobre OFDM solo en un caso donde el número de subportadoras es menor de 64. FMT se ha clasificado como una serie de modulación avanzada / adaptativa escalable (SAM), que puede cambiar el número de subportadoras y, en consecuencia, la velocidad de datos. Ha sido adoptado como esquema de transmisión en el estándar TETRA (TErrestrial Trunked RAdio) II en el Instituto Europeo de Normas de Telecomunicaciones (ETSI). Además de OFDM y FMT, existen diferentes tipos de esquemas de transmisión multiportadora, incluyendo DWMT (Discrete Wavelet Multi-Tone), OFDM / OQAM-IOTA, etc (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

En la tabla 4 se resumen las diferencias entre los esquemas de transmisión de una sola portadora y de múltiples portadoras, incluidas sus ventajas y desventajas.

Tabla 4. Diferencias entre los esquemas de transmisión de una y de múltiples portadoras.

	Transmisión de portadora Única	Transmisión de múltiples portadoras	
		OFDM/DMT	FMT
Espaciado de subportadoras		1/ (duración del símbolo)	≥ 1/ (duración del símbolo)
Dar forma al pulso	Filtro de Nyquist (filtro de coseno elevado)	Ventana (rectangular)	Filtro de Nyquist (filtro de coseno elevado)
Separación de subcanales		Ortogonalidad	Filtro pasa bandas
Intervalo de protección	No requerido	Requiere (CP)	No requerido
Banda de guarda	No requerido	Requiere (VC)	No requerido
Ventajas	Sencillo en canales de desvanecimiento planos	Alta eficiencia de ancho de banda para una gran cantidad de subportadoras (≥ 64)	Pequeño ACI
Desventajas	Se requiere un ecualizador de alta complejidad para canales de frecuencia selectiva	Baja eficiencia de ancho de banda y gran ACI para una pequeña cantidad de subportadoras	Alta eficiencia de ancho de banda para una pequeña cantidad de subportadoras (≤ 64)

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Tabla], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 120). John Wiley & Sons.

3.3.2. Modulación y Demodulación OFDM

3.3.2.1. Ortogonalidad.

Considere las señales exponenciales complejas limitadas en el tiempo $\{e^{j2\pi f_k t}\}_{k=0}^{N-1}$ que representan las diferentes subportadoras en $f_k = k/T_{sym}$ en la señal OFDM, donde $0 \leq t \leq T_{sym}$. Estas señales se definen como ortogonales si la integral de los productos para su período común (fundamental) es cero, es decir:

$$\frac{1}{T_{sym}}\int_0^{T_{sym}} e^{j2\pi f_k t} e^{-j2\pi f_i t}\, dt = \frac{1}{T_{sym}}\int_0^{T_{sym}} e^{j2\pi \frac{k}{T_{sym}} t} e^{-j2\pi \frac{i}{T_{sym}} t}\, dt$$

$$= \frac{1}{T_{sym}}\int_0^{T_{sym}} e^{j2\pi \frac{k-i}{T_{sym}} t} = \begin{cases} 1, & \forall\ entero\ k = i \\ 0, & de\ lo\ contrario \end{cases} \qquad (10)$$

Tomando las muestras discretas con las instancias de muestreo en $t = nT_s = \frac{nT_{sym}}{N}, n = 0,1,2,\ldots,N-1$, se puede escribir en el dominio de tiempo discreto como:

$$\frac{1}{N}\sum_{n=0}^{N-1} e^{j2\pi \frac{k}{T_{sym}} nT_s} e^{-j2\pi \frac{i}{T} nT_s} = \frac{1}{N}\sum_{n=0}^{N-1} e^{j2\pi \frac{k}{T_{sym}}\left(\frac{nT}{N}\right)} e^{-j2\pi \frac{i}{T_{sym}}\left(\frac{nT_{sym}}{N}\right)}$$

$$\frac{1}{N}\sum_{n=0}^{N-1} e^{j2\pi \frac{k-i}{N}(n)} = \begin{cases} 1, & \forall\ entero\ k = i \\ 0, & de\ lo\ contrario \end{cases} \qquad (11)$$

La ortogonalidad anterior es una condición esencial para que la señal OFDM esté libre de ICI (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

3.3.2.2. Modulación y Demodulación.

El transmisor OFDM mapea los bits del mensaje en una secuencia de símbolos PSK o QAM que luego se convertirán en N flujos paralelos. Cada uno de los N símbolos de la conversión en serie a paralelo (S / P) se lleva a cabo por una subportadora diferente. Sea X1[k] el símbolo de transmisión kth en la subportadora, $l = 0,1,2,\dots,\infty, k = 0,1,2,\dots,N-1$. Debido a la conversión S / P, la duración del tiempo de transmisión para N símbolos se extiende a NTs, que forma un solo símbolo OFDM con una longitud de T_{sym} $(i.e.., T_{sym} = NT_s)$ (*MIMO-OFDM Wireless Communications with MATLAB*, 2010). Sea $\Psi_{l.k}$ la señal de OFDM l-enésima en la subportadora k-enésima que se da como:

$$\Psi_{l.k} = \begin{cases} e^{j2\pi f_k(t-lT_{sym})}, & o < t \leq T_{sym} \\ 0, & en\ otra\ parte \end{cases} \qquad (13)$$

Entonces, las señales OFDM de banda de paso y de banda base en el dominio de tiempo continuo se pueden expresar respectivamente como:

$$x_l(t) = Re\left\{\frac{1}{T_{sym}}\sum_{l=0}^{\infty}\left\{\sum_{k=0}^{N-1} X_l[k]\Psi_{l.k}(t)\right\}\right\} \qquad (14)$$

Y

$$x_l(t) = \sum_{l=0}^{\infty}\sum_{k=0}^{N-1} X_l[k]e^{j2\pi f_k(t-lT_{sym})}$$

La señal OFDM de banda base de tiempo continuo en la ecuación (14) se puede muestrear en $t = lT_{sym} + nT_s \quad con \quad T_s = \frac{T_{sym}}{N} \quad y \quad f_k = \frac{k}{T_{sym}}$ para producir el símbolo OFDM de tiempo discreto correspondiente como:

$$x_1[n] = \sum_{k=0}^{N-1} X_l[k]e^{\frac{j2\pi kn}{N}} \quad para\ n = 0,1,\dots,N-1 \quad (15)$$

Tenga en cuenta que la ecuación (15) resulta ser la IDFT de N puntos de los símbolos de datos PSK o QAM $\{X_l[k]\}_{k=0}^{N-1}$ y se puede calcular de manera eficiente mediante el algoritmo IFFT (Transformada Rápida de Fourier Inversa).

De acuerdo con la discusión anterior, se puede ilustrar la modulación y demodulación OFDM por el diagrama de bloques de la Figura 31, que muestra que el símbolo de dominio de frecuencia X [k] modula la subportadora con una frecuencia de $f_k = k / T_{sym},\ para\ N = 6\ (es\ decir, k = 0; 1; 2; \ldots; 5)$, mientras que se puede demodular utilizando la ortogonalidad entre las subportadoras en el receptor. Tenga en cuenta que el símbolo original $X[k]$ tiene una duración de T_s, pero su longitud se ha ampliado a $T_{sym} = NT_s$ mediante la transmisión de N símbolos en forma paralela. El símbolo OFDM corresponde a una señal compuesta de N símbolos en forma paralela, que ahora tiene una duración de T_{sym}. Mientras tanto, la Figura 28 (b) ilustra una realización típica de ortogonalidad entre todas las subportadoras. Además, se ha demostrado que esta modulación multiportadora puede implementarse mediante IFFT y FFT en el transmisor y el receptor, respectivamente (*MIMO-OFDM WIRELESS COMMUNICATIONS WITH MATLAB*, 2010).

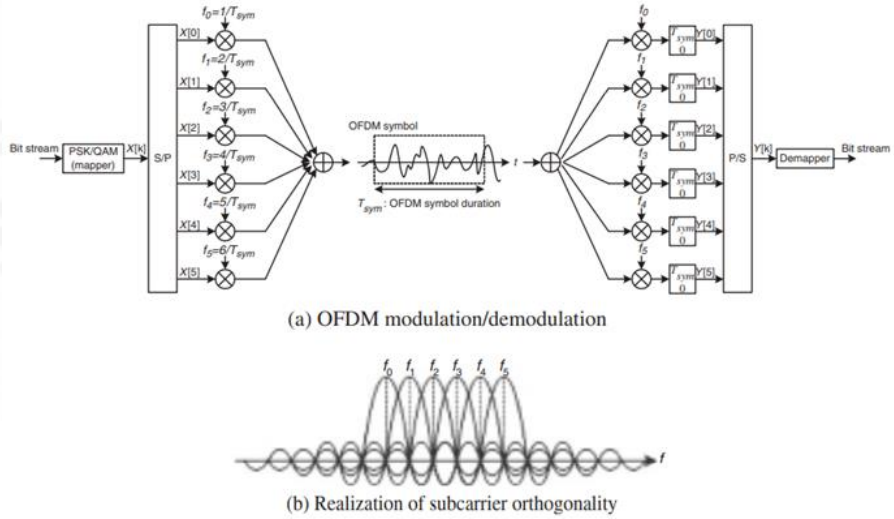

Figura 31. Diagrama de bloques ilustrativo de modulación y demodulación OFDM: N=6.

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Figura], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 125). John Wiley & Sons.

3.3.2.3. Intervalos de Protección OFDM.

3.3.2.3.1. Prefijo Cíclico (CP).

El intervalo de guarda OFDM se puede insertar de dos formas diferentes. Uno es el relleno de ceros (ZP) que rellena el intervalo de guarda con ceros. La otra es la extensión cíclica del símbolo OFDM (para cierta continuidad) con CP (prefijo cíclico) o CS (sufijo cíclico). CP ampliará el símbolo OFDM copiando las últimas muestras del símbolo OFDM en su anverso. *Sea T_G la longitud de CP en términos de muestras*. Entonces, los símbolos OFDM extendidos ahora tienen la duración de $T_{sym} = T_{sub} + T_G$. La figura 32 (a) muestra dos símbolos OFDM consecutivos, cada uno de los cuales tiene el CP de longitud T_G, mientras se ilustra el símbolo OFDM de longitud $T_{sym} = T_{sub} + T_G$ (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

Mientras tanto, la Figura 32 (b) los ilustra conjuntamente en los dominios de tiempo y frecuencia. La Figura 32 (c) muestra los efectos ISI de un canal multitrayecto en algunas subportadoras del símbolo OFDM. Se puede ver en esta figura que *si la longitud del intervalo de guarda (CP) se establece más largo o igual que el retardo máximo de un canal multitrayecto, el efecto ISI de un símbolo OFDM (trazado en una línea de puntos) en el símbolo siguiente está confinado en el intervalo de guarda para que no afecte a la FFT del siguiente símbolo OFDM*, tomado durante la duración de T_{sub}. Esto implica que el intervalo de guarda mayor que el retardo máximo del canal multitrayecto permite mantener la ortogonalidad entre las subportadoras. Como la continuidad de cada subportadora retrasada ha sido garantizada por el CP, su ortogonalidad con todas las demás subportadoras se mantiene sobre T_{sub}, de modo que:

$$\frac{1}{T_{sub}}\int_0^{T_{sub}} e^{j2\pi f_k(t-t_0)}e^{-j2\pi f_i(t-t_0)}dt = 0, \quad k \neq i$$

para la primera señal OFDM que llega con un retraso de t_0, y

$$\frac{1}{T_{sub}}\int_0^{T_{sub}} e^{j2\pi f_k(t-t_0)}e^{-j2\pi f_i(t-t_0-T_s)}dt = 0, \quad k \neq i$$

Para la segunda señal OFDM que llega con un retraso de $t_0 + T_s$ (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

La figura 33 muestra que si la longitud del intervalo de guarda (CP) se establece más corto que el retardo máximo de un canal multitrayecto, la parte de cola de un símbolo OFDM (denotado por un cuarto de círculo) afecta la parte de cabeza del siguiente símbolo, lo que resulta en el ISI. En la práctica, puede ocurrir un desplazamiento de tiempo de símbolo (STO), lo que evita que el encabezado de un símbolo OFDM coincida con el punto de inicio de la ventana FFT. En este contexto, la Figura 33 muestra que incluso si la longitud de CP se establece más larga que el retardo máximo del canal multitrayecto, ISI y / o ICI pueden ocurrir dependiendo de la temporización del punto de inicio de la ventana FFT. Más específicamente, si el punto de inicio de la ventana FFT es anterior que el final retrasado del símbolo anterior se produce ISI; si es posterior al comienzo de un símbolo, no solo aparece ISI (causado por el siguiente símbolo), sino también ICI.

Ahora suponemos que la longitud de CP se establece no más corta que el retardo máximo del canal y el punto de inicio de la ventana FFT de un símbolo OFDM se determina dentro de su intervalo de CP (es decir, no se ve afectado por el símbolo anterior). Entonces, el receptor OFDM toma la FFT de las muestras recibidas $\{yl[n]\}_{n=0}^{N-1}$ para producir donde $X_l[k], Y_l[k], H_l[k], y\ Z_l[k]$ denotan los k -ésimos componentes de frecuencia de la subportadora del l-ésimo símbolo transmitido, el símbolo recibido, la respuesta de frecuencia del canal y el ruido en el dominio de la frecuencia, respectivamente (*MIMO-OFDM Wireless Communications with MATLAB*, 2010).

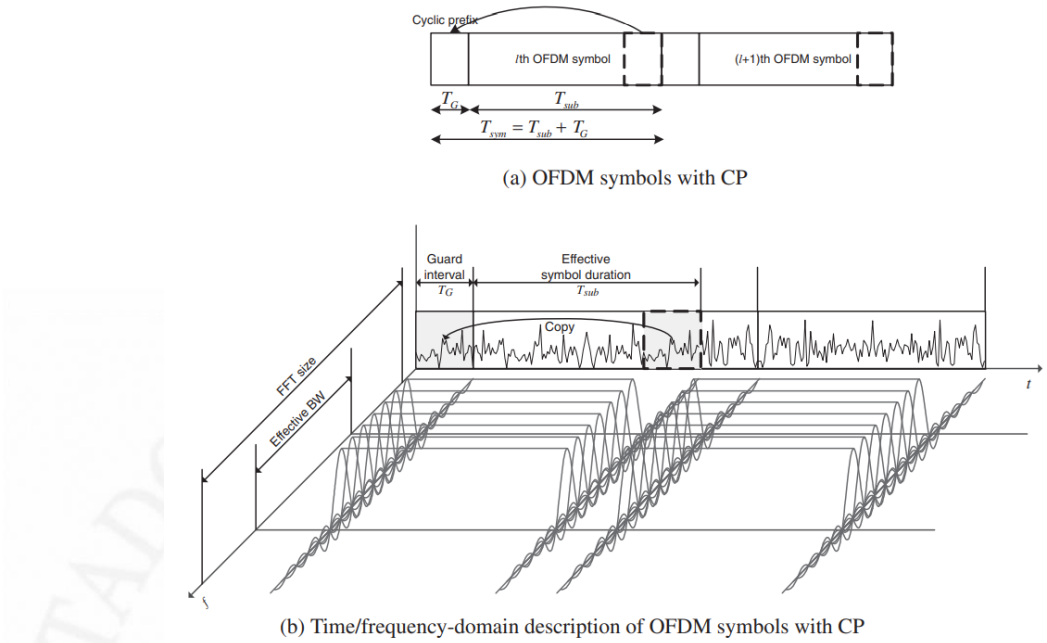

Figura 32. Símbolos OFDM con CP.

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Figura], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 129). John Wiley & Sons.

