ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DE COMUNICACIÓN INALÁMBRICO OFDM EN SDR

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

FAUSTO VINICIO LINCANGO OÑA

DIRECTOR: ING. JORGE EDUARDO CARVAJAL RODRIGUEZ, MSc.

CODIRECTOR: Dr. DIEGO JAVIER REINOSO CHISAGUANO

Quito, octubre 2018

AVAL

Certificamos que el presente trabajo fue desarrollado por Fausto Vinicio Lincango Oña, bajo nuestra supervisión.

ING. JORGE EDUARDO CARVAJAL RODRIGUEZ, MSc
DIRECTOR DEL TRABAJO DE TITULACIÓN

Dr. DIEGO JAVIER REINOSO CHISAGUANO
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Fausto Vinicio Lincango Oña, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Fausto Vinicio Lincango Oña

DEDICATORIA

Escribir unos cuantos párrafos a las personas que han sido trascendentes en toda esta etapa de mi vida me parece poco, sin embargo, ñaño, mami, papi, Miriancita y el resto de mi familia, gracias por estar presentes en mi vida, este logro es conjunto y se los dedico.

Para mi madre, mamita tu ejemplo de lucha, constante superación y perseverancia no han dejado de ser mi inspiración. Dios me dio la gran bendición de ser tu hijo, todo este camino has sido mi ayuda incondicional te amo. Proverbios 31:10

Para mi hermano Ariel, gracias por ser un guerrero e inspirarme a continuar, las más grandes lecciones de la vida se aprenden al pasar por fuego, tu vida es un regalo inmenso, te amo hermano. Salmos 133.1

Para mi padre, para enseñar no es necesario hablar todo el tiempo, las enseñanzas de vida que marcan a los demás se hacen con el ejemplo. ¡Gracias pa te amo! Proverbios 17:6

Para mi tía que más que mi tía es mi segunda madre, gracias Miriancita, he aprendido que la familia está presente a pesar de los problemas y las circunstancias, la quiero mucho, gracias por creer en mí y confiar en que lo lograría. Proverbios 31:29

Querida familia, ustedes han estado presentes a pesar de todas las circunstancias que han ocurrido, he podido observar que el amor y el cariño de todos ha sido un impulso para continuar y no abandonar, no me cansare de decir que este logro es conjunto para todos NOSOTROS. Hechos 16:31

AGRADECIMIENTO

La vida está llena de momentos que marcan a las personas, le doy gracias a Dios pues todo ha sido bueno y me ha dado la oportunidad de vivir una vida llena de momentos especiales, momentos junto a personas muy especiales en mi vida. Todo lo que hasta ahora he tenido, he aprendido y demás cosas, han sido porque él ha estado junto a mí. Las palabras no serían suficientes para agradecer por todo, gracias Dios…EBENEZER.

A mi director, Ingeniero Jorge Carvajal, gracias por su compromiso, paciencia, ayuda y guía, en verdad supo cómo apoyarme en la realización de este trabajo. A mi codirector, Doctor Diego Reinoso, gracias por estar presto a apoyarme, por su dedicación y compromiso, fue gran respaldo durante todo este proceso.

Gracias familia por estar ahí, por estar conmigo, por estar presente. ¡Papi gracias por ser ese ejemplo vivo de responsabilidad! Mami, gracias por ser ese apoyo incondicional, por tu ejemplo de lucha incansable, dedicación y entrega te amo. Ñaño, con tu vida me has enseñado como afrontar las dificultades y que las adversidades pueden ser una motivación para continuar, ¡eres mi ejemplo de vida!, aunque soy el mayor, me has enseñado muchas cosas.

A mis amadas tías, Mirian, Elizabeth, Cecilia (Suquita), Nelly, Pilita, mi tío Diego, mi tío Fernando, mi tío Nelson y mi querido tío Washo y mi papito Alfredo, gracias por amarme y a pesar de todo seguir siendo una familia, las circunstancias han hecho que la familia sea fuerte y el amor sea real. Les agradezco a mis primos, Dieguito, Nenita, Juanito y Daniel por ser mis compañeros de locuras, juegos, Luigui gracias por ser un primo, un amigo y por confiar en mí, ñaño Nelson gracias por ser el ejemplo de nosotros los menores, te admiro mucho.

Ñaña Anina, Sandrita, ñaño Victor (tucoman), ñaño coquin, mimosito, y mis abuelitos Jorge y Magdalena, siempre hay una ocurrencia o una historia si algo es seguro es que la alegría nunca esta demás, los amo, gracias porque ese amor está presente a pesar de las distancias. ¡Los amo mucho!, A mis primos de igual manera, Andreita, Sebitas, Puka, los amo. ¡Gracias por todo!

Tío Ramiro, Tía Ceci, gracias porque el amor va más allá de cualquier circunstancia, los quiero mucho.

¡Mi Silvita!! Gracias por ser una mamá conmigo, te quiero mucho. Somos familia, los quiero mucho, han sido una bendición que Dios me dio, también a mis ñañas Majito y Andre y Edwincito, los quiero mucho y gracias por todo el cariño.

Gracias a mis amigos y hermanos David L., Caty y Rebe, su amor y discipulado me enseño mucho, Jou gracias por ser tan tú, las conversaciones y confianza. Comandantes Nelson y Esthela, gracias por confiar, creer en mí y darme la oportunidad de servir, mi vida quedó marcada, los quiero mucho.

Gracias mejores amigos, siempre están cuando los necesito. Esteban eres el hermano con el que más bullying hemos hecho, la amistad subió de nivel y para mí es un privilegio ser tu amigo. Pato, no estés celoso jajaja eres otro de mis mejores amigos, el único con el que he llorado en clase…. a no espera… que me ha hecho llorar jajaja gracias por la paciencia, consejos y apoyo!, Jorge-senpai gracias por ese acolite, esa amistad incondicional y las enseñanzas de todo tipo, esos “black belts” ¡nos esperan!, Luisfe nos ha pasado de todo, literal de todo jaja gracias ñaño, sé que puedo contar contigo a pesar de las situaciones y circunstancias. ¡Los quiero mis jardines!

En la etapa de la universidad conocí mucha gente, pero quiero agradecer por su aprecio, cariño y esa locura única. Gracias amigos por tanta paciencia y por su amistad. Sammy, eres mi mejor amiga te quiero mucho Dtb, Jiss, Galo, Mary, Kathy, Dianita, los quiero mucho, el bullying es lo mejor que pudo pasarnos. Elvis ya… basta no estés celoso jaja sin bullying no hay amistad, Dashi enserio eres genial, única y detergente, gracias por esa locura y sabiduría cósmica. Ricardo M. eres único amigo (has creado un monstruo). Aunque los conocí algo tarde, enserio el bullying es algo que une a la gente, gracias Vane, Jesy, Gabo, Alvarin y todos los chic@s del 6to, nunca es aburrido compartir con Uds, si o que joven.

En la vida no existen las casualidades, y las cosas llegan en el mejor momento, y ¡así es! Aunque hubiese querido pasar más tiempo haciendo música y disfrutando de algo que complementa la vida, debo admitir que el tiempo que he estado en el coro mixto ha sido especial, gracias master Ramiro por el acolite, el apoyo, las bromas, su dedicación… sé que tendrá muchos grandes éxitos y marcará algo en la cultura de la EPN. Niki, Diani, Nico, Carlitos, Sarita, Aleja, gracias por la paciencia, la locura y las anécdotas.

A través de las experiencias y lecciones de la vida he comprendido que siempre hay algo más para aprender y que la vida es una aventura, gracias a todos por su vida y su cariño.

ÍNDICE DE CONTENIDO

RESUMEN

En este proyecto final de carrera se realiza la implementación de un prototipo de comunicaciones inalámbricas empleando la tecnología OFDM, desarrollado en GNU Radio y Radio Definido por Software. Este prototipo implementado es un recurso académico de apoyo para la asignatura de comunicaciones inalámbricas. Debido a que puede mostrar el procesamiento de cada etapa de la comunicación, este prototipo es una herramienta didáctica para el estudio de esta tecnología.