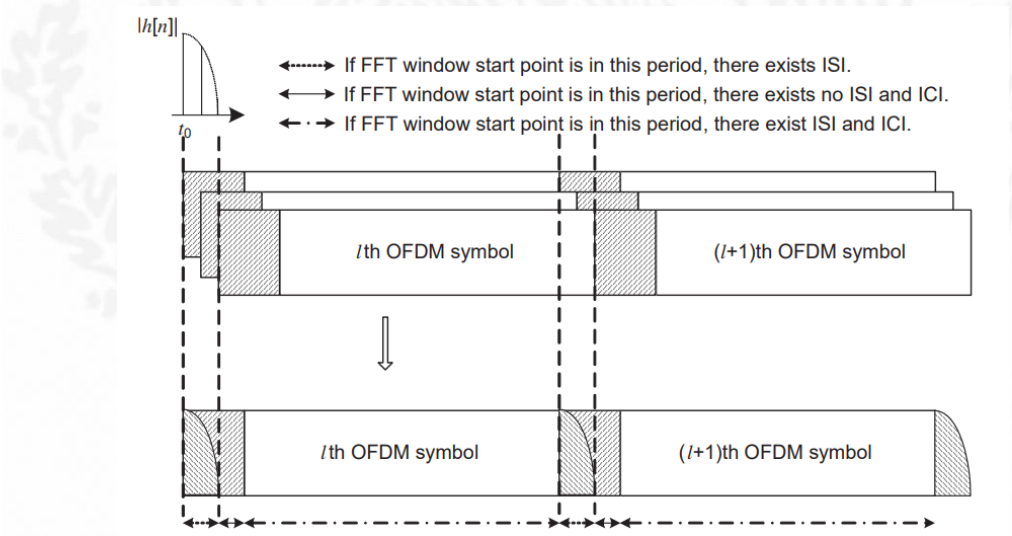

Figura 33. Efecto ISI/ICI en función del punto de la venta FFT.

Nota. Adaptado de *MIMO-OFDM Wireless Communications with MATLAB* [Figura], por Soo, Y., Young, W., & Kim, J., (2010), (2ª ed., pp. 129). John Wiley & Sons.

4. Procedimiento

4.1. Procedimiento y Descripción del Diseño de un Multiplexor OFDM

4.1.1. Transmisor OFDM

4.1.1.1. Bloques Utilizados.

A continuación, se muestran los bloques utilizados y su función.

Tabla 5. Bloques utilizados en el Transmisor OFDM.

Bloque	**Función**
Wav File Source	Secuencia de lectura desde un archivo PCM (.wav) de Microsoft, la salida es flotante.
Float To Uchar	Convierta la secuencia de flotadores en una secuencia de caracteres sin firmar.
Stream to Tagged Stream	Convierte una secuencia normal en una secuencia etiquetada. Todo lo que hace este bloque es agregar etiquetas de longitud en intervalos regulares
Stream CRC32	Flujo de bytes, que forman un paquete. El primer byte del paquete tiene una etiqueta con la clave "longitud" y el valor es el número de bytes en el paquete.
Packet Header Generator	Genera un encabezado para un paquete etiquetado y transmitido. Entrada: una secuencia etiquetada. Salida: Una secuencia etiquetada que contiene el encabezado. Los detalles del encabezado se establecen en un objeto de formateador de encabezado (de tipo packet_header_default o una subclase del mismo).
Repack Bits	Vuelva a empaquetar bits del flujo de entrada en bits del flujo de salida.
Virtual Sink	Cuando se combina con un bloque de fuente virtual, esto es esencialmente lo mismo que dibujar un cable.

Virtual Source	Cuando se combina con un bloque de receptor virtual, esto es esencialmente lo mismo que dibujar un cable entre dos bloques.
Chunks to Symbols	Asignar una secuencia de índices de símbolos desempaquetados a una secuencia de puntos flotantes o de constelaciones complejas en dimensiones D.
Tagged Stream Mux	Combina secuencias etiquetadas.
OFDM Carrier Allocator	Este bloque convierte un flujo de símbolos complejos de modulación escalar en vectores que son la entrada para un IFFT en un transmisor OFDM.
FFT	Este bloque toma un vector de flotadores o valores complejos y calcula el FFT.
OFDM Cyclic Prefixer	Agrega un prefijo cíclico y realiza la forma de pulsos en los símbolos OFDM.
Multiply Const	Multiplica el flujo de entrada por una constante escalar o vectorial (vectorial si es vectorial).
Tag Gate	Controlar la propagación de etiquetas.
Variable	Este bloque asigna un valor a una variable única. Puede usar la variable en el campo de parámetros de otro bloque simplemente usando el ID del bloque de variables.
QT GUI RANGE	Este bloque crea una variable con una selección de widgets.
QT GUI Time Sink	Este es un sumidero gráfico basado en QT que toma el conjunto de corrientes flotantes y las traza en el dominio del tiempo.
QT GUI Frequency Sink	Este es un sumidero gráfico basado en QT que toma el conjunto de flujos de coma flotante y traza el PSD.
UHD: USRP Sink	Bloque utilizado para transmitir muestras a un dispositivo USRP (es decir, actuar como transmisor).

Nota. Elaboración Propia.

4.1.1.2. Variables.

- samp_rate: Tasa de muestreo.
- sft_len: La longitud de FFT (entero).
- packet_len: La longitud del paquete.
- cp_len: Longitud del prefijo cíclico en muestras totales (entero).
- header_equalizer: Ecualizador de encabezado.
- header_formatter: Formato de encabezado.
- occupied_carriers: Vector de vectores que describe qué portadores de OFDM están ocupados.
- pilot_carriers: Un vector de vectores que describe qué portadores OFDM están ocupados con símbolos piloto.
- rolloff: El factor de roll-off indica el porcentaje de ancho de banda que excede la señal de la familia del coseno alzado respecto ancho de banda que ocuparía el pulso rectangular cuya respuesta impulsional presentara los mismos pasos por cero.
- pilot_symbols: Simbolos piloto
- packet_length_tag_key: Nombre de la etiqueta que da la longitud del paquete en la entrada.
- length_tag_key: Clave de la etiqueta de longitud de fotograma.
- sync_word 1: El primer símbolo del preámbulo de sincronización. Esto tiene que ser con ceros en portadores alternos.
- sync_word 2: El segundo símbolo del preámbulo de sincronización. Esto tiene que ser llenado por completo.
- header_mod: Modulación de cabecera.
- payload_mod: Modulación de carga útil.
- payload_equalizer: Ecualizador de carga útil.

4.1.1.3. Conexión de los Bloques.

En primer lugar, en el entorno de diseño GNU Radio se realiza la conexión de los bloques pertenecientes al transmisor OFDM, el cual se divide en dos estructuras importantes: el Mapeo de los datos en paquetes para su transmisión y la Implementación de la OFDM y su transmisión.

4.1.1.3.1. Mapeo de los Datos en Paquetes para su Transmisión.

Con respecto al Mapeo de los datos en paquetes para su transmisión, se puede elegir entre una gran variedad de formatos, pero en este caso se ha optado por una señal de audio en formato wav. La señal de audio se pasa a través de un bloque Float to Uchar convirtiendo la secuencia de floats en una secuencia de caracteres sin firmar, que posteriormente al conectarse con el bloque Stream to Tagged Strean se convierte en una secuencia etiquetada de longitud en intervalos regulares, con un número de elementos por paquete de flujo etiquetado (Packet length) de 96 bits y una clave de etiqueta de longitud (Length Tag Key) “packet_len” ("Stream to Tagged Stream - GNU Radio", 2019).

El formato de datos que se está manejando hasta el momento son byte representado en bloques con un color magenta en las entradas y salidas respectivamente. Ahora bien, mediante el bloque Stream CRC32 se crea el CRC del flujo de bytes que forman el paquete; para esto se debe indicar en mode “Generate CRC”, en Length tag name “packet_len” y en Packed “yes”.

A continuación, de la salida del bloque Stream CRC32 se realiza la conexión a un bloque Packet Header Generator y a un bloque Repack Bits. Los bloques se configuran de la siguiente forma:

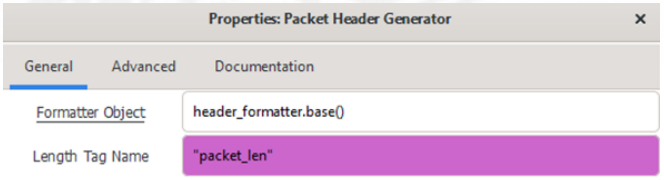

Figura 34. Configuración del bloque Packet Header Generator.

Nota. Elaboración Propia en el Software GNU Radio.

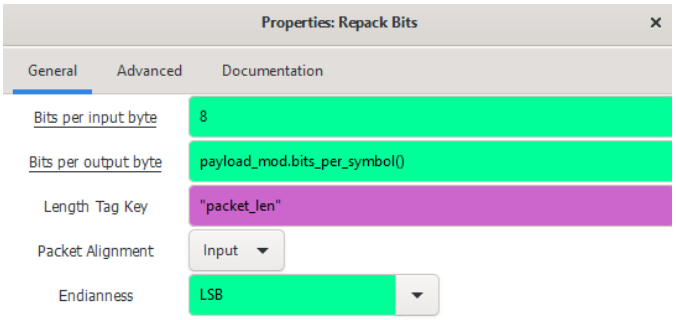

Figura 35. Configuración del bloque Repack Bits.

Nota. Elaboración Propia en el Software GNU Radio.

Una vez configurado los bloques anteriores, la salida de cada uno de los bloques anteriores se conecta a un bloque Virtual Sink con un Stream ID: “Header Bits” en el caso del bloque Packet Header Generator y el Stream ID: “Payload Bits” para el bloque Repack Bits.

Se colocan 2 bloques Virtual Source con los siguientes Stream ID; “Header Bits” para el primer bloque y “Payload Bits” para el segundo.

A continuación, el Virtual Source con el Stream ID: “Header Bits”, se conecta con un bloque Chunks to Symbols, con la finalidad de asignar una secuencia de índices de símbolos a una secuencia de constelaciones complejas en una dimensión especifica ("Chunks to Symbols - GNU Radio", 2019). Ahora bien, para el Virtual Source con el Stream ID: “Payload Bits” se debe seguir el mismo procedimiento de conexión, pero la configuración es diferente, tal y como se observa en la Figura 36 y 37.

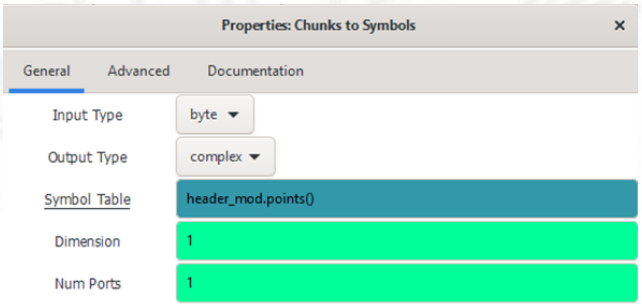

Figura 36. Configuración del bloque Chunks to Symbol para el virtual Source “Header Bits”.

Nota. Elaboración Propia en el Software GNU Radio.

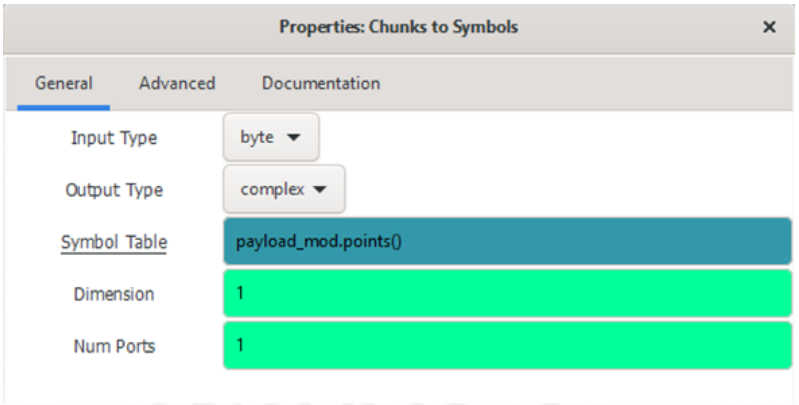

Figura 37. Configuración del bloque Chunks to Symbol para el Virtual Source "Payload Bits"

Nota. Elaboración Propia en el Software GNU Radio.

Después de configurar los bloques Chunks to Symbol, se conectan sus salidas a un bloque Tagged Stream Mux, que recibe la secuencia de constelaciones complejas con una longitud de paquete determinada, las cuales son enviadas secuencialmente desde cada flujo de entrada ("Tagged Stream Mux - GNU Radio", 2019). Dentro del bloque se suman todas las etiquetas de longitud individuales, proporcionando una señal de salida que tiene una nueva etiqueta de longitud que cumplirá con las características necesarias para la asignación de portadora OFDM.

La salida del bloque Tagged Stream Mux se conecta con un bloque Virtual Sink asignado con un Stream ID: "Pre-OFDM".

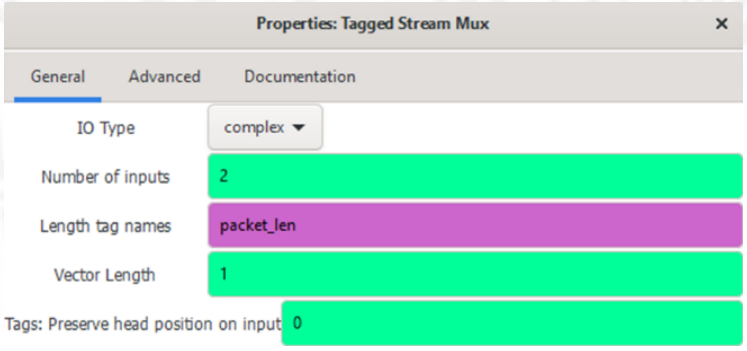

Figura 38. Configuración del bloque Tagged Stream Mux.

Nota. Elaboración Propia en el Software GNU Radio.

4.1.1.3.2. Implementación de la OFDM y Transmisión.

Ahora bien, para la Implementación de la OFDM y Transmisión de la misma, se coloca un bloque Virtual Source y se le asigna el Stream ID: “Pre-OFDM”. Después, se conecta la salida de este bloque con la entrada de un bloque OFDM Carrier Allocator; este bloque permite colocar los pilotos y los símbolos de datos útiles (payload) en determinadas posiciones de la subportadora del vector de entrada IFFT.

Con respecto a la asignación, las variables “occupied carriers” y “pilot carriers” permiten identificar la posición dentro de un Frame donde se almacenan los datos y los símbolos del piloto, respectivamente. Hay que mencionar, además que cada símbolo OFDM lleva una cantidad de símbolos de datos establecida por la variable “occupied carriers” y las portadoras que no se utilizan, es donde se pueden colocar los símbolos del piloto ("OFDM Carrier Allocator - GNU Radio", 2019).

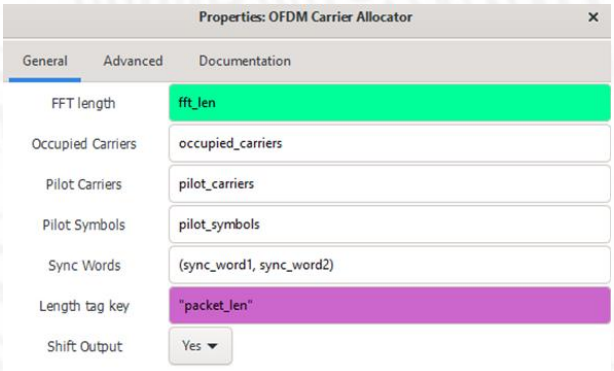

Figura 39. Configuración del bloque OFDM Carrier Allocator.

Nota. Elaboración Propia en el Software GNU Radio.

Con respecto a la conversión al dominio del tiempo, se utiliza un bloque FFT el cual toma la señal de salida del bloque OFDM Carrier Allocator y calcula el IFFT. Es necesario recalcar que incluso si la señal de entrada es real, la salida será compleja, por lo que se debe usar un bloque Complex to Mag si se desea ver la magnitud.

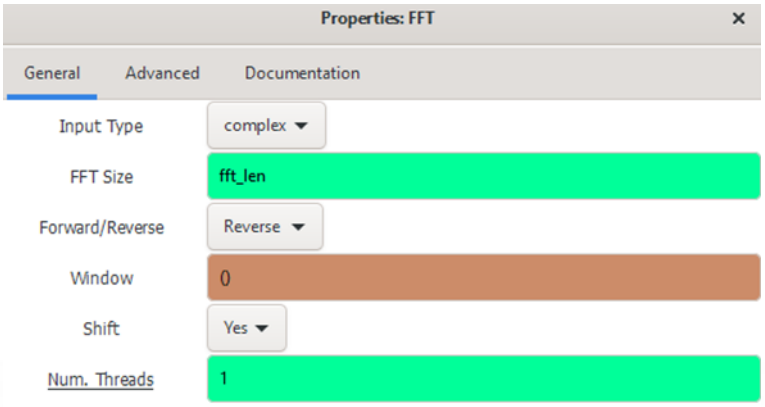

Figura 40. Configuración del bloque FTT.

Nota. Elaboración Propia en el Software GNU Radio.

Después de la conversión al dominio del tiempo, se puede agregar un CP delante de cada símbolo OFDM mediante el uso de un bloque OFDM Cyclic Prefixer conectándolo a la salida del bloque FFT y configurándolo de la siguiente forma:

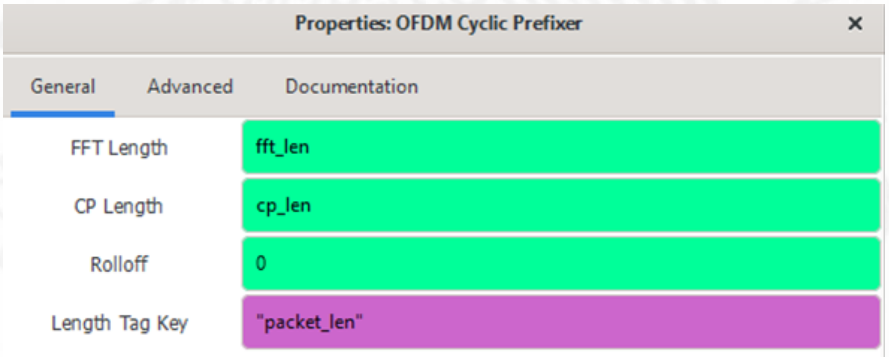

Figura 41. Configuración del bloque OFDM Cyclic Prefixer.

Nota. Elaboración Propia en el Software GNU Radio.

A continuación, se conecta la salida del bloque OFDM Cyclic Prefixer a un bloque Virtual Sink y se le asigna a este último el Stream ID: “Time Domain”.

Se coloca un nuevo bloque Virtual Source y se le asigna como Stream ID: “Time Domain”. A continuación, se conecta con un bloque Multiply Const el cual multiplica el flujo de entrada por una constante escalar para mantener constante la señal ("Multiply Const - GNU Radio", 2018). Después, al utilizar un bloque Tag Gate se evita que las etiquetas (tags) se propaguen durante la transmisión.

Con respecto a la transmisión, se utilizó un Bloque UHD: USRP Sink, ya que es el bloque compatible con la USRP B200 SDR 1X1, la cual es la tarjeta encargada de la transmisión. A fin de que no existan problemas durante la transmisión, a continuación, se detalla el proceso de la configuración de este bloque. Para llevar a cabo la configuración del bloque UHD: USRP Sink, se utilizaron tres bloques QT GUI Range.

El primer bloque QT GUI Range se utilizó para controlar la ganancia rf de la tarjeta USRP B200 SDR 1X1. Para esto, el bloque ahora identificado con el Id: “rf_gain” (Figura 42) necesita tener un valor por defecto (Default Value) de 10, empezando en 0 y terminando en 60 con pasos de 1.

Por su parte, el segundo bloque QT GUI Range ahora identificado con el Id: “frequency” (Figura 43), permite controlar la frecuencia en la cual está transmitiendo la USRP. Su configuración consiste en asignar un valor por defecto de 430 MHz, en un intervalo de frecuencias de 70 MHz (Start) a 6 GHz (Stop) con pasos de 1kHz (Step). Ahora bien, el tercer bloque QT GUI Range identificado con el Id: “bandwidth” (Figura 44), permite controlar el ancho de banda de la USRP. Se le asigno un valor por defecto de 1 MHz, comprendido en un intervalo de frecuencias de 1 MHz a 50 MHz con pasos de 100 Hz. Se debe tener en cuenta que el tamaño de los pasos se puede modificar de acuerdo con las necesidades.

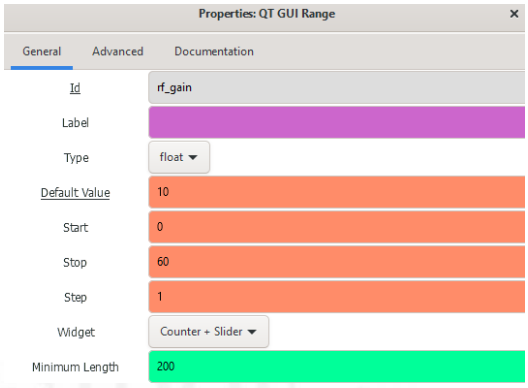

Figura 42. Configuración del bloque QT GUI Range para el Id "rf_gain".

Nota. Elaboración Propia en el Software GNU Radio.

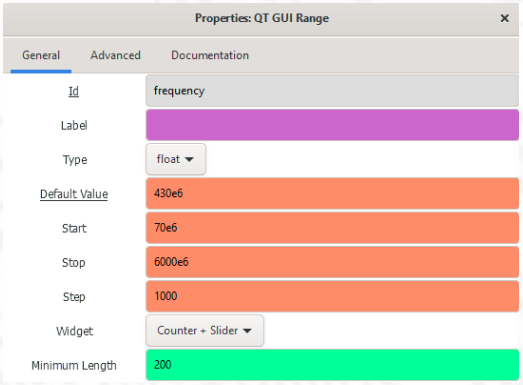

Figura 43. Configuración del bloque QT GUI Range para el Id "frequency"

Nota. Elaboración Propia en el Software GNU Radio.

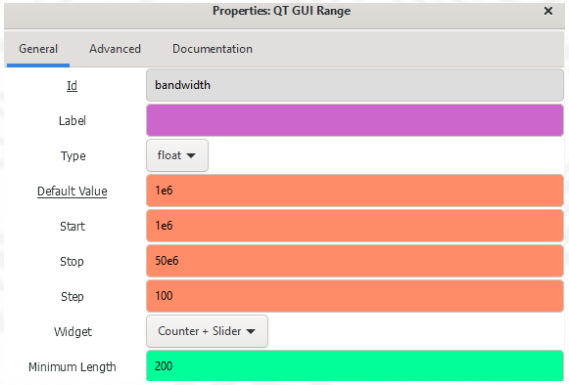

Figura 44. Configuración del bloque QT GUI Range para el Id "bandwidth".

Nota. Elaboración Propia en el Software GNU Radio.

Después de configurar los bloques QT GUI Range, ya es posible realizar la configuración del bloque UHD: USRP Sink.

En primer lugar, se configura el parámetro Device Addres en donde se ingresa el numero serial de la tarjeta, en este caso es “serial=30F9A60”. En segundo lugar, el parámetro Sync se configura con la opción “No Sync” ya que no se utiliza un PLL externo. En tercer lugar, la tasa de muestreo (Samp rate) asignando el valor “samp_rate” ya que de acuerdo con el teorema de muestreo de Nyquist-Shannon, para poder digitalizar una señal analógica y transmitirla por un medio eléctrico a grandes distancias y poder recuperarla en el extremo distante con la máxima fidelidad posible, se requiere que la señal analógica sea muestreada al menos dos veces su frecuencia máxima.

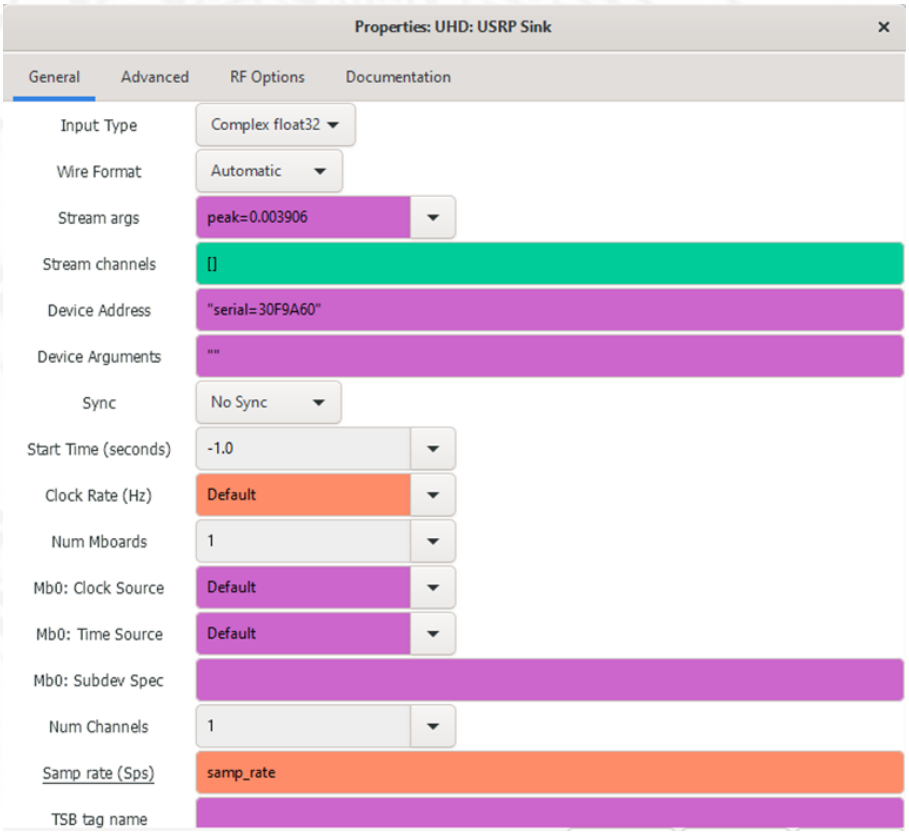

Figura 45. Configuración del bloque UHD: USRP Sink para en su apartado General.

Nota. Elaboración Propia en el Software GNU Radio.

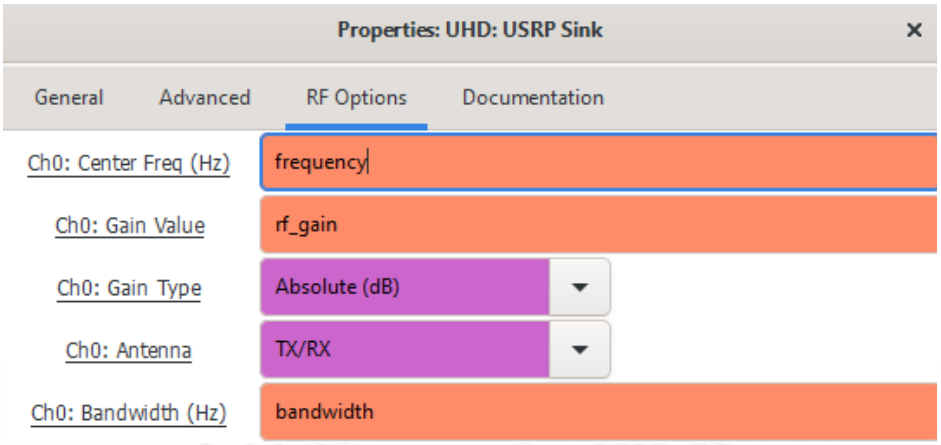

Figura 46. Configuración del bloque UHD: USRP Sink en su apartado RF Options.

Nota. Fuente Propia.

Llegados a este punto, se ha conseguido un Multiplexor OFDM compuesto por el generador de paquetes, el multiplexor de la señal, la asignación de IFFT y CP y el apartado donde se llevó a cabo la configuración de los parámetros para la transmisión.

4.1.2. Receptor OFDM

A continuación, se detalla el proceso del Receptor OFDM, el cual consta de tres estructuras principales que son la Recepción y ecualización a través de frames, la Estimación y Ecualización del Canal, y la Recuperación de datos después de la demodulación.

4.1.2.1. Bloques Utilizados.

Tabla 6. Bloques utilizados en el Receptor OFDM.

Bloque	**Función**
Osmocom Source	Bloque utilizado para recibir muestras a un dispositivo HackRF One (es decir, actuar como receptor).
Schmidl & Cox OFDM synch.	La evaluación de la compensación de frecuencia gruesa se realiza en este bloque. Además, los toques iniciales del ecualizador no se calculan aquí.
Delay	Retrasar la entrada en un cierto número de muestras. Los retrasos positivos insertan cero elementos al principio de la secuencia.
Frequency Mod	Este bloque es un seno complejo controlado por amplitud de entrada. Emite una señal, que tiene un aumento de fase momentáneo que es proporcional a la sensibilidad y la amplitud de entrada.
Packet Header Generator	Genera un encabezado para un paquete etiquetado y transmitido. Entrada: una secuencia etiquetada. Esto se consume por completo, no se agrega al flujo de salida. Salida: Una secuencia etiquetada que contiene el encabezado. Los detalles del encabezado se establecen en un objeto de formateador de encabezado (de tipo packet_header_default o una subclase del mismo).
Multiply	Multiplique en todos los flujos de entrada.

Header/Payload Demux	Este bloque está diseñado para demultiplexar paquetes de una transmisión en ráfaga. La aplicación típica para este bloque es el caso cuando está recibiendo paquetes con una longitud aún por determinar. Este bloque pasará la sección de cabecera a otros bloques para su demodulación. Usando la información del encabezado demodulado, luego generará la carga útil.
Virtual Sink	Cuando se combina con un bloque de fuente virtual, esto es esencialmente lo mismo que dibujar un cable entre dos bloques.
Virtual Source	Cuando se combina con un bloque de receptor virtual, esto es esencialmente lo mismo que dibujar un cable entre dos bloques.
FFT	Este bloque toma un vector de flotadores o valores complejos y calcula el FFT.
OFDM Channel Estimation	Estimación del canal y la desviación de frecuencia gruesa para OFDM a partir de preámbulos.
OFDM Frame Equalizer	Realiza la ecualización en una o dos dimensiones en un fotograma OFDM etiquetado.
OFDM Serializer	Serializa símbolos de modulaciones complejas de subportadoras OFDM.
Constellation Decoder	Decodificar los puntos de una constelación desde un espacio complejo a bits (desempaquetados) basados en el mapa del objeto.
Packet Header Parser	En cierto sentido, este es el bloque inverso al Generador de Encabezado de Paquetes. La diferencia es que el encabezado analizado no se genera como una secuencia, sino como un diccionario PMT, que se publica en el puerto de mensajes con el id "header_data".