El proceso de implementación del prototipo fue desarrollado por etapas. Primero se implementaron transmisor y receptor OFDM por separado, posteriormente se configuraron los equipos USRP con los parámetros específicos de radio frecuencia. Una vez que los equipos fueron conectados y configurados se estableció el escenario de pruebas para comprobar el funcionamiento del prototipo.

Las pruebas de funcionamiento se realizaron en un ambiente controlado, verificando la compatibilidad en la comunicación de los equipos con las herramientas de GNU Radio, adicionalmente se verificó la funcionalidad de los equipos USRP.

Finalmente, el prototipo entrega un sistema de comunicación inalámbrico que muestra las diferentes etapas a través de los instrumentos de visualización que GNU Radio entrega, en las pruebas de funcionamiento es posible observar el comportamiento del prototipo en un ambiente controlado y en un canal real como la interfaz aire.

PALABRAS CLAVE: OFDM, SDR, comunicaciones inalámbricas, GNU Radio.

ABSTRACT

This project presents the implementation of a wireless communication prototype, which uses OFDM technology and is developed in GNU radio and software defined radio.. This implemented prototype is an academic support resource for the wireless communications subject, since it has the ability to show each stage of the communications process. In this way, the prototype becomes a useful tool for the study of this technology.

The prototype implementation process was developed by stages, first the OFDM transmitter and receiver were implemented separately. After that, the USRP equipment was configured using the specific parameters for radio frequency. Once the devices were connected and configured, a test scenario was established in order to check the functionality of the prototype.

The test run was carried out using a controlled environment in order to test out the compatibility for communications between hardware and GNU Radio software. The functionality of the USRP devices was also checked.

Finally, the GNU Radio software displays each stage of the signal processing by using its visualization tools. During the test run, it is possible to observe the communications system's behavior inside a controlled environment using a real transmission medium such as air interface.

KEYWORDS: OFDM, SDR, GNU Radio, wireless communication.

1. INTRODUCCIÓN

En los últimos años se ha podido observar un crecimiento exponencial de las tecnologías inalámbricas, sobretodo aquellas que emplean OFDM (*Orthogonal Frequency Division Multiplexing*), siendo esta tecnología una de las mayormente utilizadas por varios estándares. Es por esto que resulta importante el estudio de esta tecnología de la manera más clara posible.

OFDM posee ciertas características que la posicionan como una tecnología robusta en comunicaciones, por este motivo es ampliamente empleada. A pesar que esta tecnología tiene variaciones en su capa física dependiendo del estándar que la esté empleando, sus principales características son las mismas y el estudio de esta tecnología es amplio. Por esta razón, una herramienta que permita observar el procesamiento que realiza esta tecnología en cada etapa resulta un recurso académico muy útil para su estudio.

SDR (Software Defined Radio) es una tecnología que en los últimos años se ha utilizado para el estudio de comunicaciones inalámbricas, permite desarrollar sistemas de comunicaciones flexibles, escalables y programables, permitiendo relacionar pruebas de concepto e implementar tecnologías inalámbricas conocidas. De este modo los SDRs resultan un potente escenario de pruebas o “testbed”.

El diseño e implementación de sistemas de comunicación con SDRs permite dar un gran avance en el estudio de las comunicaciones inalámbricas, al usar SDRs como un recurso académico y pedagógico de apoyo los usuarios pueden consolidar y verificar los conocimientos adquiridos teóricamente. También permite experimentar, confirmar y constatar las características de las tecnologías que sean implementados.

En este proyecto se presenta la implementación de un prototipo que permite usar las ventajas de SDR para el estudio de OFDM.

1.1 Objetivos

El objetivo general de este proyecto técnico es implementar un prototipo de sistema de comunicación inalámbrico con tecnología OFDM desarrollado con GNU Radio y SDRs.

Los objetivos específicos de este proyecto técnico son:

- Describir las características fundamentales de OFDM, SDR y GNU Radio.
- Implementar transmisor y receptor OFDM.
- Acoplar transmisor y receptor a los equipos USRP.

- Probar el funcionamiento del prototipo de sistema transmisor-receptor OFDM

1.2 Alcance

El estudio técnico presentará un prototipo de un sistema de comunicación inalámbrico basado en OFDM, mostrando así tanto un transmisor como un receptor conectados por un canal real, implementado con GNU Radio y equipos USRP.

El transmisor y el receptor OFDM estarán formados por los bloques que se observan en la figura 1.1 y 1.2, respectivamente. En el transmisor, la fuente será una señal de audio que será digitalizada (muestreada, cuantificada, codificada) y modulada con QPSK (Quadrature Phase Shift Keying). Luego esta señal pasará por el bloque OFDM de N=48 subportadoras de datos, que está formado por la conversión serie-paralelo, la IFFT (Inverse Fast Fourier Transform) con un tamaño de ventana de 64 y la inserción de prefijo cíclico [1]. Finalmente, esta señal pasará por el bloque de radiofrecuencia y la antena transmisora que será implementado con los equipos USRP. En la figura 1.1 se observa el esquema del transmisor OFDM.

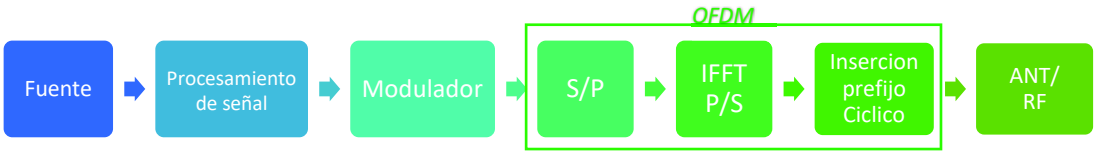

Figura 1. 1 Transmisor OFDM

El receptor realiza el proceso inverso al transmisor, desde la señal que se recibe en el bloque de radiofrecuencia implementado con el equipo USRP, el bloque OFDM. Conformado por conversor serie-paralelo, FFT (Fast Fourier Transform), cabe recalcar que adicionalmente al proceso del receptor se añadirá el bloque de estimación de canal que ayuda a determinar el comportamiento del canal para obtener una demodulación con menor cantidad de errores [2], finalmente se recuperará la señal de audio que fue transmitida. Cabe recalcar que como el Proyecto no se enfoca en estimación de canal y ni el sincronismo de OFDM, no se profundizará las diferentes técnicas que existen tanto para la estimación como la sincronización.

En la figura 1.2 se presenta el esquema del receptor OFDM.

Figura 1. 2. Receptor OFDM

Para probar el funcionamiento del prototipo se genera un escenario que consiste en enviar una señal de audio que será transmitido a través del sistema transmisor-receptor OFDM. En el lado del receptor se debe recibir dicha señal de audio. Se comprobará la compatibilidad del prototipo transmisor-receptor OFDM con los bloques de transmisor y receptor predefinidos por GNU Radio.

Se realizarán pruebas en 3 fases en todas estas se transmitirá audio y se deberá recibir dicha señal de audio. La primera fase consiste en verificar el correcto funcionamiento en simulación de los esquemas transmisor-receptor OFDM implementados con los bloques transmisor y receptor predefinidos por GNU radio. Estas pruebas se realizarán dentro de un mismo ordenador y un canal simulado. La segunda fase consiste en verificar la compatibilidad de la primera etapa, pero ahora además se verificará la conectividad con un equipo USRP. Se verificará la correcta configuración de dicho equipo, en esta fase se utilizará un mismo ordenador y un equipo USRP, así mismo se dispondrá un canal real con una distancia mínima dado que las antenas transmisora y receptora pertenecen al mismo equipo USRP. Finalmente se conectarán dos equipos USRP a dos ordenadores, es decir, un USRP conectado a un ordenador para transmisión y un USRP conectado a otro ordenador para recepción. En esta fase final también se podrá variar la ganancia en el transmisor para observar y escuchar en recepción como se altera la señal.

1.3 Marco Teórico

1.3.1 Sistemas de comunicaciones inalámbricos

Son sistemas que permiten enviar información mediante señales que viajan en una o varias ondas electromagnéticas que se propagan en un medio de transmisión no guiado como es el aire [3].