Repack Bits	Vuelva a empaquetar bits del flujo de entrada en bits del flujo de salida.
Stream CRC32	Flujo de bytes, que forman un paquete. El primer byte del paquete tiene una etiqueta con la clave "longitud" y el valor es el número de bytes en el paquete.
Tag Debug	Este bloque recopila todas las etiquetas que se le envían en todos los puertos de entrada y las muestra en stdout de forma formateada.
File Sink	Este archivo se puede leer en cualquier entorno de programación que pueda leer archivos binarios.
Variable	Este bloque asigna un valor a una variable única. Puede usar la variable en el campo de parámetros de otro bloque simplemente usando el ID del bloque de variables.
QT GUI RANGE	Este bloque crea una variable con una selección de widgets. El ID será el nombre de la variable, de modo que el ID se puede utilizar como parámetro en otro bloque para hacerlo ajustable en tiempo real.
QT GUI Time Sink	Este es un sumidero gráfico basado en QT que toma el conjunto de corrientes flotantes y las traza en el dominio del tiempo. Cada señal se traza con un color diferente, y las funciones y se pueden usar para cambiar la etiqueta y el color de un número de entrada determinado.
QT GUI Frequency Sink	Este es un sumidero gráfico basado en QT que toma el conjunto de flujos de coma flotante y traza el PSD.
UHD: USRP Sink	Bloque utilizado para transmitir muestras a un dispositivo USRP (es decir, actuar como transmisor).

Nota. Elaboración propia.

4.1.2.2. Variables

- samp_rate: Tasa de muestreo.
- sft_len: La longitud de FFT (entero).
- packet_len: La longitud del paquete.
- cp_len: Longitud del prefijo cíclico en muestras totales (entero).
- header_equalizer: Ecualizador de encabezado.
- header_formatter: Formato de encabezado.
- occupied_carriers: Vector de vectores que describe qué portadores de OFDM están ocupados.
- pilot_carriers: Un vector de vectores que describe qué portadores OFDM están ocupados con símbolos piloto.
- rolloff: El factor de roll-off indica el porcentaje de ancho de banda que excede la señal de la familia del coseno alzado respecto ancho de banda que ocuparía el pulso rectangular cuya respuesta impulsional presentara los mismos pasos por cero.
- pilot_symbols: Simbolos piloto
- packet_length_tag_key: Nombre de la etiqueta que da la longitud del paquete en la entrada.
- length_tag_key: Clave de la etiqueta de longitud de fotograma.
- sync_word 1: El primer símbolo del preámbulo de sincronización. Esto tiene que ser con ceros en portadores alternos.
- sync_word 2: El segundo símbolo del preámbulo de sincronización. Esto tiene que ser llenado por completo.
- header_mod: Modulación de cabecera.
- payload_mod: Modulación de carga útil.
- payload_equalizer: Ecualizador de carga útil.

4.1.2.3. Conexión de los Bloques.

4.1.2.3.1. Recepción y Ecualización a Través de Frames.

En primer lugar, se agrega un bloque Osmocom Source el cual permite recibir la señal a través de nuestra tarjeta SDR HackRF One. Para llevar a cabo la configuración del bloque Osmocom Source, se utilizan tres bloques QT GUI Range.

El primer bloque QT GUI Range se utilizó para controlar la ganancia rf de la tarjeta SDR HackRF One. Para esto, el bloque ahora identificado con el Id: “rf_gain” (Figura 47) necesita tener un valor por defecto (Default Value) de 10, empezando en 0 y terminando en 60 con pasos de 1.

Por su parte, el segundo bloque QT GUI Range ahora identificado con el Id: “frequency” (Figura 48), permite controlar la frecuencia en la cual está transmitiendo la HackRF One. Su configuración consiste en asignar un valor por defecto de 430 MHz, en un intervalo de frecuencias de 25 MHz (Start) a 4 GHz (Stop) con pasos de 1kHz (Step). Ahora bien, el tercer bloque QT GUI Range identificado con el Id: “bandwidth” (Figura 49), permite controlar el ancho de banda de la HackRF One. Se le asigno un valor por defecto de 2 MHz, comprendido en un intervalo de frecuencias de 2 MHz a 20 MHz con pasos de 100 Hz.

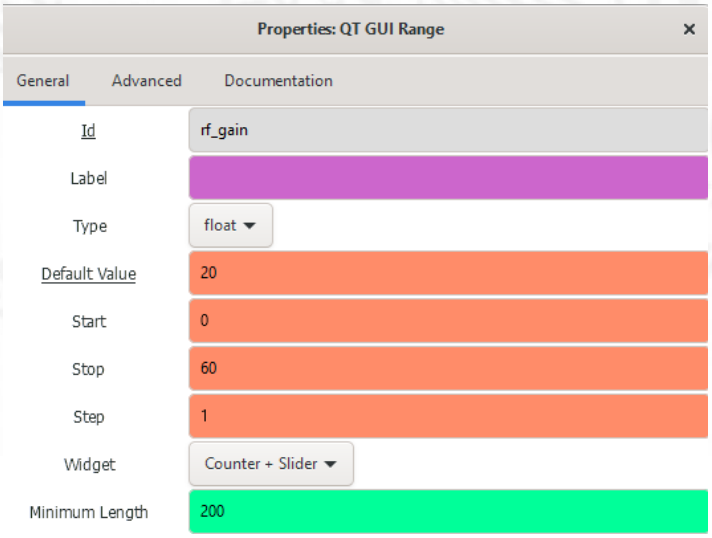

Figura 47. Configuración del bloque QT GUI Range para el Id “rf_gain”.

Nota. Elaboración Propia en el Software GNU Radio.

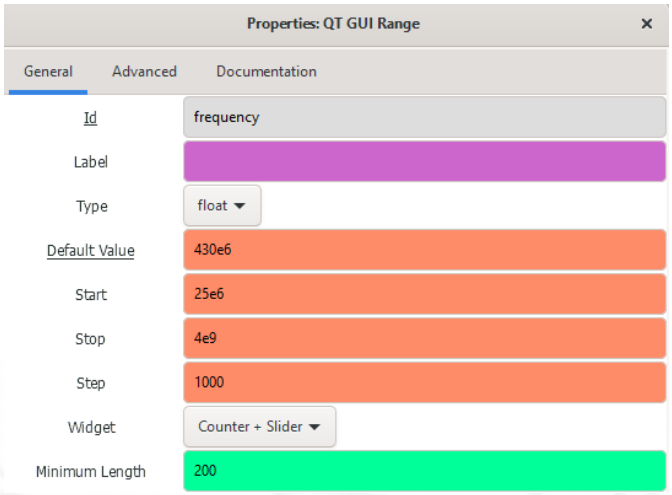

Figura 48. Configuración del bloque QT GUI Range para el Id "frequency".

Nota. Elaboración Propia en el Software GNU Radio.

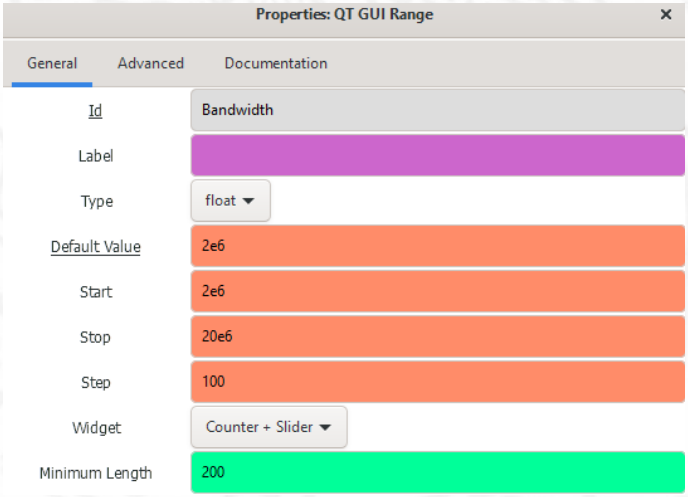

Figura 49. Configuración del bloque QT GUI Range para el Id "Bandwidth".

Nota. Elaboración Propia en el Software GNU Radio.

Después de configurar los bloques QT GUI Range, ya es posible realizar la configuración del bloque Osmocom Source. En algunos casos el equipo en el que se realiza la simulación no soporta la tasa de muestreo, por lo cual puede reducirse hasta 100 kHz para garantizar la recepción de la señal.

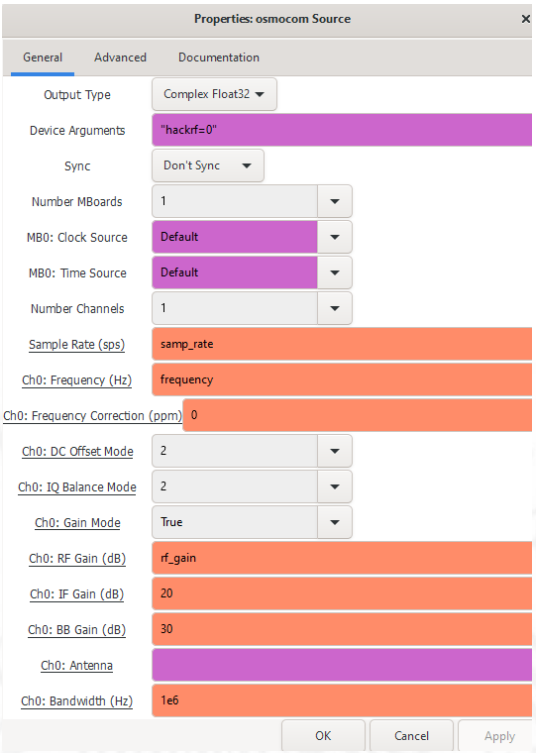

Figura 50. Configuración del bloque Osmocom Source.

Nota. Elaboración Propia en el Software GNU Radio.

Ahora bien, la salida del bloque Osmocom Source se conecta a un bloque Schmidl & Cox OFDM synch y a un bloque Delay.

El bloque Schmidl & Cox OFDM synch recibe del bloque Osmocom Source, las muestras complejas como señal de entrada y realiza la evaluación de la compensación de frecuencia gruesa. La salida de este bloque se conecta con un bloque Frequency Mod, que es un seno completo controlado por amplitud de entrada y que tiene un aumento de fase momentánea que es proporcional a la sensibilidad y la amplitud de entrada ("Schmidl & Cox OFDM synch. - GNU Radio", 2019).

Más específicamente, toma una señal real de banda base (x_m[n]) y emite una señal de frecuencia modulada (y[n]) de acuerdo con:

$$y[n] = e^{j2\pi\frac{f\Delta}{f_s}\sum x[n]}$$

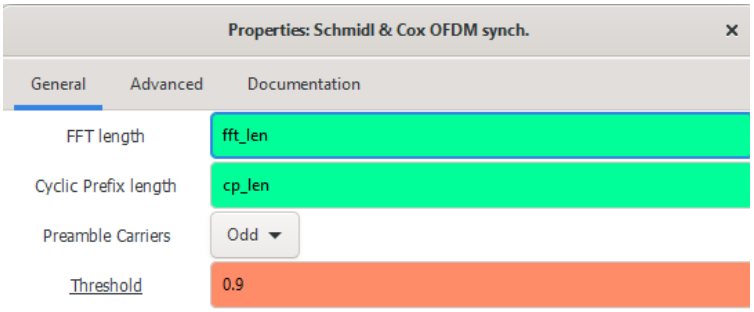

Figura 51. Configuración del bloque Schmidl & Cox OFDM synch.

Nota. Elaboración Propia en el Software GNU Radio.

Por su parte, el bloque Delay retrasa la señal en 80n muestras lo equivalente a la longitud del FFT con prefijo cíclico.

A continuación, las salidas del bloque Delay y Frecuency Mod se conectan a una entrada de un bloque Multiply respectivamente. De modo que la señal de salida es una multiplicación en cada uno de los flujos de entrada.

Ahora bien, la salida del bloque Multiply se conecta a la entrada (In) del bloque Header/Payload Demux. Este bloque está diseñado para demultiplexar paquetes de una transmisión en ráfaga. Al no contar aun con una longitud de paquetes determinada, pasará la sección de cabecera a otros bloques para su demodulación. Después, se utilizará la información del encabezado demodulado para generar la carga útil.

A continuación, se conecta la salida hfr (out_hdr) del bloque Header/Payload Demux a un bloque Virtual Sink, al que se le asigna el Stream ID: “Header Stream”. Por su parte, se conecta la salida de carga útil (out_payload) a otro bloque Virtual Sink al cual se le asigna el Stream ID: “Payload Stream”. Los flujos de carga útil y de encabezado se extraen en el bloque Header/Payload Demux y luego se realiza la estimación de canal para compensar las distorsiones en el mismo.

Dicho lo anterior, se explica a continuación el proceso de la estimación del canal con la finalidad de compensar las distorsiones del canal, para posteriormente pasar a la ecualización a través de frames de la señal recibida.

4.1.2.3.2. Estimación de Canal y Ecualización de Canal.

En primer lugar, se coloca un bloque Virtual Source al cual se le asigna el Stream ID: “Header Stream”. A continuación, la salida de este bloque se conecta a un bloque FFT, que toma los valores de la señal de entrada compleja y calcula el FFT, permitiendo convertir la señal del dominio del tiempo al dominio de la frecuencia con una longitud de FFT de 64.

Con el fin de realizar la estimación del canal OFDM se utiliza un bloque OFDM Channel Estimation en el cual se reciben los símbolos OFDM (en el dominio de la frecuencia). Internamente lo que se espera es que los primeros símbolos sean símbolos de sincronización, que se utilizan para estimar el desplazamiento de frecuencia gruesa y los tonos iniciales del ecualizador para que en la salida se obtengan los símbolos de datos sin los símbolos de sincronización.

El primer símbolo de sincronización se encuentra en el dominio de la frecuencia y cuenta con una longitud igual a la longitud FFT. El segundo símbolo de sincronización también se encuentra en el dominio de la frecuencia y cumple la función de describir la sincronización de Schmidl & Cox.

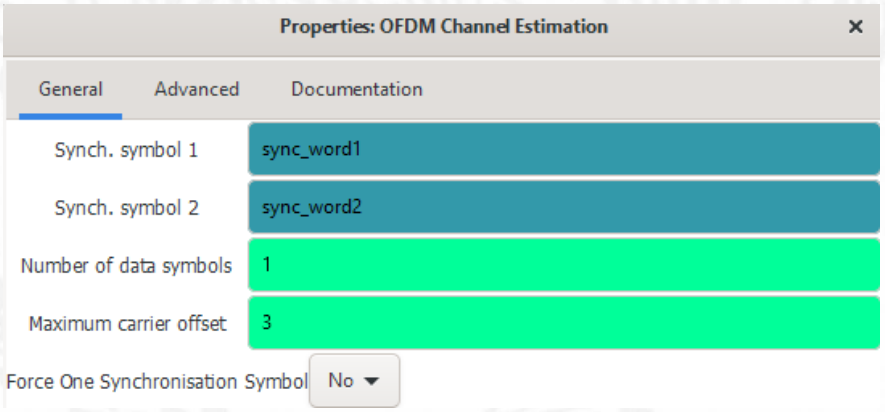

Figura 52. Configuración del bloque OFDM Channel Estimation.

Nota. Elaboración Propia en el Software GNU Radio.

A continuación, mediante el uso de un bloque OFDM Frame Equalizer se elimina el desplazamiento del portador grueso y después se realiza la ecualización en una o dos dimensiones en un Frame OFDM etiquetado. Se debe tener en cuenta que la etiqueta con el desplazamiento del portador grueso no se elimina (ofdm_sync_carr_offset). La señal de entrada es una serie de símbolo OFDM y la salida es una señal igual a la entrada, pero ecualizada y corregida en frecuencia ("OFDM Frame Equalizer - GNU Radio", 2019). La configuración del bloque se muestra a continuación:

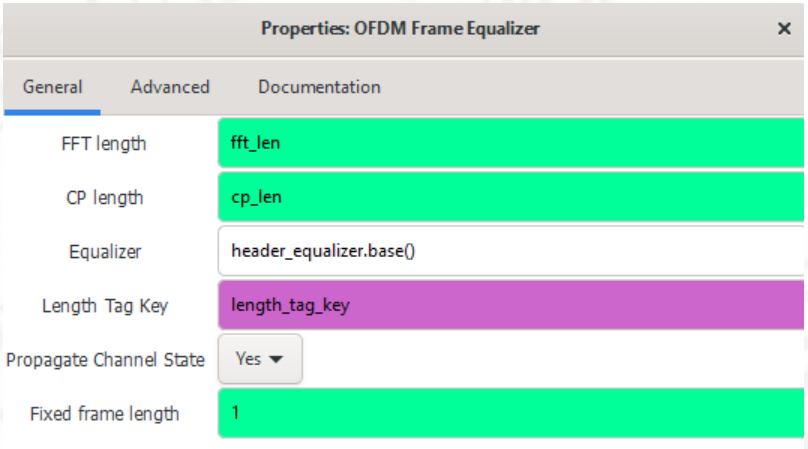

Figura 53. Configuración del bloque OFDM Frame Equalizer.

Nota. Elaboración Propia en el Software GNU Radio.

El bloque OFDM Serializer es el bloque que realiza las funciones inversas al bloque OFDM Carrier Allocator utilizado en el transmisor para aisgnar el tamaño de la FFT, los símbolos y portadoras pilotos. La función de este bloque consiste en generar los símbolos de datos complejos como un flujo etiquetado, descartando los símbolos pilotos y haciendo posible corregir un desplazamiento de portadora al ser pasado por otra etiqueta con dicho desplazamiento ("OFDM Serializer - GNU Radio", 2019).

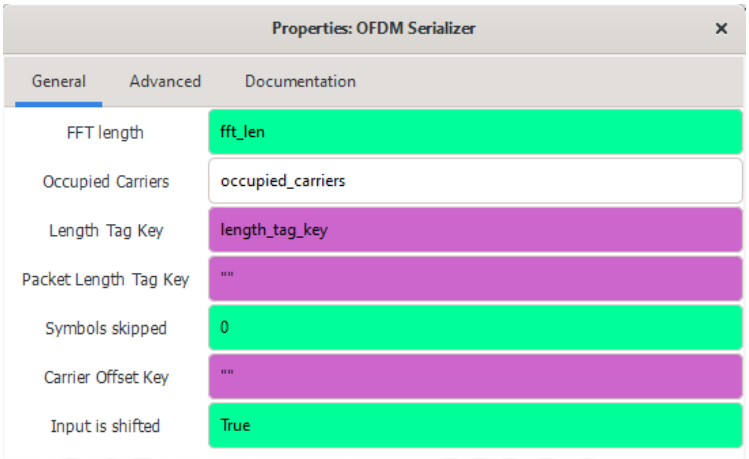

Figura 54. Configuración del bloque OFDM Serializer.

Nota. Elaboración Propia en el Software GNU Radio.

La salida del bloque OFDM Serializer se conecta con un bloque Constellation Decoder, que permitirá decodificar la constelación de la modulación que se esté trabajando siendo en este caso BPSK.

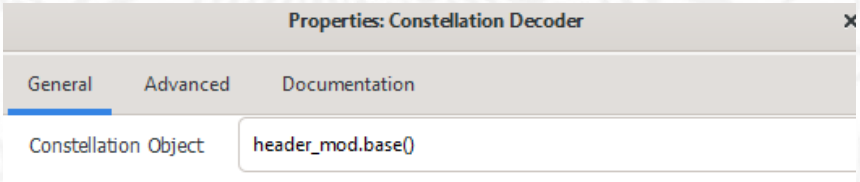

Figura 55. Configuración del bloque Constellation Decoder.

Nota. Elaboración Propia en el Software GNU Radio.

A continuación, se utiliza el bloque Packet Header Parser que genera un diccionario PMT, que se ingresa en el puerto de mensajes con el ID “Header_data” del bloque Header/Payload Demux.

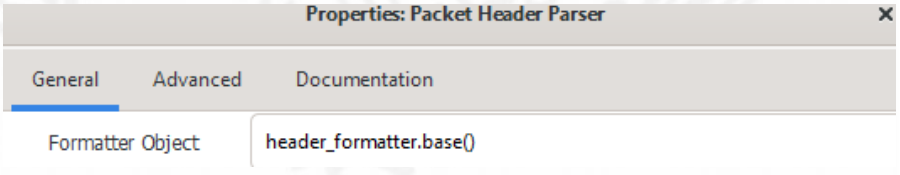

Figura 56. Configuración del bloque Packet Hedaer Parser.

Nota. Elaboración Propia en el Software GNU Radio.

4.1.2.3.3. Recuperación de Datos después de la Demodulación.

Con respecto a la recuperación de datos después de la demodulación, antes que nada, se colocó un bloque Virtual Source con el Stream ID: “Payload Stream” para identificar que la señal corresponde a la corriente de datos provenientes de Virtual Sink con el mismo Stream ID. Enseguida se conecta con un nuevo bloque FFT que toma los valores complejos de la señal, con la finalidad de transformar la señal del dominio del tiempo al de la frecuencia.

Para que el este proceso se lleve a cabo correctamente es necesario indicar la longitud FFT mediante la asignación de la variable “fft_len” (valor entero de 64). Después establecer en “Forward” la opción Forward/Reverse. Luego, no se indica el tipo de ventana dado que no se necesita realizar matemáticas de ventana. Este proceso se puede observar en la siguiente figura:

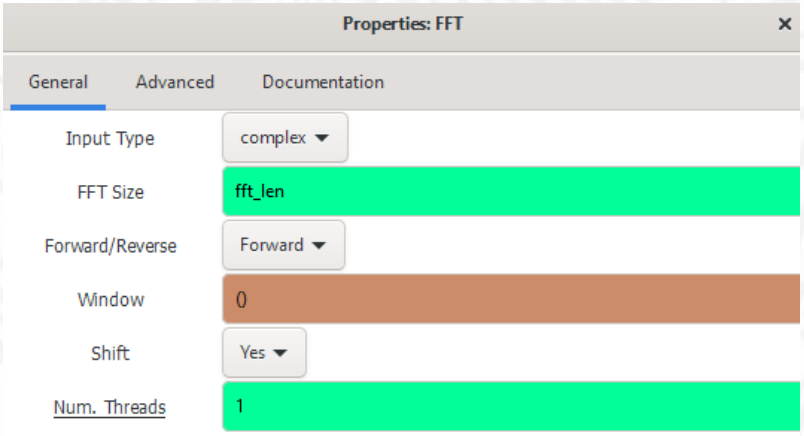

Figura 57. Configuración del bloque FFT para el virtual Source “Payload Stream”.

Nota. Fuente Propia.

A continuación, se conecta la salida del bloque FFT con la entrada de un nuevo bloque OFDM Frame Equalizer, con el propósito de eliminar el desplazamiento del portador grueso con respecto al número de portadoras. Después, se realiza la ecualización en su respectiva dimensión en un Frame OFDM etiquetado ("OFDM Frame Equalizer - GNU Radio", 2019).

Llegados a este punto, se procede a conectar la salida del bloque OFDM Frame Equalizer a la entrada de un nuevo bloque OFDM Serializer. Mediante este bloque se generan los símbolos de datos complejos como un flujo etiquetado sin símbolos piloto. No obstante, para que este proceso pueda llevarse a cabo es necesario que en el parámetro Occupied Carriers se asignen las mismas portadoras ocupadas en el bloque OFDM Carrier Allocator que se encuentra implementado en el transmisor. Además, la clave de etiqueta de longitud del paquete (Length Tag Key) y la tecla de etiqueta de longitud (Packet Length Tag Key) son necesarias para identificar el número de símbolos complejos en el paquete y la longitud del Frame de entrada en los símbolos OFDM, respectivamente ("OFDM Serializer - GNU Radio", 2019).

Por último, se agrega un bloque Virtual Sink que se conecta con la salida del bloque OFDM Serializer y se le asigna el Stream ID: “Payload IQ”.

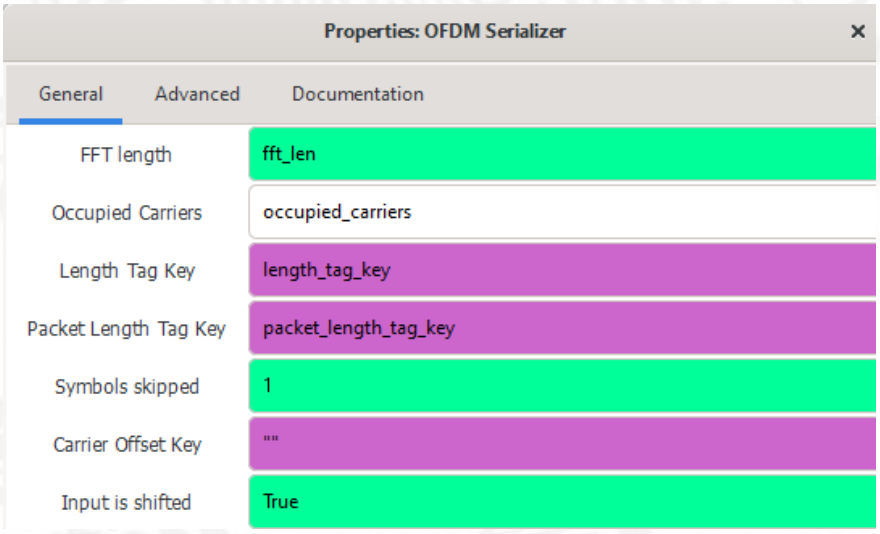

Figura 58. Configuración del bloque OFDM Serializer para el Virtual Source “Payload Stream”

Nota. Elaboración Propia en el Software GNU Radio.

4.1.2.3.4. Recuperación de Bits.

Ahora bien, para la Recuperación de Bits, se agrega un bloque Virtual Source con el Stream ID: “Payload IQ”. Después se conecta a la entrada de un nuevo bloque Constellation Decoder en donde se decodifican los puntos de una constelación desde un espacio complejo a bits (desempaquetados).

A continuación, se conecta la salida del bloque anterior con la entrada del bloque Repack Bits en donde se vuelven a empaquetar los bits de flujo de la entrada en bits de flujo de salida etiquetados ("Repack Bits - GNU Radio", 2019). Se debe tener en cuenta que el parámetro Pack Alignment se utiliza para decidir qué paquete de datos se debe alinear. Además, en cada byte de entrada nuevo, comienza a leer en el LSB y también comienza a copiar en el LSB.

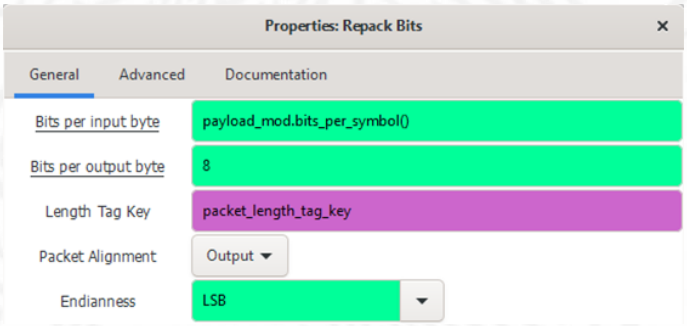

Figura 59. Configuración del bloque Repack Bits.

Nota. Elaboración Propia en el Software GNU Radio.

Una vez se han empaquetado los bits, la señal se envía a un bloque Stream CRC32 que recibe en su entrada el flujo de bytes, que forman el paquete. El primer byte tiene una etiqueta con clave “length” y el valor es el número de bytes en el paquete. Ahora bien, en la salida son los mismos bytes que los entrantes, pero la etiqueta se restablece a la nueva longitud “packet_length_tag_key” ("Stream CRC32 - GNU Radio", 2019).

Con respecto a su configuración, el parámetro Mode se establece en “Check CRC” y Packed en “Yes” ya que los datos son bits empaquetados.

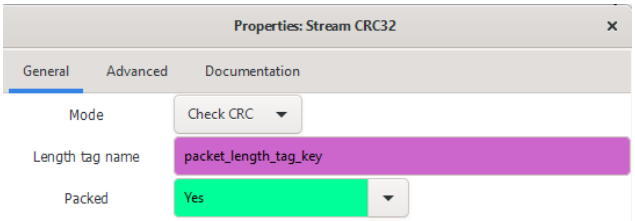

Figura 60. Configuración del bloque Stream CRC32.

Nota. Elaboración Propia en el Software GNU Radio.

A continuación, se agregó un bloque Tag Debug para que recopile todas las etiquetas recibidas desde el bloque Stream CRC32. La señal de entrada del bloque Tag Debug es mostrada en stdout de forma formateada, esto con la finalidad de poder adjuntarse y ver todas las etiquetas que salen del bloque depuradas ("Tag Debug - GNU Radio", 2019).

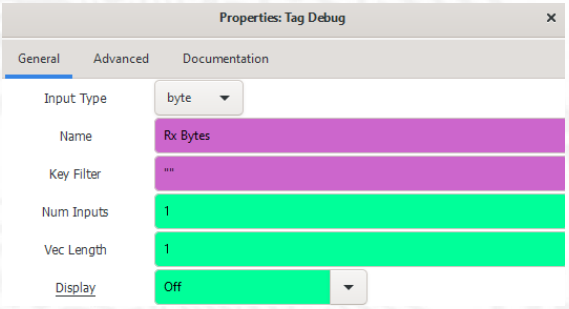

Figura 61. Configuración del bloque Tag Debug.

Nota. Elaboración Propia en el Software GNU Radio.

Por último, la salida del bloque Stream CRC32 se conectó a la entrada del File Sink utilizado para escribir una secuencia en un archivo binario que se pueda leer en cualquier entorno de programación.

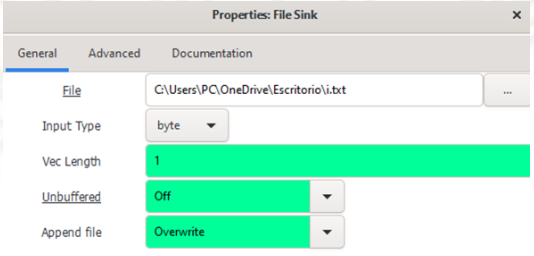

Figura 62. Configuración del bloque File Sink.

Nota. Elaboración Propia en el Software GNU Radio.

5. Resultados

El diagrama del transmisor OFDM se muestra en las Figura 63, en donde se muestra el Mapeo de los datos en paquetes para su transmisión y la implementación de la modulación OFDM y su transmisión

El Mapeo de los datos de paquetes a su vez se divide en dos partes que son el generador de paquetes y el multiplexor de la señal.

Por su parte, la implementación de la modulación OFDM y su transmisión, también se divide en dos partes, las cuales son el asignador de portadoras junto con el IFFT y CP, y la transmisión.