1.3.2 OFDM

Existen varios esquemas o técnicas de transmisión monoportadoras y multiportadoras, OFDM o multiplexación por división de frecuencias ortogonales es un esquema de transmisión con múltiples portadoras o multiportadora [4]. La principal diferencia respecto

a las técnicas de transmisión monoportadoras radica en el número de portadoras que transportan los datos o información, las técnicas monoportadoras envían la información sobre una sola portadora. Por su parte, OFDM transmite información sobre varias portadoras de datos separadas ortogonalmente.

Esta tecnología comenzó su desarrollo hace varios años, al inicio su implementación era muy compleja pues dependía de un uso de múltiples generadores sinusoidales de subportadoras y por ello un número de demoduladores igual a los generadores. La aplicación práctica de OFDM no era eficiente y a su vez no era viable. Para combatir esto, OFDM experimentó una reformulación teórica basada en la transformada discreta de Fourier o DFT y posteriormente con la implementación de la FFT o transformada rápida de Fourier [5]. El uso de la FFT permite que OFDM lleve a cabo una transmisión por subportadoras paralelas y por tanto combate las interferencias ISI *(interferencia inter-simbolo*) e ICI *(interferencia inter-carrier*).

En la actualidad existen varias tecnologías inalámbricas y estándares como IEEE 802.11, LTE (*Long Term Evolution*), WiMAX *(Worldwide Interoperability for Microwave Access*), ISDB-Tb (*Integrated Services of Digital Broadcasting – Terrestrial*) que emplean OFDM en su capa física debido a sus ventajas sobre la interferencia, modulación adaptable y robustez frente al multitrayecto [6].

En un sistema OFDM existen varios parámetros que se deben analizar tales como: número de subportadoras, tiempo de guarda, prefijo cíclico, duración del símbolo, tipo de modulación por subportadora, sincronismo, ecualización, entre otros.

1.3.2.1 Características de OFDM

OFDM presenta varias características que describen tanto su robustez como su complejidad respecto a otros sistemas, así mismo permite comprender de mejor manera esta tecnología. Algunas de estas características se describen a continuación.

Ortogonalidad y alta eficiencia espectral

Debido a la ortogonalidad de las subportadoras en OFDM, no es necesario dejar bandas de guarda entre las subportadoras. Esto reduce drásticamente el ancho de banda que utiliza respecto a FDM (*Frecuency Division Multiplexing*), siendo así una de sus características principales, puesto que mejoran el uso del espectro radioeléctrico [7]. Así pues, divide un canal de radio en subcanales separados ortogonalmente con el fin de que cada uno puedo transportar una sub-portadora. Esto permite observar al canal de banda ancha como un conjunto de canales paralelos de banda estrecha [5].

En la figura 1.3 se observa la comparación entre el uso del espectro de OFDM con FDM.

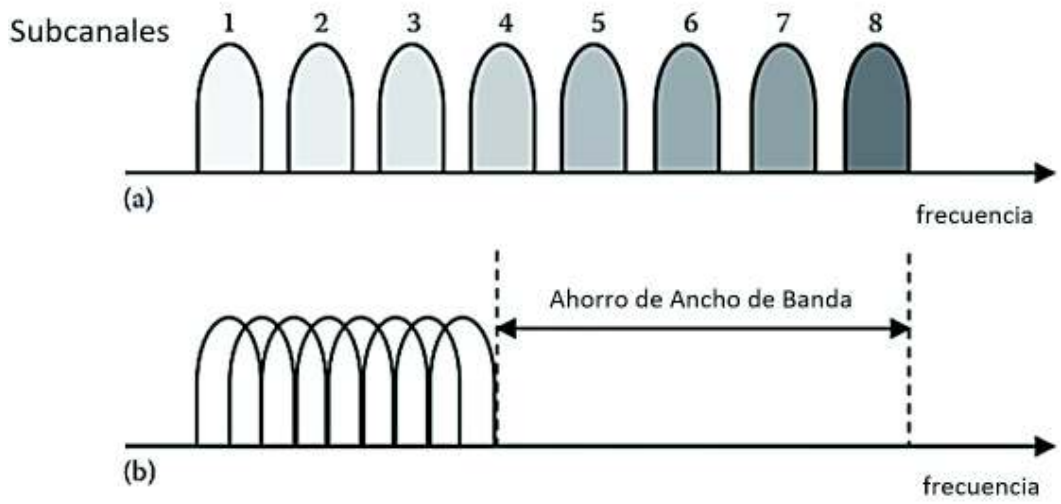

Figura 1. 3 Multiplexación de canales: a) FDM convencional y b) OFDM [8].

Robustez frente al ISI e ICI

El ISI o interferencia inter simbólica se produce por la propagación multitrayecto, este fenómeno se produce cuando una onda viaja o se propaga por varios caminos [4]. Para combatir esta irregularidad OFDM implementa el uso de un prefijo cíclico (el cual se menciona posteriormente).

La ICI (interferencia interportadora o inter-carrier) se ve mitigada por la característica de ortogonalidad del sistema, al mantener la ortogonalidad entre subportadoras OFDM combate este fenómeno. La implementación del prefijo cíclico proporciona una mejora significante en el sincronismo combatiendo tanto ISI como ICI.

Mejora en la implementación de OFDM con el uso del algoritmo de la FFT

Una característica principal de OFDM es el uso de la FFT, de esta manera se consigue una transmisión por medio de subportadoras paralelas, ayudando así a la eliminación de interferencia entre ellas [8].

La transformada de Fourier permite transformar señales del dominio del tiempo al dominio de la frecuencia y viceversa. La transformada continua de Fourier se expresa en la ecuación 1.1, y su inversa en la ecuación 1.2:

$$X(jw) = \int_{-\infty}^{+\infty} x(t)e^{-jwt}\,dt$$

Ecuación 1. 1 Transformada de Fourier [9].

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} x(jw)e^{jwt}\,dt$$

Ecuación 1. 2 Transformada Inversa de Fourier [9].

Con la transformada continua se entregan infinitos valores, cosa que no es posible implementar. Para solucionar este inconveniente surgió como alternativa la implementación de la transformada discreta de Fourier, está permite tomar un número finito de muestras necesarias "N" para reconstruir la señal en frecuencia [10]. Es decir, discretizar la señal descomponiéndola en una suma de secuencias sinusoidales. En las ecuaciones 1.3 y 1.4 se muestran las ecuaciones de la transformada discreta y la transformada discreta inversa de Fourier respectivamente.

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi nk}{N}}, \qquad k = 0,1,\dots,N-1$$

Ecuación 1. 3.DTF Transformada Discreta de Fourier [11].

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi nk}{N}}, \qquad n = 0,1,\dots,N-1$$

Ecuación 1. 4. IDTF Inversa de la Transformada Discreta de Fourier [11].

La DTF e IDFT fueron implementadas en OFDM a través de un conjunto de generadores de señal o también llamados bancos de osciladores y filtros análogos. Esta implementación resultaba costosa, poco factible y tenía un alto costo en procesamiento lo que incurría en una baja eficiencia y relación costo beneficio para su implementación.

El número de cálculos realizados por la DFT está en función del número de muestras N y las multiplicaciones k, la carga computacional asciende en función de N^2, siendo N^2 el número total de procesos o multiplicaciones necesarias. Para reducir drásticamente este procesamiento fue propuesto y posteriormente implementado el algoritmo de la FFT o transformada rápida de Fourier, el número de multiplicaciones se reduce mediante simetría en los cálculos a: $\frac{N}{2}\log_2 N$ multiplicaciones necesarias [12].

En la tabla 1.1 se observa la comparación de los cálculos necesarios entre la DFT y el algoritmo de la FFT

Tabla 1. 1 Comparación cálculos necesarios DFT y FFT [12].

N	N^2; DFT	$\frac{N}{2}\log_2 N$; FFT	Reducción de cálculos de procesamiento
32	1024	80	92%
256	65536	1024	98%
1024	1048576	5120	99.5%

Con el algoritmo de la FFT y la IFFT se reduce notoriamente el número de cálculos necesarios para procesar N muestras y se facilita la implementación. Además, el número de subportadoras utilizadas está directamente ligado al número de muestras que se utilicen en la FFT, la cual se denomina N_{FFT} o simplemente N al número de muestras de la FFT.