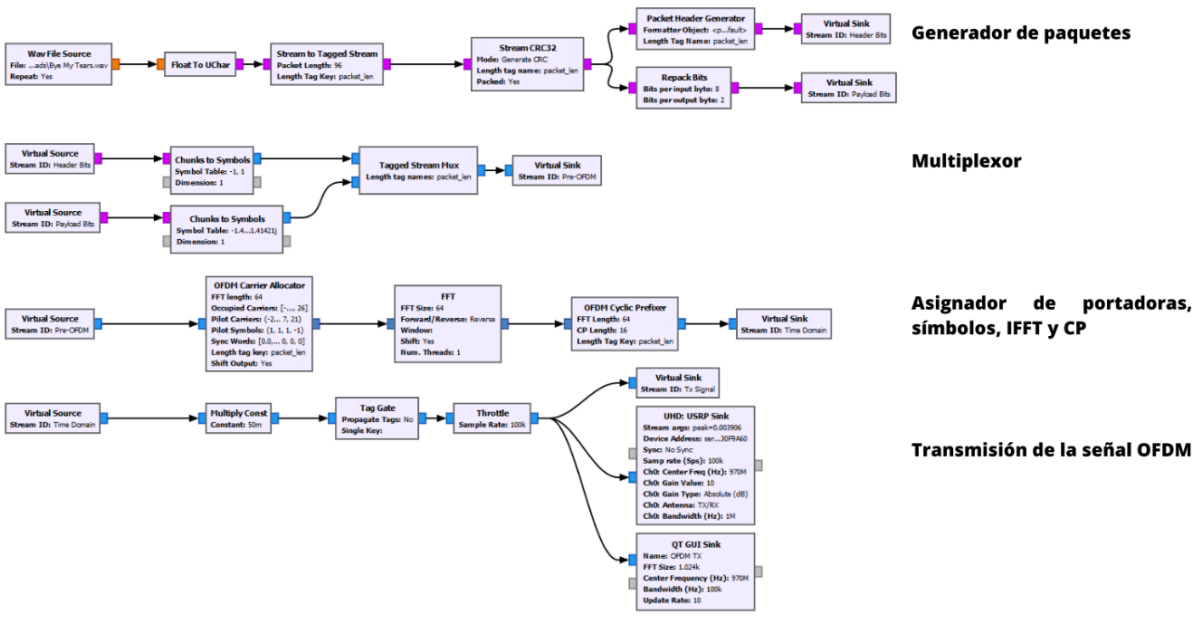

Figura 63. Transmisor OFDM.

Nota. Adaptado de Bansal, K., & Tripathi, V. (2015). *OFDM Tranmission and Reception of Packets using GNU-Radio and USRP - Communications Lab Project* [Ebook] (pp. 1-7).

La señal original que en este caso es un archivo de audio (.wav) mediante el uso de un bloque Float to Uchar se convirtió en una secuencia de bytes que no tienen una etiqueta de longitud establecida.

Después mediante el uso del bloque Stream to Tagged Stream, se convirtió la señal en una secuencia etiquetada de longitud establecida a 96. A continuación, el flujo de bytes ya empaquetados que entraron en el bloque Stream CRC32, a su primer byte se le asigno una nueva etiqueta con la clave “packet_len”, de esta manera el valor es el número de bytes en el paquete. Luego, el flujo de bytes fue ingresado a un bloque Packet Header Generator en donde se genera un encabezado predeterminado para el paquete y transmitido en función de la longitud especifica en bits ("Packet Header Generator - GNU Radio", 2019) y el Objeto formateador a un Virtual Sink con el Stream ID: “Header Bits”. Al mismo tiempo, el flujo de bytes de la salida del bloque Stream CRC32, se empaquetaron en bits comenzando su lectura en el LSB y su flujo de salida es enviado a un bloque Virtual Sink con el Stream ID: “Payload Bits”.

Ahora bien, para multiplexar la señal se utilizaron bloques Chunks to Symbol que asignaron una secuencia de índices de símbolos desempaquetados a nuestro flujo de Bits y pasándolos a una señal compleja con una dimensión establecida que a continuación son ingresadas en el bloque Tagged Stream Mux, donde los paquetes se envían secuencialmente desde cada flujo de entrada y la suma de todas las etiquetas de longitudes individuales es la señal de salida con una nueva etiqueta de longitud que se ingresó en un bloque Virtual Sink con el Stream ID: “Pre-OFDM”.

La señal “Pre-OFDM” se ingresó a un bloque OFDM Carrier Allocator convirtiéndolo en un flujo de símbolos complejos en vectores que son la entrada para la IFFT. A continuación, se aplica una conversión de serie a paralelo y se realiza la operación IFFT en los datos complejos obtenidos. La señal de salida se agrupo nuevamente, según el número de subportadoras de transmisión establecidas y fue ingresada en el bloque OFDM Cyclic Prefixer en donde se agregó un prefijo cíclico y se realizó la forma de los pulsos en los símbolos OFDM obteniendo nuestra señal en el dominio del tiempo. Como resultado se tiene la señal OFDM en el dominio del tiempo que posteriormente el flujo de datos se multiplica por una constante escalar y se evita que las etiquetas se propaguen.

Después, se utilizó un bloque Throttle, al cual se le asigno el valor de la variable “samp_rate” ´para mantener el flujo de datos a una velocidad constante.

Un bloque UHD: USRP Sink se utilizó para transformar los datos digitales en el dominio del tiempo y convertirlo en frecuencia de transmisión para enviarlos a través de un canal inalámbrico mediante el uso de la tarjeta USRP B200. Se configuraron varios parámetros como la frecuencia de muestreo, ancho de banda, frecuencia portadora, ganancia de canal, la antena, etc.

En la Figura 64, se puede observar que la señal OFDM transmitida no se encuentra tan definida, ya que al utilizar un archivo de audio (.wav), el aumento de la frecuencia de muestreo o el número de bits de cada muestra aumenta la cantidad de espacio utilizado y las frecuencias que son más de la mitad de la frecuencia de muestreo no pueden representarse correctamente en muestras digitales.

La mitad de la frecuencia de muestreo representa por lo tanto un límite superior llamada la frecuencia de Nyquist, y la forma de onda analógica debe ser completamente debajo de este límite para ser representada correctamente digitalmente.

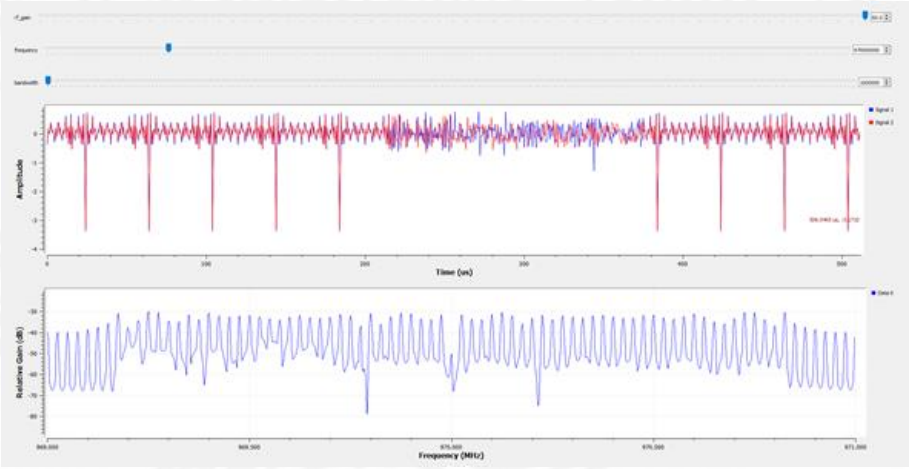

Figura 64. Señal OFDM en tiempo y frecuencia con una señal de origen de audio (.wav).

Nota. Elaboración Propia en el Software GNU Radio.

Con respecto a la Figura 65, se genera un número de muestras de números aleatorios de [min, max), lo que significa que no se incluirá el valor máximo. Esto es ideal para crear bytes de información para un modulador, de ahí que la señal OFDM este mejor definida tanto en tiempo como en frecuencia.

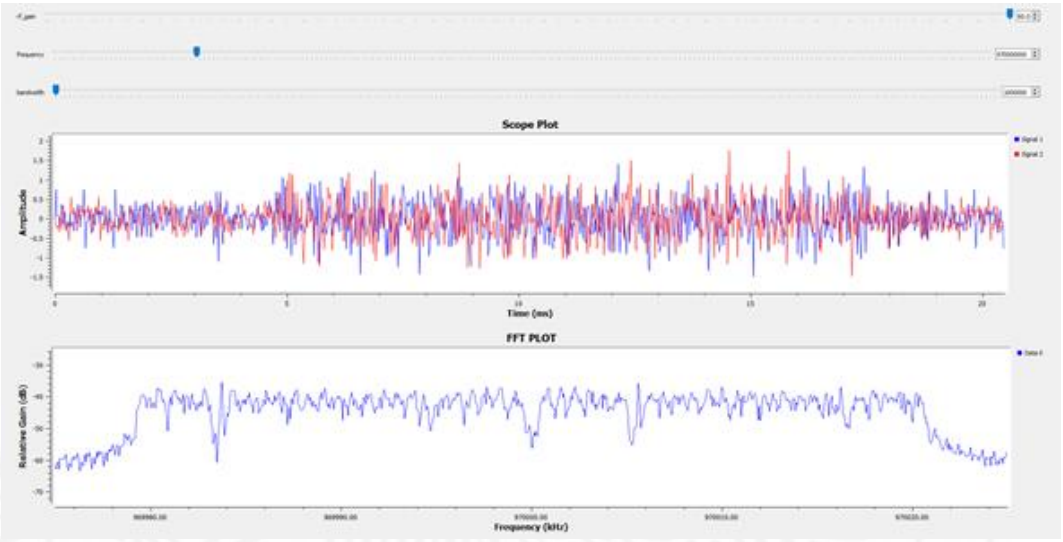

Figura 65. Señal OFDM en tiempo y frecuencia con una señal de origen "Random Source".

Nota. Elaboración Propia en el Software GNU Radio.

A fin de comprobar que la señal OFDM se transmitió en la frecuencia especifica dentro del bloque UHD: USRP Sink, se utilizó el software SDR Console v3.1. En este software se seleccionó la Radio para la recepción de la señal, en este caso fue la tarjeta HackRF One (hackRF).

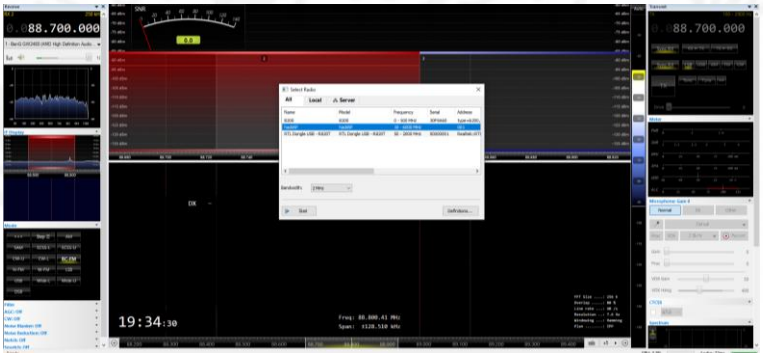

Figura 66. Selección de la Radio para la recepción.

Nota. Elaboración Propia en el Software SDR Console v3.1.

Una vez, ya ubicados en la señal de transmisión (430 MHz), se puede observar que mientras no se esté transmitiendo con la tarjeta USRP B200, no encontraremos una señal de con la suficiente potencia para ser recibida por la HackRF One.

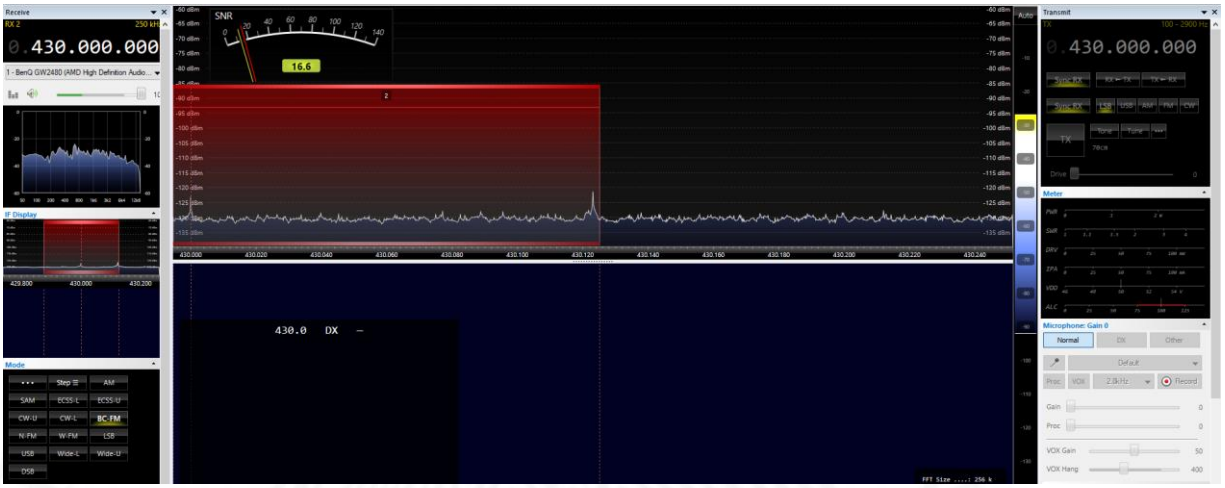

Figura 67. Tarjeta HackRF One sintonizada a la señal de transmisión.

Nota. Elaboración Propia en el Software SDR Console v3.1.

Ahora bien, en la Figura 68, se observa que la tarjeta USRP B200 transmitió la señal a la frecuencia que se especifica en el bloque UHD: USRP Sink y con esto se comprobó que la tarjeta HackRF One es capaz de recibir la señal.

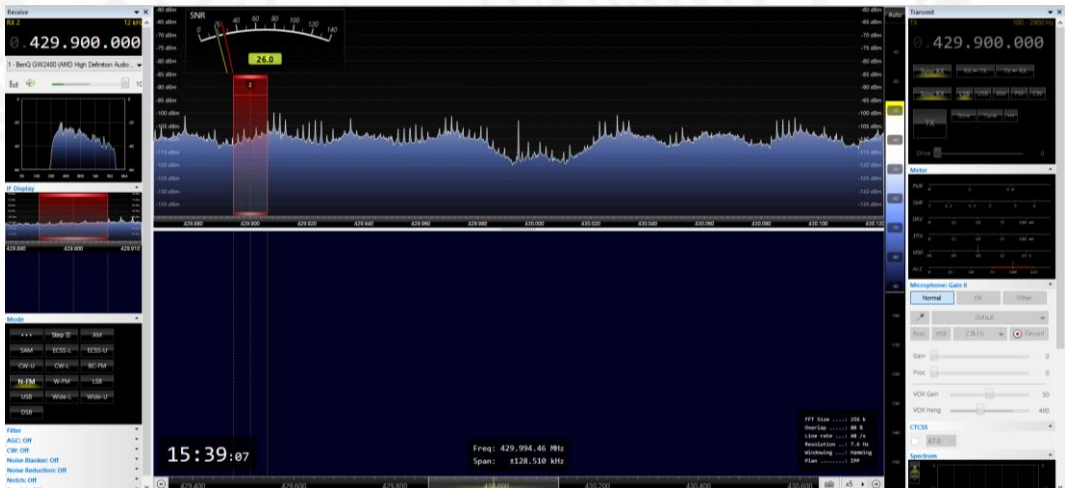

Figura 68. Recepción de la señal con la tarjeta HackRF One.

Nota. Elaboración Propia en el Software SDR Console v3.1.