Modulación

En otros esquemas monoportadora se necesitaría un alto nivel de modulación para conseguir una alta velocidad de símbolos o gran tasa de transferencia de símbolos para un mismo flujo. OFDM permite la transmisión de una trama digital sobre N líneas paralelas denominadas subportadoras ortogonales contiguas, cada una de estas líneas con un tipo de modulación digital. El nivel de modulación que necesitan las N líneas paralelas de OFDM es menor a los niveles de modulación que necesitan los esquemas monoportadora para conseguir en promedio una gran tasa de transferencia.

En estas sub-portadoras viajan embebidos símbolos independientes productos de alguna modulación digital como BPSK, QPSK ,16-QAM, etc. Estas modulaciones dependen de las necesidades y requerimientos del sistema [13].

La modulación M-PSK se expresa en la ecuación 1.5.

$$s(t) = Acos\left(2\pi f_c t + \frac{2\pi}{n}\right);$$

Ecuación 1. 5 Modulación M-PSK.

Donde:

$\boldsymbol{s(t)}$:Señal resultante

A: Amplitud

$\boldsymbol{f_c}$: Frecuencia de la señal portadora

$\frac{\boldsymbol{2\pi}}{\boldsymbol{n}}$: desplazamiento en fase, depende de los niveles de modulación. $2^n = M$

n : número de bits por muestra

M : niveles de modulación.

En la figura 1.4 se puede observar un ejemplo de la modulación M-PSK, con M=4 también llamada QPSK:

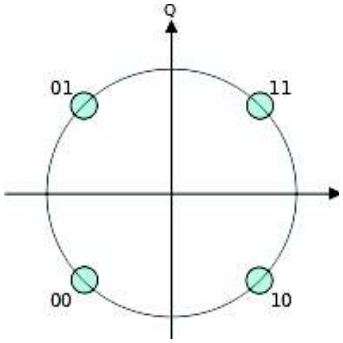

Figura 1. 4 Diagrama de Constelación M-PSK, M=4 llamada QPSK.

Subportadoras

Las subportadoras se originan al dividir el espectro y se asigna un ancho de banda angosto a cada una, no todas las subportadoras transportarán información [14]. Se las puede clasificarlas según sus funciones:

- **Subportadoras de datos**: donde se transmite la información.
- **Subportadoras piloto**: estas subportadoras se encargarán de la sincronización y estimación de canal, el número de subportadoras depende de la tecnología. Por ejemplo, para el estándar IEEE 802.11, OFDM utiliza 4 subportadoras piloto en la capa física.
- **Subportadoras nulas**: también llamadas “null” o subportadoras virtuales, son subportadoras que no transmiten información, su utilidad es de comportarse como bandas de guarda.
- **Subportadora DC:** también llamada subportadora de índice cero o DC, es la portadora central que tiene una frecuencia igual a la frecuencia central de transmisión.

En la figura 1.5 se muestran los tipos de subportadoras.

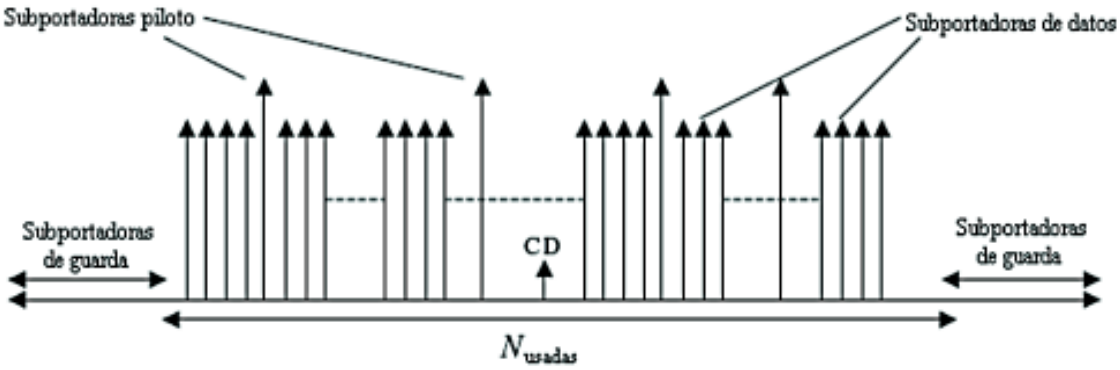

Figura 1. 5. Tipos de Subportadoras en OFDM [17].

Prefijo cíclico y Tiempo de Guarda

El intervalo de guarda o tiempo de guarda es un espacio vacío que se deja entre símbolos OFDM consecutivos, este espacio o periodo tiene como objetivo combatir la interferencia inter-símbolo ISI e interferencia entre portadoras ICI ocasionada por el multitrayecto [4]. Al tener un intervalo de guarda insertado satisfactoriamente entre símbolos OFDM consecutivos se consigue combatir al ISI, sin embargo, no puede enfrentarse completamente con la perdida de la ortogonalidad entre subportadoras.

En la figura 1.6 se muestra la implementación del intervalo de guarda entre símbolos OFDM.

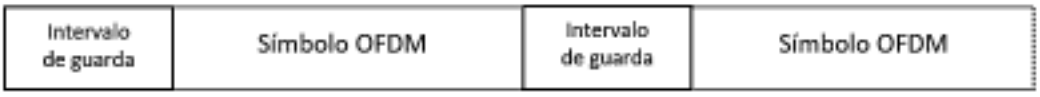

Figura 1. 6 Implementación del Intervalo de Guarda

Para implementar el intervalo de guarda se puede usar un prefijo cíclico (CP), que se define como una copia de la parte final del símbolo OFDM que es agregado al inicio del símbolo, el CP ayuda a preservar la ortogonalidad de las subportadoras y previene el ISI entre símbolos OFDM consecutivos [15]. Se puede observar esta explicación en la figura 1.7.

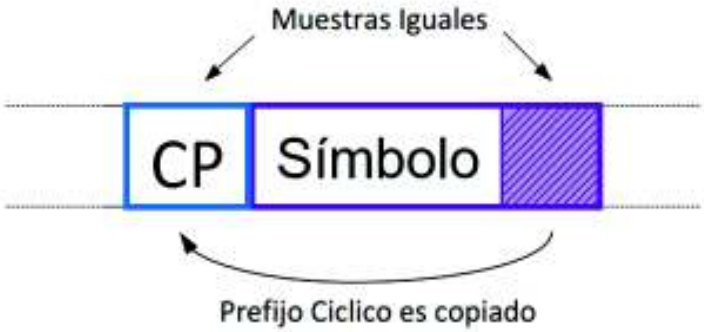

Figura 1. 7 Prefijo Cíclico [13].

La duración del símbolo OFDM es denominado T , y la duración del CP será T_G dado que este será introducido donde antes se ubicaba el intervalo o tiempo de guarda teniendo como máxima duración la cuarta parte de la duración de T. Con esto la duración del nuevo símbolo OFDM incluido el prefijo cíclico será la suma de T y T_G. La duración total del símbolo OFDM se representa en la ecuación 1.6.

$$T_{simbolo\ OFDM\ extendido} = T + T_G$$

Ecuación 1. 6 Duración símbolo OFDM con Prefijo Cíclico

1.3.2.2 Transmisor OFDM

Los elementos principales de un transmisor OFDM se observan en la figura 1.8, siendo los bloques S/P, IFFT e inserción de prefijo cíclico característicos de OFDM.

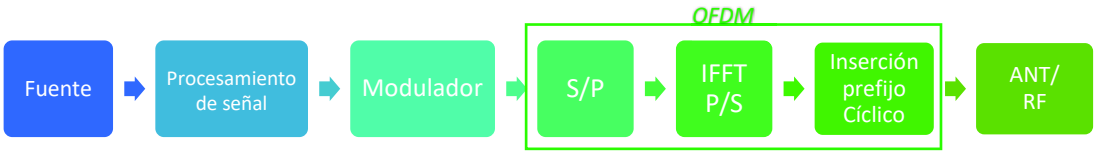

Figura 1. 8 Transmisor OFDM

La fuente puede generar información de diferentes tipos como son: texto, multimedia, señales, entre otras. El bloque procesamiento de señal permite digitalizar la señal que lleva la información, este proceso convierte una señal analógica o señal de tiempo continuo en una señal digital o señal en tiempo discreto facilitando el procesamiento de estas señales. Las partes de la digitalización o procesamiento son: Muestreo, Cuantificación y codificación.