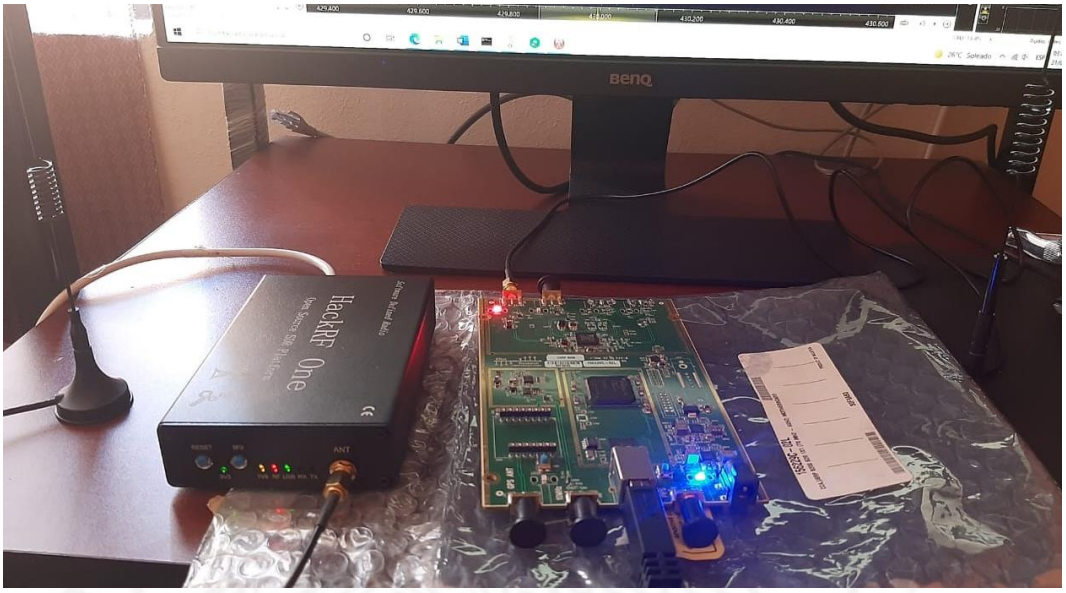

Figura 69. Tarjeta HackRF One en modo Receptor y Tarjeta USRP B200 en modo Transmisor.

Nota. Fuente Propia.

Con respecto al receptor (Figura 70), este es conformado por tres etapas, la recepción y ecualización a través de frames, la recuperación de datos después de la demodulación y la recuperación de bits. Ahora bien, la etapa de Recepción y ecualización a través de frames está dividida en dos secciones que son la Detección del símbolo OFDM y Estimación y ecualización de la señal. En primer lugar, en la etapa de Detección del símbolo OFDM se utilizó un bloque Osmocom Source que es el encargado de recibir la señal a través de un canal inalámbrico y transformarla en datos digitales en el dominio del tiempo.

Mientras se realiza la conversión descendente de la señal recibida, se realiza la sincronización de la frecuencia portadora. Después de esta conversión, se logra la sincronización del tiempo de los símbolos. A continuación, mediante el bloque Header/Payload Demux se realizó la demultiplexación de los paquetes recibidos con una longitud aún por determinar. Este bloque pasará la sección de cabecera a otros bloques para su demodulación. Usando la información del encabezado, luego se generará la carga útil.

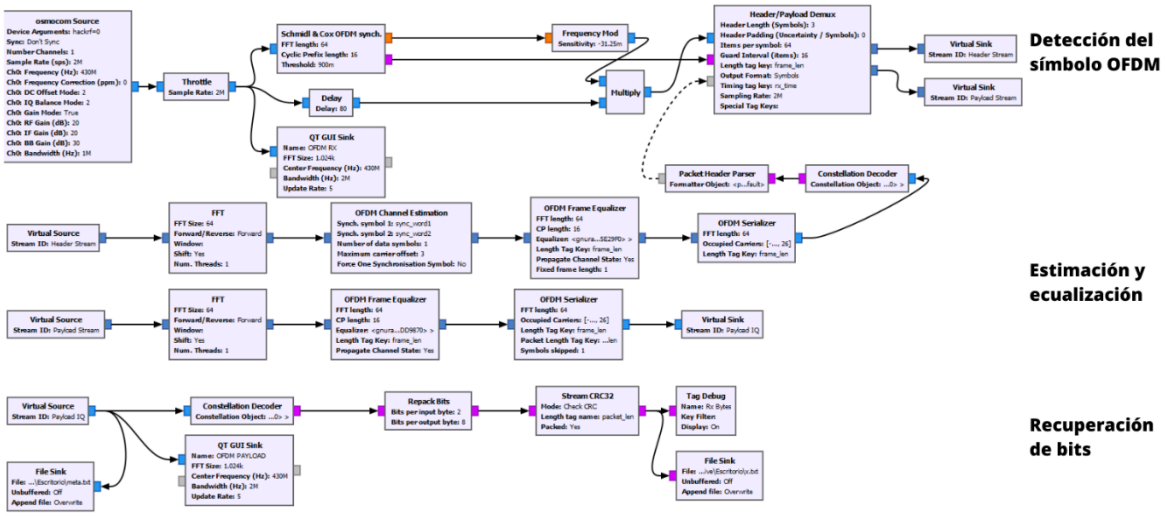

Figura 70. Receptor OFDM.

Nota. Adaptado de Bansal, K., & Tripathi, V. (2015). *OFDM Tranmission and Reception of Packets using GNU-Radio and USRP - Communications Lab Project* [Ebook] (pp. 1-7).

Ahora veamos la estimación y ecualización del canal y la señal. En primer lugar, la señal de salida del bloque Virtual Source con el Stream ID: “Header Stream” fue procesada a través de un bloque FFT para la demodulación de la señal. Después, la estimación del canal se realiza utilizando un bloque OFDM Channel Estimation que utiliza los primeros símbolos (símbolos de sincronización), para estimar el desplazamiento de frecuencia gruesa y los toques iniciales del ecualizador ("OFDM Channel Estimation - GNU Radio", 2019). Luego se utilizó un bloque OFDM Frame Equalizer que primero elimina el desplazamiento del portador grueso y después realiza la ecualización en una dimensión de un frame OFDM etiquetado de modo que se permita generar los símbolos de datos complejos como un flujo etiquetado, descartando los símbolos piloto que se agregaron en el transmisor OFDM. Mediante las estimaciones se obtienen los datos complejos que se desmapean de acuerdo al diagrama de constelaciones de transmisión que son enviados mediante el bloque Packet Header Parser a la entrada header_data del bloque Header/Payload Demux, de esta manera el demodulador de encabezado tiene la opción de especificar un desplazamiento de carga útil.

A continuación, se realizó el mismo procedimiento con la señal de salida del bloque Virtual Source con el Stream ID: “Payload Stream”, omitiendo el uso del bloque OFDM Channel Estimation y enviando la señal de salida del bloque OFDM Serializer (flujo etiquetado sin símbolos piloto) a un Virtual Sink con el Stream ID: “Payload IQ” para la recuperación de los bits más adelante.

Después, el flujo de datos de salida del bloque Virtual Sink con el Stream ID: “Payload IQ” se desmapean de acuerdo al diagrama de constelaciones de transmisión y son empaquetados en bits donde en cada byte de entrada nuevo se comienza a copiar en el LSB. Para concluir, se comprueba el CRC y se le asigna un nuevo valor de bytes al paquete con la finalidad de ser recibidos en un bloque Tag Debug y ver todas las etiquetas que salen de ese bloque con fines de depuración.

Con la intención de demostrar que la señal OFDM es recibida, en la Figura 71 se muestra la señal en el dominio de la frecuencia y en la Figura 72 en el dominio del tiempo.

Se puede observar del lado izquierdo la señal recibida por la tarjeta SDR HackRF One y del lado derecho la señal transmitida a través de la tarjeta USRP B200 en ambos casos.

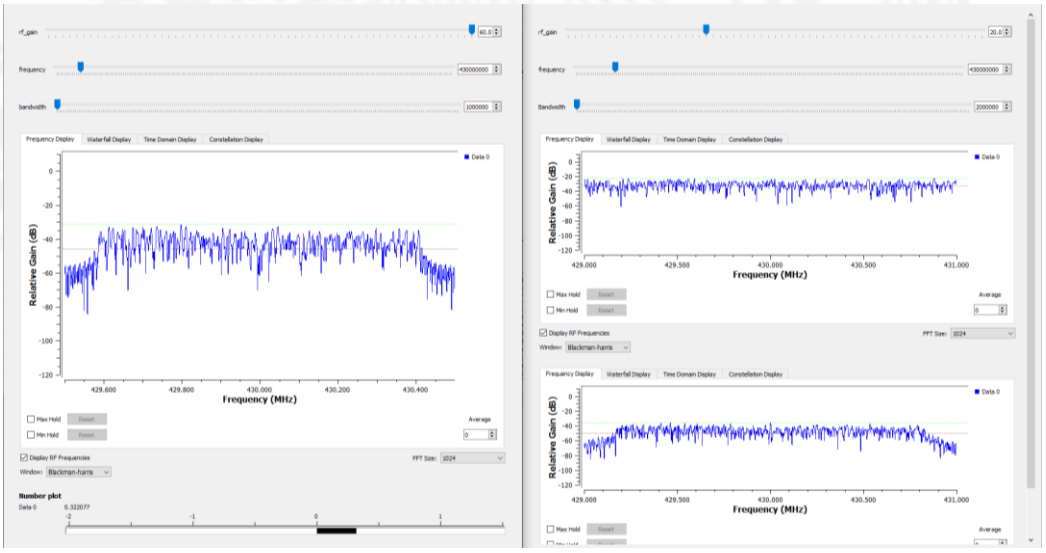

Figura 71. Transmisión y Recepción de la señal OFDM en el dominio de la Frecuencia.

Nota. Elaboración Propia en el Software GNU Radio.

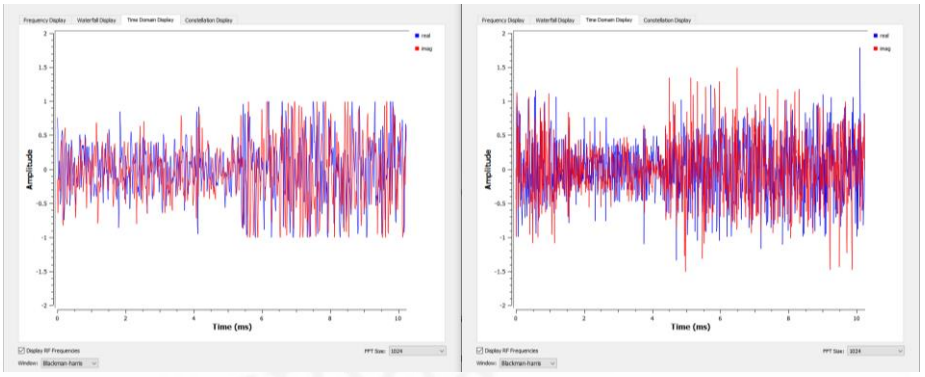

Figura 72. Transmisión y Recepción de la señal OFDM en el dominio del Tiempo.

Nota. Elaboración Propia en el Software GNU Radio.

Llegados a este punto, ya es posible recuperar los bits de la señal original en un archivo de txt; para esto se utilizó el bloque File Sink (Figura 73) que se encargó de sobrescribe la información existente de un archivo txt con la información procedente de la demodulación de la señal OFDM.


```
Tag Debug: Rx Bytes
nput Stream: 00
 Offset: 1248  Source: n/a    Key: packet_num  Value: 4027
 Offset: 1248  Source: n/a    Key: rx_time  Value: {69 0.434736}
 Offset: 1248  Source: n/a    Key: ofdm_sync_carr_offset  Value: 0
 Offset: 1248  Source: n/a    Key: ofdm_sync_chan_taps  Value: #[(0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (0.239023,1.31937) (1.12205,0.763412) (1.32394,-0.321374) (0.780751,-1.12678)
-0.289726,-1.25267) (0.405721,-1.37858) (-1.34185,0.328033) (-0.743968,1.15722) (0.294234,1.33021) (1.15805,0.781645) (1.32422,-0.293857) (0.80801,-1.146) (-0.300348,-1.33419)
-1.13172,-0.791427) (-1.35535,0.259119) (-0.798665,1.18966) (0.215425,1.40462) (0.104566,0.0907116) (0.142138,-0.00632583) (1.26854,0.679504) (0.0297821,-0.172072)
-0.120771,-0.130047) (-0.159543,-0.00663234) (-0.150454,0.0994119) (-0.0327472,0.189194) (0.147779,0.140559) (0,0) (0.159952,-0.133042) (0.0237442,-0.168411) (-0.0887178,-0.125699)
-0.15561,-0.0598509) (-0.138034,0.109671) (-0.0610417,0.164844) (-0.758991,1.18544) (0.167128,0.034907) (0.147667,-0.100582) (0.0652146U,-0.175025) (-0.0696408,-0.116834)
-0.142061,-0.0589698) (-0.129835,0.0884549) (-0.0319073,0.137842) (0.0876525,0.140234) (0.179451,0.0835504) (0.116757,-0.0762958) (0.0677346,-0.144506) (-0.0579639,-0.122408)
-0.175675,-0.0642217) (-1.15005,-0.825014) (-0.044556,0.175489) (0.104432,0.155446) (0.138479,0.0470407) (0.126234,-0.0986301) (0.0164624,-0.150435) (0,0) (0,0) (0,0) (0,0) (0,0)]
 Offset: 1248  Source: n/a    Key: packet_len  Value: 96
----------------------------------------------------------------------
JUUU
----------------------------------------------------------------------
Tag Debug: Rx Bytes
nput Stream: 00
 Offset: 1344  Source: n/a    Key: packet_num  Value: 1236
 Offset: 1344  Source: n/a    Key: rx_time  Value: {70 0.06596}
 Offset: 1344  Source: n/a    Key: ofdm_sync_carr_offset  Value: 0
 Offset: 1344  Source: n/a    Key: ofdm_sync_chan_taps  Value: #[(0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (0.237217,-1.12682) (-1.03068,-1.1572) (-1.29164,-0.0313028) (-1.00054,1.11009)
0.355158,1.09562) (-0.926956,1.14574) (1.15911,-0.499818) (0.298268,-1.46525) (-0.834882,-0.9636) (-1.30586,0.168063) (-0.85544,1.09948) (0.495447,1.28496) (1.25326,0.593458)
1.15688,-0.401436) (0.505913,-1.31339) (-0.856079,-1.24546) (-1.34891,-0.262814) (-0.243844,0.0482102) (-0.0487506,0.201023) (-1.24862,0.72293) (0.121443,0.0179965)
0.164255,-0.166745) (0.025817,-0.174819) (-0.112469,-0.118483) (-0.216221,0.0489327) (-0.0890539,0.19273) (0,0) (0.206298,0.142161) (0.155523,-0.118372) (0.0161286,-0.16825)
-0.119877,-0.126289) (-0.19426,-0.00689958) (-0.108269,0.0991515) (-1.35059,0.0943688) (0.181295,0.0633277) (0.117673,-0.0421488) (0.0731077,-0.156581) (-0.0731561,-0.138805)
-0.166732,0.0131144) (-0.112377,0.143099) (-0.0143819,0.145976) (0.103104,0.112592) (0.130204,-0.035717) (0.0671876,-0.162141) (-0.0383654,-0.159855) (-0.136726,-0.08954)
-0.18036,0.0644532) (-1.42238,-0.202999) (0.119465,0.124375) (0.16737,-0.00314113) (0.0868358,-0.110737) (-0.0667063,-0.133009) (-0.127342,-0.0668001) (0,0) (0,0) (0,0) (0,0) (0,0)]
 Offset: 1344  Source: n/a    Key: packet_len  Value: 96
----------------------------------------------------------------------
J
----------------------------------------------------------------------
Tag Debug: Rx Bytes
nput Stream: 00
 Offset: 1440  Source: n/a    Key: packet_num  Value: 3131
 Offset: 1440  Source: n/a    Key: rx_time  Value: {70 0.987668}
 Offset: 1440  Source: n/a    Key: ofdm_sync_carr_offset  Value: 0
 Offset: 1440  Source: n/a    Key: ofdm_sync_chan_taps  Value: #[(0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (-1.27707,-0.0414093) (-0.777565,1.02323) (0.700137,1.25135) (1.39715,0.119142)
0.729862,-1.13952) (-0.0577746,-1.34126) (-1.38728,-0.172803) (-0.811904,1.05297) (0.508128,1.16374) (1.49211,0.215425) (0.733838,-1.14955) (-0.523291,-1.33352) (-1.38782,-0.147279)
-0.897098,0.967829) (0.36254,1.25393) (1.44013,0.341186) (1.01781,-0.915827) (-0.030415,-0.164632) (-0.163138,-0.0410974) (-1.31885,0.316769) (0.0191838,0.143322)
0.15746,0.066167) (0.127167,-0.105581) (0.00319949,-0.174029) (-0.143708,-0.117353) (-0.232182,0.089903) (0,0) (0.110501,0.143434) (0.207674,-0.0390156) (0.0393241,-0.178908)
-0.123891,-0.10389) (-0.158696,0.0563489) (-0.070276,0.134027) (0.583022,1.33008) (0.169017,-0.0425598) (0.115386,-0.147779) (-0.0837013,-0.13718) (-0.143432,-0.00679813)
-0.0924595,0.160656) (0.0565475,0.133432) (0.156482,-0.00866684) (0.0834667,-0.144618) (-0.0444189,-0.138199) (-0.163432,-0.00632823) (-0.082471,0.17472) (0.0910544,0.146568)
0.148733,-0.0118117) (1.1608,-0.811813) (-0.0715676,-0.1278) (-0.140176,-0.0180168) (-0.134823,0.12707) (0.0741895,0.146244) (0.140245,0.0113245) (0,0) (0,0) (0,0) (0,0) (0,0)]
 Offset: 1440  Source: n/a    Key: packet_len  ValueU:U 96
```


Figura 73. Paquetes de bytes recibidos en el Receptor.