Modulador

A la salida del bloque de procesamiento la señal es modulada dependiendo de los requerimientos del sistema. El modulador se encarga de mapear el flujo que ingresa y

según el esquema de modulación digital con el cual se trabaje, lo convertirá en un flujo de salida con símbolos complejos en serie a una tasa de símbolos por segundo [16]. Los tipos de modulación digital más utilizados por OFDM son: BPSK, QPSK, 16-QAM, 64-QAM, entre otros.

Se debe recalcar que la cabecera de datos OFDM tendrá un tipo de modulación y la carga útil, tendrá otro. Esto se indica con más detalle en la sección de Implementación.

Bloque Conversor serie/paralelo

Hasta este punto los datos viajan en serie, pero en este punto debe cambiar el modo de transmisión a paralelo, este proceso se realiza para poder trabajar con la transformada rápida inversa de Fourier o IFFT. Este bloque es el encargado de convertir el flujo de símbolos en serie que llegan del modulador en un flujo de símbolos en paralelo, haciendo corresponder cada uno de estos símbolos modulados a las subportadoras según las reglas de asignación de portadoras con las cuales se esté trabajando [8].

IFFT (*Inverse Fast Fourier Transform*)

OFDM requiere una señal que sea la sumatoria de señales tales que, en el dominio de la frecuencia mantengan constante las frecuencias adyacentes y la separación entre estas, por esta razón se utiliza la IFFT. La IFFT tomará los flujos paralelos previamente mapeados para convertirlos en muestras en tiempo de una señal, para después de la transmisión, en el lado del receptor sea la FFT la que realice el trabajo de separar los datos entre subportadoras [13].

El principio de la IFFT es tomar un número definido de muestras N_{FFT} en el tiempo y entregar como resultado N_{FFT} muestras en el dominio de la frecuencia [17].

Así mismo, la IFFT del lado del transmisor entregará datos de forma paralela para lo cual obligatoriamente se necesita un bloque que convierta la transmisión a tipo serie nuevamente.

Prefijo Cíclico

Posteriormente se debe introducir un prefijo cíclico (CP) para combatir tanto el ICI como el ISI y completar el símbolo OFDM. Por ejemplo, en varias tecnologías la duración del CP puede ser de ¼, 1/8 hasta 1/32 de la duración del símbolo, sin embargo, nunca la duración del prefijo cíclico deberá ser mayor a ¼ de la duración del símbolo OFDM [18]. Algunos ejemplos se muestran en la tabla 1.2

Tabla 1. 2. Duración del Prefijo Cíclico OFDM en Varios Estándares [19] [20].

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

La información ahora se encuentra lista para ser transmitida por lo que es enviada al bloque de radio frecuencia, de esta manera este bloque se encarga de insertar la información a una portadora para que pueda ser transportada.

1.3.2.3 Receptor OFDM

El receptor OFDM debe hacer el proceso opuesto al transmisor, deberá tomar la señal y procesarla de manera que, pueda recuperar la información y los bits que fueron enviados. El receptor OFDM esta principalmente formado por los 4 bloques que se pueden observar en la figura 1.9.

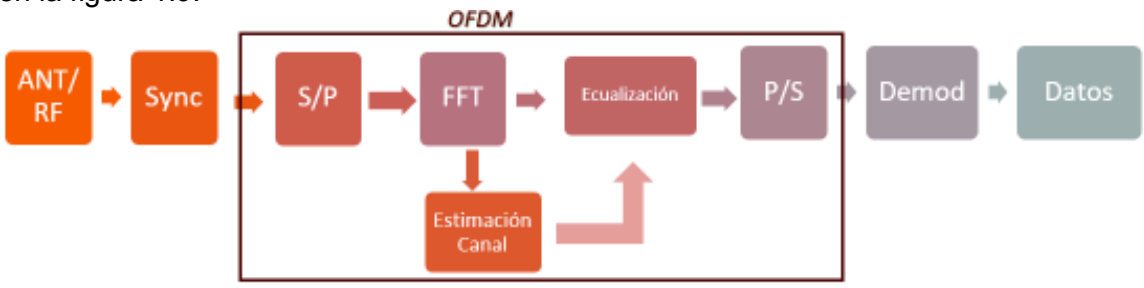

Figura 1. 9 Receptor OFDM

Después de viajar en el canal de transmisión la señal llega al bloque RF el cual entrega una señal en serie embebida que deberá ser procesada, la sincronización y detección de tramas y símbolos OFDM serán realizadas por un bloque de sincronismo.

Sincronismo

Principalmente el/los bloques de sincronización se encargan de discriminar la cabecera, carga útil y señales de sincronismo, también se encargará de ser necesario de recuperar la sincronización tanto en tiempo como en frecuencia. Así mismo este bloque será el encargado de retirar el prefijo cíclico.

Un sistema que pierde la sincronización pierde totalmente su funcionalidad es por ello que existen varias técnicas, algoritmos y estudios para un correcto manejo, control e implementación de las técnicas de sincronismo en OFDM.

Conversión serie-paralelo y bloque FFT

Estos bloques realizarán el mismo proceso que el transmisor, el conversor S/P se encarga de convertir un flujo de símbolos en serie a varios flujos en paralelo con el propósito que puedan ser tratadas e introducidas en el bloque FFT [21]. El bloque FFT realiza el proceso inverso del bloque IFFT, tomará las muestras en tiempo de la señal y las convertirá al dominio de la frecuencia. Entregando así flujos paralelos que el bloque de estimación de canal o ecualización lo procese.

Estimación de Canal y Ecualización

El bloque de estimación de canal en general se encarga de "estimar" o calcular, evaluar, o tasar el comportamiento del canal para poder ecualizar de una manera correcta la señal recibida [22], de esta manera se puede combatir o mitigar la ISI y otros cambios que podrían ser introducidos por el canal [23]. La estimación de canal entrega la información sobre los efectos del canal, por ejemplo: la propagación multitrayecto, desvanecimiento entre otros. Esta información permite saber qué acciones se deberán tomar para reducir y contrarrestar los efectos antes mencionados.

La ecualización compensa los efectos del canal en la señal OFDM, se realiza sobre el dominio de la frecuencia, es decir, cuando la señal ha sido procesada a través del algoritmo de la FFT y previamente a la detección de símbolos. Aunque, se llama ecualización técnicamente es un proceso donde se compensa un valor escalar complejo en cada portadora.

Existen varios métodos de sincronización, estimación y ecualización del canal para OFDM, no obstante, no es el objetivo de este trabajo estudiar de manera detallada los diferentes tipos, técnicas o algoritmos que existen. En el capítulo 2 se indica una breve descripción sobre el tipo de sincronismo, estimación de canal y ecualización que se emplea en el proyecto.

Posterior al bloque de estimación de canal y ecualización los flujos de información deben ser convertidos a un solo flujo en serie, por ello nuevamente se emplea un conversor P/S que entregara un flujo de bits al demodulador.

Demodulación

El demodulador es el encargado de recibir el flujo de símbolos que entrega el conversor P/S, posteriormente los interpreta de manera que puedan ser recuperados dependiendo del tipo de modulación que se haya utilizado. Finalmente, después de interpretar el flujo de símbolos se espera que la información sea recuperada.

A continuación, se describe los aspectos más relevantes de la tecnología SDR (Radio definida por software).

1.3.3 SDR

La Radio Definida por Software o SDR por sus siglas en ingles *Software Defined Radio* nace a partir de la idea de diseñar equipos que puedan ser reconfigurables de una manera sencilla, de esta manera se tendrán componentes como: mezcladores, filtros, sumadores, generadores de señal, moduladores/demoduladores, detectores, etc. Que generalmente se implementan en hardware, pero ahora Implementados mediante software [24].