Nota. Elaboración Propia en el Software GNU Radio.

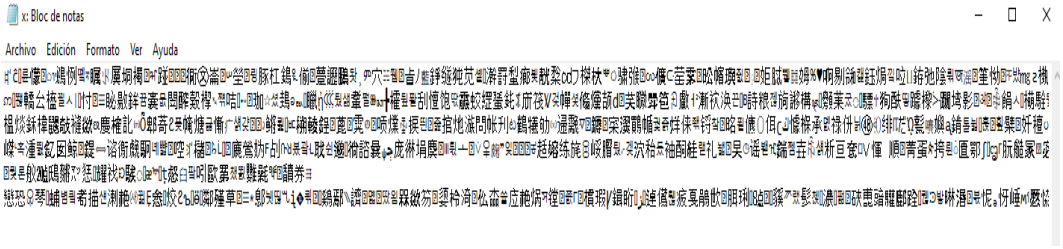

Figura 74. Recuperación de los bytes en formato UTF-16 LE.

Nota. Elaboración Propia en el Bloc de Notas de Windows 10.

A continuación, se convirtió la información de los bytes recuperados en formato UTF-16 LE en binario, lo cual permite observar que los bytes recuperados se encuentran en paquetes de 32 bytes, es decir el tamaño máximo es de 256 que en el caso de cambiar la señal de audio (.wav) como señal de origen a una fuente de números Random (Random File Source) se podrían recibir sin ningún problema.

Figura 75. Conversión de los bytes recuperados de formato UTF-16 LE al sistema Binario.

Nota. Elaboración Propia en el Bloc de Notas de Windows 10.

6. Evaluación o impacto económico

El impacto económico de este proyecto es a nivel comunidad en específico, dirigido al departamento de Ingeniería Electrónica del Tecnológico Nacional de México Campus Oaxaca en lo que respecta a dar solución al proyecto de “Sismología y Riesgo Sísmico”.

Para ser más específicos, consiste en reducir los gastos de la adquisición de un sistema de comunicaciones de RF capaz de transmitir y recibir diferentes formatos de datos en tiempo real provenientes de diferentes áreas dentro de la institución. De ahí que, al optarse por el uso de softwares de código abierto como es el caso de GNU Radio, no es necesario la adquisición de licencias que en muchos casos pueden llegar a sobrepasar el costo asignado al proyecto por licencia de programa. Asimismo, se puede modelar e implementar una gran variedad de diseños a bajo costo, compatibles con una gran variedad de dispositivos que se encuentran en el mercado como la HackRF One, RTL-SDR, LoRa, etc.

Ahora bien, con este proyecto se consigue un Impacto Directo ya que presenta una utilidad ofrecida del producto final al Tecnológico Nacional de México Campus Oaxaca. Dicho lo anterior, los impactos son positivos debido a que el Multiplexor SDR-OFDM ofrece diferentes ventajas como calidad y velocidad de transmisión, aprovechamiento del ancho de banda al dividirlo en varias subportadoras, eficiencia energética, rendimiento y eficiencia espectral.

Por lo cual, se cuenta con las características necesarias para establecer un sistema de comunicación capaz de satisfacer las demandas de tráfico actual en tecnologías 5G e IOT.

A través de la puesta en marcha del Multiplexor SDR-OFDM, se logrará beneficiar a la mayoría de los departamentos del Tecnológico Nacional de México Campus Oaxaca porque les permitirá obtener datos de relevancia de dispositivos como sensores y bases de datos en donde se almacene la información de prácticas y proyectos realizados tanto por estudiantes como por docentes, solo necesitando una computadora con el respectivo programa y una tarjeta de SDR compatible con el sistema, reduciendo la necesidad de adquisición de diferentes sistemas de comunicaciones para cada departamento.

7. Conclusiones y recomendaciones

En conclusión, el objetivo de este trabajo fue implementar un multiplexor OFDM para IOT y 5G utilizando GNU Radio y tarjetas SDR, que se pueda utilizar para la transmisión de diferentes formatos de datos de origen como lo son del tipo txt, wav, video, csv, etc., los cuales son utilizados en diferentes áreas de las telecomunicaciones como lo son la transmisión de información obtenida de una gran variedad de sensores (csv y txt).

Además, este trabajo se puede utilizar como base para el diseño e implementación de nuevos algoritmos de módems OFDM que permitan una mejor transmisión, recepción y demodulación de los datos que se desean transmitir.

Por su parte, los resultados obtenidos muestran que las tarjetas SDR utilizadas, proporcionan una gran flexibilidad en cuanto al diseño y prueba de un sistema de comunicación, ya que mediante la utilización de los bloques dentro del software GNU Radio, se pueden manipular en tiempo real diferentes variables como lo son la ganancia de la tarjeta, la antena utilizada para transmitir, el serial del dispositivo que realizara la transmisión o la recepción, el ancho de banda, la tasa de muestreo y el análisis de la señal en el dominio del tiempo y/o en de la frecuencia.

Como recomendación, los bloques utilizados en el transmisor OFDM, algunos de ellos como lo es el caso del OFDM Carrier Allocator necesitan un tamaño de buffer mínimo mayor o igual a 50kHz para que durante la simulación no se presenten errores. Además, se necesita de un computador con las suficientes características para soportar la transmisión y/o recepción de la señal, debido a que en algunos casos puede llegar a cerrarse el programa o no se puede visualizar los bloques de análisis de la señal.

Ahora bien, la señal OFDM transmitida y recibida en lo que respecta al dominio de la frecuencia y del tiempo, presentan ciertas variaciones ya que se están utilizando dos tarjetas SDR diferentes como lo son la USRP B200 y la HackRF One, las cuales tienen características diferentes en cuanto a la tasa de muestreo, la ganancia máxima y el ancho de banda disponible para transmitir y recibir.

Además, las antenas utilizadas no cuentan con las características necesarias para realizar una transmisión y recepción a mayor distancia, por lo cual el alcance es uno de los principales problemas.

8. Fuentes de Información

[1] *Ettus USRP B200: 1x1, 70MHz-6GHz SDR/Cognitive Radio*. Digilent. (2021). Consultado el 19 de enero de 2021, en https://digilent.com/shop/ettus-usrp-b200-1x1-70mhz-6ghz-sdr-cognitive-radio/.

[2] Jaramillo, N., & Ochoa, A. (2017). Tecnología 5G. *Ingeniería, Matemáticas y Ciencias De La Información*, (8), 41-45. Consultado el 19 de enero de 2021, de http://dx.doi.org/10.21017/rimci.2017.v4.n8.a31.

[3] *Radiación: Redes móviles 5G y salud.* Who.int. (2021). Consultado el 19 de enero de 2021, de https://www.who.int/news-room/questions-and-answers/item/radiation-5g-mobile-networks-and-health.

[4] MILENIO. (2021). ¿Qué es la red 5G y qué beneficios traerá? Consultado el 19 de Enero de 2021, en https://www.milenio.com/tecnologia/red-5g-en-mexico-que-es-y-cuales-son-su-ventajas-y-riesgos.

[5] OFDM Carrier Allocator - GNU Radio. (2020). Consultado el 20 de enero de 2021, en https://wiki.gnuradio.org/index.php/OFDM_Carrier_Allocator

[6] YAKOTT. (2019). *GNU Radio Companion + HackRF: Osmocom Source Block* [Video]. Consultado el 21 de enero de 2021, de https://www.youtube.com/watch?v=2Brjd78c6gU&list=PLZIQVCPY0T5ak1Z3aYsDmxeQTKZAireht&index=11.

[7] YAKOTT. (2020). *GNU Radio Companion + USRP B210: UHF USRP Sink Block* [Video]. Consultado el 22 de enero de 2021, en https://www.youtube.com/watch?v=Xl4xZHIuhS8.

[8] *Paquete rx fg*. (2021). [Imagen]. Consultado el 22 de enero de 2021, de https://wiki.gnuradio.org/index.php/File:Packet_rx_fg.png.

[9] *Tutorial Básico ofDM - GNU Radio*. Wiki.gnuradio.org. (2020). Consultado el 22 de enero de 2021, en https://wiki.gnuradio.org/index.php/Basic_OFDM_Tutorial.

[10] Bansal, K., & Tripathi, V. (2015). *OFDM Tranmission and Reception of Packets using GNU-Radio and USRP - Communications Lab Project* [Ebook] (pp. 1-7). IIT. Consultado el 22 de enero de 2021, de https://www.ee.iitb.ac.in/student/~vishrant/ofdm-tranmission-reception.pdf.

[11] Tekin, T., & Karakaya, B. (2018). Implementación SDR del transmisor y receptor OFDM en caso de compensación de tiempo y frecuencia. *2018 26Th Signal Processing And Communications Applications Conference (SIU),* 1-4. https://doi.org/10.1109/siu.2018.8404269

[12] Luque, J., Tume, J., & Meloni, L. *An SDR implementation of an OFDM transmitter and receiver in GNU Radio* [Ebook] (pp. 1-4). UNICAMP. Consultado el 23 de enero de 2021, de https://www.ccp-br.fee.unicamp.br/JP3I_website/2ndJP3I-papers-final/Paper15-final.pdf.

[13] Galvis, A., & De Sanctis Gil, L. *SDR: La alternativa para la evolución inalámbrica a nivel físico* [Ebook] (pp. 1-8). GIDATI. Consultado el 23 de enero de 2022, de http://roboticslab.uc3m.es/publications/Articulo1.pdf.

[14] *GNU Radio - Wikipedia, la enciclopedia libre*. Es.wikipedia.org. Consultado el 23 de enero de 2022, de https://es.wikipedia.org/wiki/GNU_Radio.

[15] Soo, Y., Young, W., & Kim, J. (2010). *MIMO-OFDM Wireless Communications with MATLAB* (2ª ed., pp. 111-162). Juan Wiley.

[16] F. Manavi e Y. R. Shayan, *"Implementation of OFDM modem for the physical layer of IEEE 802.11a standard based on Xilinx Virtex-II FPGA*, 2004 IEEE 59th Vehicular Technology Conference. VTC 2004-Primavera (IEEE Cat. No.04CH37514), 2004, pp. 1768-1772 Vol.3, doi: 10.1109/VETECS.2004.1390560.

[17] Monn, S. (2021). *Modern Digital Radio Communication Signals and Systems* (2ª ed., pp. 187-192). Salmer.

[18] *USRP Hardware Driver y USRP Manual: Transport Notes*. Files.ettus.com. (2016). Consultado el 23 de enero de 2022, en https://files.ettus.com/manual/page_transport.html#transport_usb_installwin.

[19] Tsai, P. (2007). *Diseño del receptor de banda base OFDM para comunicaciones inalámbricas* (pp. 17-28). John Wiley & Sons.

[20] *INSTITUTO FEDERAL DE TELECOMUNICACIONES*. Ift.org.mx. (2022). Consultado el 24 de enero de 2022, en http://www.ift.org.mx/sites/default/files/conocenos/pleno/sesiones/acuerdoliga/dof131119648acc.pdf.

[21] Instituto Federal de Telecomunicaciones. (2018). *Inventario de Bandas de Frecuencias Clasificadas como Espectro Libre* (pp. 23-25). ift.

[22] Instituto Federal de Telecomunicaciones. (2019). *Descripciones Técnicas de las Bandas de Frecuencias Incluidas en el Programa Anual de Uso y Aprovechamiento de Bandas de Frecuencias 2019* (p. 3). ift.

[23] López, Raikel Bordón, & Montejo Sánchez, Samuel. (2015). La Radio Cognitiva y su Impacto en el Uso Eficiente del Espectro de Radio. *Ingeniería Electrónica, Automática y Comunicaciones*, *36*(1), 42-55. Recuperado en 18 de mayo de 2022, de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59282015000100004&lng=es&tlng=es.

[24] Bansal, K., & Tripathi, V. (2015). *OFDM Tranmission and Reception of Packets using GNU-Radio and USRP - Communications Lab Project* [Ebook] (pp. 1-7). Consultado el 23 de enero de 2022, de https://www.ee.iitb.ac.in/student/~vishrant/ofdm-tranmission-reception.pdf.

[25] Montenegro, G., Rodríguez, V., & Castillo, F. (2019). *Radio definido por software, futuro de las comunicaciones inalámbricas* [Ebook] (pp. 1-6). UASLP. Consultado el 18 de mayo de 2022, desde http://www.uaslp.mx/Comunicacion-Social/Documents/Divulgacion/Revista/Quince/234/234-05.pdf.

[26] *HackRF Radio definida por Software One SDR, kit de placa base de 1MHz a 6GHz, placa de desarrollo*| *Tablero de demostración*|. aliexpress.com. (2022). Consultado el 18 de mayo de 2022, de https://es.aliexpress.com/item/4000852849224.html.

[27] *Wav File Source - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 18 de mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Wav_File_Source.

[28] *Float To UChar - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 18 de mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Float_To_UChar.

[29] *Stream to Tagged Stream - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 18 de mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Stream_to_Tagged_Stream.

[30] *Stream CRC32 - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 18 de mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Stream_CRC32.

[31] *Packet Header Generator - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 18 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Packet_Header_Generator.

[32] *Repack Bits - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 18 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Repack_Bits.

[33] *Chunks to Symbols - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 19 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Chunks_to_Symbols.

[34] *Tagged Stream Mux - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 19 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Tagged_Stream_Mux.

[35] *OFDM Carrier Allocator - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 19 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=OFDM_Carrier_Allocator.

[36] *Multiply Const - GNU Radio*. Wiki.gnuradio.org. (2018). Consultado el 19 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Multiply_Const.

[37] *OFDM Channel Estimation - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 19 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=OFDM_Channel_Estimation.

[38] *OFDM Frame Equalizer - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 19 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=OFDM_Frame_Equalizer.

[39] *OFDM Serializer - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 19 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=OFDM_Serializer.

[40] *Tag Debug - GNU Radio*. Wiki.gnuradio.org. (2019). Consultado el 19 de Mayo de 2022, en https://wiki.gnuradio.org/index.php?title=Tag_Debug.