Existen diferentes tipos de equipos y se tiene problemas de compatibilidad entre estos. SDR fue ideado para combatir problemas tanto de compatibilidad como de operabilidad. También define un conjunto de procesos y técnicas orientadas a realizar el procesamiento de señales de radio por medio de un dispositivo de propósito general [25]. Dicho dispositivo puede ser modificado por medio de software logrando así un cambio dinámico, automático y eficiente entre tecnologías sin tener que incurrir en costes, de ahí surge la necesidad de diseñar un dispositivo capaz de entregar en banda base, paso banda, banda ancha, la señal de radio para su procesamiento digital, incluso si la señal es analógica [26].

En la figura 1.10 se puede observar una descripción simplificada de un sistema SDR.

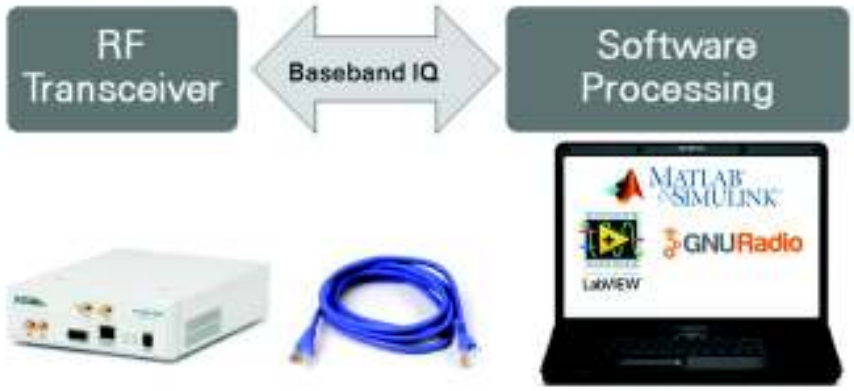

Figura 1. 10 Descripción simplificada de una configuración de SDR construida alrededor de un NI USRP.

El dispositivo debe ser configurable y debe tener la posibilidad de recibir distintos anchos de banda y frecuencias para posteriormente con procesamiento digital poder realizar la sincronización y la detección de la señal recibida. Estas características de reconfiguración y manipulación permiten que los equipos SDR puedan reemplazar fácilmente a muchos tipos de equipos de radio frecuencia en hardware. Además, al poder usar los equipos SDR

para crear, implementar o diseñar estándares de comunicaciones inalámbricas es posible realizar estudios, análisis e investigaciones [27].

A continuación, se describen varias ventajas de la tecnología SDR.

Ventajas SDR

- **Múltiples modos de operación**: una SDR puede trabajar en varios modos según la aplicación o uso que se quiera darle. Por ejemplo: AM, FM, SSB, QPSK, OFDM, TDMA, etc. [28].
- **Reconfiguración:** las SDR pueden estar trabajando en un modo de operación, sin embargo, es posible configurarla para que trabaje en otro tipo de operación o tecnología de comunicación, esto dependerá de la configuración o programación del software.
- **Actualización Over-The Air:** ciertos parámetros pueden variar e irse calibrando cuando el equipo este encendido, es decir, cuando el equipo esté en funcionamiento o la comunicación entablada.
- **Menor costo de desarrollo:** de manera didáctica trabajar con SDR reduce los costos de adquisición de muchos equipos de hardware para aplicaciones específicas, además se puede realizar mayor cantidad de aplicaciones.

Aunque hoy en día las SDR son muy utilizadas aún no está muy claro si son una tecnología, un estándar o un sistema. Por esta razón estos equipos tienen muchas limitantes. Se podría decir que son un conjunto de tecnologías, tanto de software, hardware para procesamiento de señales, tecnología de RF, que trabajando en conjunto hacen realidad al concepto de sistema SDR [29]. Así mismo, tampoco se encuentran estandarizadas, razón por lo cual no pueden ser aprovechadas a su totalidad.

En la actualidad hay muchas iniciativas de programas, comunidades y empresas que desarrollan entornos en los cuales se puede trabajar con SDR, existe tanto software licenciado como software libre [30]. A continuación, se muestran varias herramientas de software que trabajan en SDR.

- Joint Tactical Radio System
- SDR Forum
- GNU Radio
- Simulink Matlab (instalación de varias librerías, drivers y plugins)
- LabVIEW

- Secure SDR Multilateral Information Exchange Agreement (SSDR MIEA)
- European Defense Agency: WINTSEC (*Wireless INTeroperability for SECurity),* ESSOR *(European Secured Software Defined Radio Referential*)

En estas plataformas, programas, aplicaciones o entornos de desarrollo se puede trabajar en SDR, para ello se puede usar equipos comerciales que se encuentran disponibles en la actualidad. Unos cuantos ejemplos se muestran en la tabla1.3.

Tabla 1. 3 Ejemplos Tipos de Equipos SDR [27].

<table>
<tr><th>Uso Amateur</th><th>Uso Profesional</th><th>Uso Táctico</th></tr>
<tr><td>FlexRadio:
Flex-6400, Flex-6600, Flex-6700</td><td rowspan="3">Spectrum</td><td>Rohde & Schwarz</td></tr>
<tr><td>USRP:
USRP 2920, USRP 2910, USRP n210</td><td>Harris:
PRC-117G, RF-7800t-HH</td></tr>
<tr><td></td><td>Thales (Europa)</td></tr>
</table>

Arquitectura Equipos SDR

Todas las SDR tienen muchas similitudes en sus arquitecturas, pero principalmente difieren en las capacidades de ciertos componentes específicos según sus modelos y fabricantes, sin embargo, sus principales elementos se mencionan a continuación:

Entorno de Desarrollo: es el software en el cual se podrá crear las aplicaciones o programas para uso específico del usuario.

El código fuente: es el programa o aplicación que se desarrolló sobre el entorno de desarrollo o mediante programación, dependerá del software que se esté utilizando.

Procesamiento: Esta etapa contiene procesadores de señal o en varios equipos incluye una tarjeta madre o *motherboard* y una o varias *daughterboards* para el procesamiento. En algunos equipos la *motherboard* es una FPGA que puede ser reconfigurable a manera que el usuario pueda manipularla libremente, pero también hay equipos que no permiten esta reconfiguración sino solamente actualizaciones del firmware por parte de los distribuidores [31].

- **Motherboard:** llamada placa madre o placa base es la parte fundamental de un sistema electrónico, es la encargada del procesamiento a través la unidad de procesamiento central (CPU) también llamado microprocesador o núcleo. Muchos de los componentes cruciales del sistema, la memoria y los conectores para dispositivos de entrada y salida están incluidos en esta placa [32].

- **Daughterboard:** llamada placa hija, placa secundaria o tarjeta de expansión, es un tipo de placa de circuitos que se conecta o se conecta a la placa base o motherboard para ampliar sus funciones y servicios. Una placa secundaria complementa la funcionalidad existente de la placa base [32].

ADC/DAC: son los convertidores analógicos digitales y viceversa incorporados en los equipos, comunican las etapas de software y *front-end*. Por lo general este bloque representa la limitación de los equipos al momento de convertir los datos, la capacidad de este bloque viene dada desde su fabricación y no puede ser manipulado por el usuario.

Front End: representa a la etapa de radio frecuencia de la comunicación, esta etapa es la que principalmente el usuario manipula y establece parámetros de frecuencia tanto de transmisión como recepción dependiendo de los equipos [31].

En la figura1.11 se observa la distribución típica de los elementos principales de SDRs.

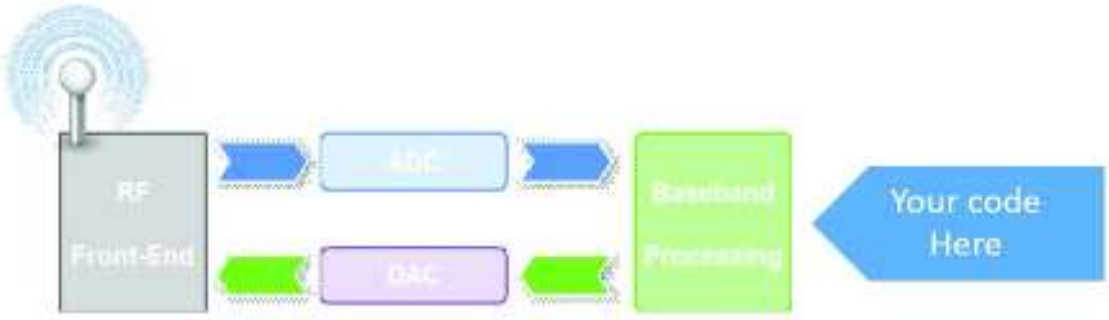

Figura 1. 11 Diagrama de bloques típico de los SDR [7].

1.3.4 USRP Universal Software Radio Peripheral

USRP de sus siglas en ingles Universal Software Radio Peripheral son equipos SDR desarrollados por la compañía Ettus Research1. Los equipos USRP son los mayormente preferidos para el desarrollo de algoritmos, creación de prototipos y desarrollo de tecnologías inalámbricas, porque presentan la facilidad de realizar diferentes aplicaciones debido a sus ventajas y características permitiendo así un alto rendimiento en el hardware de RF y una facilidad de expansión [33].

La mayoría de estos equipos al igual que muchos otros se conectan mediante enlaces de alta velocidad como USB 2.0-3.0 o gigabit Ethernet a un ordenador o host con software

[1] Ettus Research ™, una marca de National Instruments (NI) desde 2010, es el proveedor mundial líder de plataformas de radio definidas por software, incluida la familia de productos Universal Software Radio Peripheral (USRP ™).

basado o que soporte la tecnología USRP. En la figura1.12 se muestra la arquitectura de estos equipos.

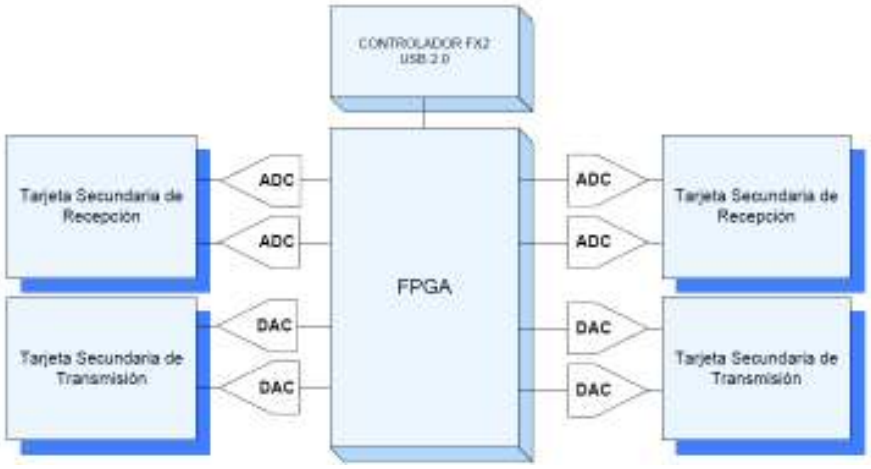

Figura 1. 12. Arquitectura General de la Tarjeta Madre de los USRP [33].

En general la arquitectura de los equipos USRP incluye una motherboard interconectada con otras daughterboard encargadas de trasmisión y recepción respectivamente. Hay versiones de equipos USRP con una FPGA configurable como también USRP que no tienen esta opción [33].

1.3.4.1 USRP NI 2920

El modelo NI 2920 es un equipo SDR que permite tanto la transmisión como la recepción de datos, posee una interfaz Gb Ethernet para comunicarse con el ordenador, tiene una tarjeta FPGA no reconfigurable por el usuario, sino dependiente del Firmware del fabricante y controlada mediante drivers UHD (USRP Hardware Driver) [33].

Las características del USRP NI 2920 se presentan en la tabla 1.4.

Tabla 1. 4 Características del USRP 2920 para transmisión y recepción [33].

Transmisión	Recepción
Rango de frecuencia 50 MHz - 2.2 GHz	Rango de frecuencia 50 MHz - 2.2 GHz
Pasos de frecuencia <1 kHz	Pasos de frecuencia <1 kHz
Máxima Potencia de salida (Pout): 50 MHz - 1.2 GHz: 50 mW - 100 mW (17 dBm to 20 dBm) 1.2 GHz - 2.2 GHz: 30 mW - 70 mW (15 dBm - 18 dBm)	Rango de ganancia 5 0 dB - 31.5 dB
Rango de ganancia 1 0 dB to 31 dB	Pasos de Ganancia 0.5 dB
Pasos de Ganancia 1.0 dB	Potencia máxima de entrada (Pin) 0 dBm

Precisión de frecuencia 2.5 ppm	Figura de Ruido 5 dB to 7 dB
Ancho de banda máximo instantáneo en tiempo real: 16-bit ancho de muestra 20 MHz 8-bit ancho de muestra 40 MHz	Precisión de Frecuencia 2.5 ppm
Velocidad máxima de muestreo I / Q 16-bit ancho de muestra 25 MS/s 8-bit ancho de muestra 50 MS/s	Ancho de banda máximo instantáneo en tiempo real:: 16-bit ancho de muestra 20 MHz 8-bit ancho de muestra 40 MHz
Convertidor digital a analógico (DAC) 2 canales, 400 MS/s, 16 bit	Velocidad máxima de muestreo I/Q 16-bit ancho de muestra 25 MS/s 8-bit ancho de muestra 50 MS/s
Rango dinámico sin alteraciones DAC (sFDR) 80 dB	Convertidor analógico a digital (ADC) 2 canales, 100 MS / s, 14 bits
	ADC sFDR 88 dB

El USRP 2920 es compatible con muchos de los programas de entorno mencionados anteriormente en la sección 1.3.3, pero se puede especificar que por lo general se utilizan en: Simulink de Matlab, GNU Radio y LabVIEW. Estas tres aplicaciones tienen drivers o controladores más actualizados, esto permite operar los equipos desde su entorno, además, tienen soporte tanto por Ettus Research como por National Instruments.

La arquitectura del USRP 2920 se observa en la figura 1.13. desde la interfaz Gbit Ethernet, hasta las antenas de transmisión y recepción RX1/TX1 Y RX2.

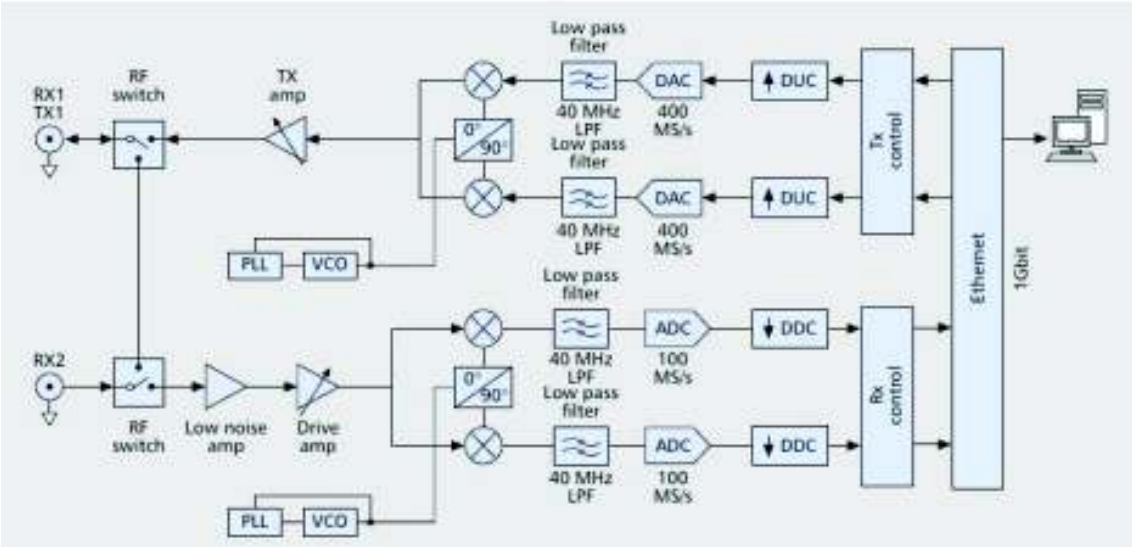

Figura 1. 13 Arquitectura USRP NI 2920 [33].

En la figura 1.14 podemos observar las principales interfaces del equipo USRP 2920.

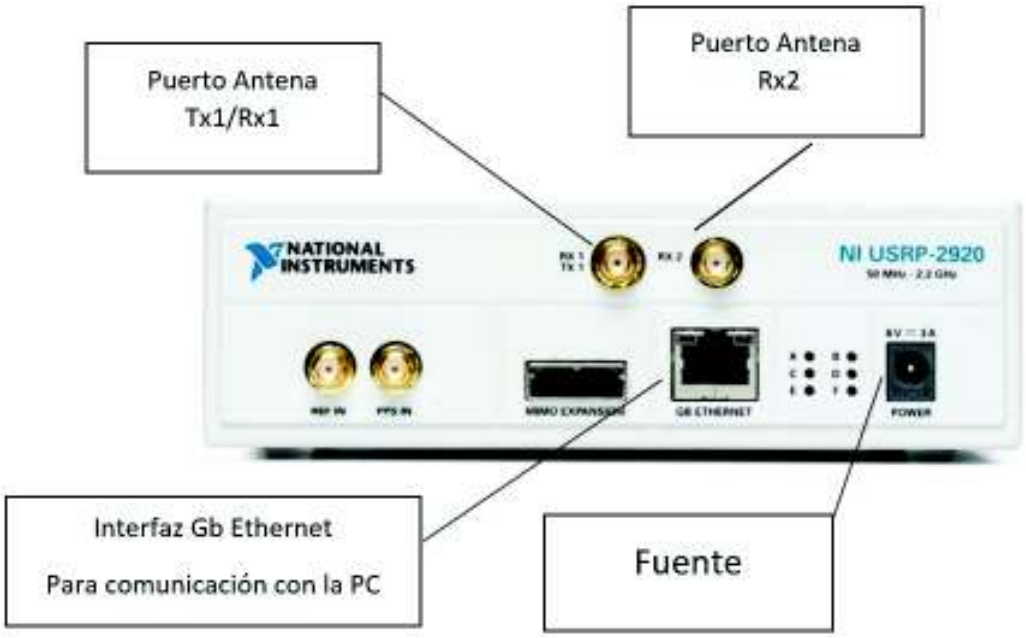

Figura 1. 14 Panel Frontal NI USRP-2920.

1.3.5 GNU Radio

GNU Radio es un kit de herramientas para el desarrollo de software libre y open-source, es decir, con código abierto. Proporciona bloques de procesamiento de señales para implementar aplicaciones de radiofrecuencia mediante software. Puede ser utilizado con hardware de bajo costo para implementar un ambiente o entorno de pruebas para realizar simulaciones [34]. Ademas, se encarga de realizar el DSP (Digital Signal Processing) o procesamiento digital de la señal. Este posee elementos como filtros, moduladores, codificadores, sincronizadores, vocoders, entre otros bloques. [29].

GNU Radio es utilizado ampliamente en: la industria, el gobierno, la educación e investigación, por usuarios aficionados a las comunicaciones inalámbricas y sistemas de radiofrecuencia. En fin, en GNU se puede simular, crear, implementar prototipos, todo desde el mismo espacio de trabajo.

1.3.5.1 Características GNU Radio

Compatibilidad entre sistemas operativos.

GNU radio es una plataforma o entorno de desarrollo con pocos requerimientos por tal motivo puede trabajar sobre varias distribuciones de sistemas operativos, además existe compatibilidad entre los archivos contenedores que se generan de tal manera que se pueda usar en cualquiera de las plataformas como pueden ser:

- Windows

- Linux: Ubuntu, CentOs, Fedora, Raspbian
- MacOS

En la figura 1.15 se observa la compatibilidad de GNU Radio al ejecutarse sobre una distribución de MacOS.

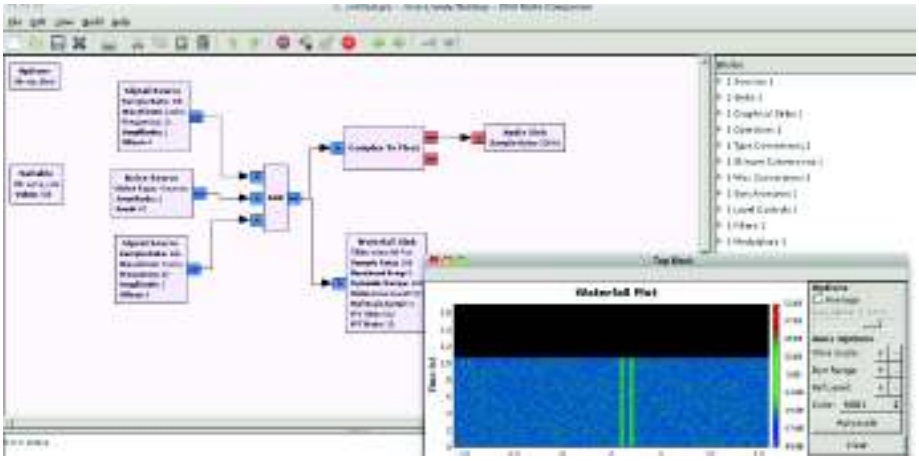

Figura 1. 15. GNU Radio ejecutándose sobre MacOS.

Programación Por Bloques

Una de las características más poderosas de GNU Radio es su interfaz de usuario el cual es amigable, trabaja con un método de programación por medio de conexión de bloques y gestiona el funcionamiento de cada bloque. El usuario es el encargado de conectar los bloques dependiendo de lo que este construyendo. Además, tendrá que prestar atención tanto en las entradas como las salidas de los bloques al momento de conectar, pues tanto las entradas como salidas dependerán del tipo de variable que se esté utilizando.

En la figura 1.16 se observa un bloque de GNU radio, con su entrada y salida.

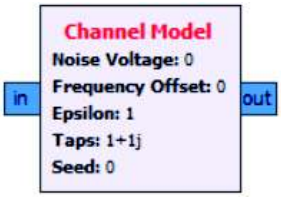

Figura 1. 16 Bloque Modelo de canal en GNU Radio.

Creación de Bloques propios

GNU Radio presenta muchas herramientas entre sus elementos aún así cuando un bloque especifico no se encuentra se lo puede crear y agregarlo a la librería. Por lo tanto, ampliar la librería de GNU Radio no tiene mucha complejidad.

Un ejemplo es el bloque presentado en la figura 1.17, que representa un bloque conversor de variables tipo caracteres a complejos.

Figura 1. 17 Bloque Conversor de variable caracteres a complejos [24].

Este bloque fue desarrollado para minimizar el proceso de conversión que comúnmente primero convierte los caracteres a variables del tipo flotante y posteriormente de flotantes a complejos como se observa en la figura 1.18. dicho proceso se simplifica con la implementación única del bloque b_char2complex.

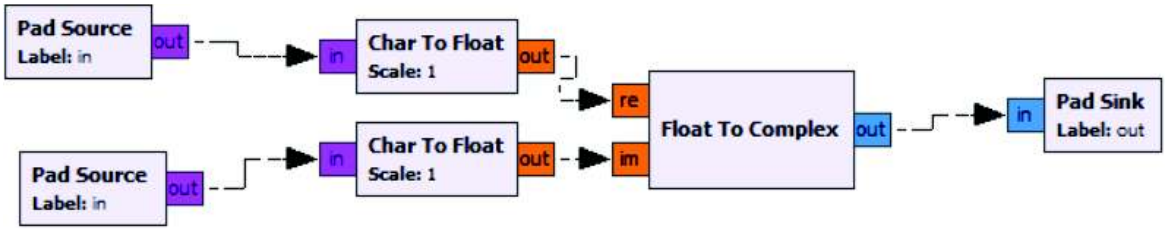

Figura 1. 18 Esquema de bloques convertir variables del tipo caracteres a complejo [24].

Lenguajes de programación

Es verdad que GNU Radio presenta una interfaz basada en programación por bloques, sin embargo, también es posible desarrollar aplicaciones o herramientas mediante la programación de códigos basados en lenguajes de programación como lo son C++ y Python. GNU Radio está desarrollado en Python, pero también en C++ por lo tanto sus aplicaciones también pueden ser desarrolladas en estos dos lenguajes. Esto permite que los usuarios puedan implementar sistemas de radiofrecuencia con alto rendimiento en tiempo real sobre un entorno de desarrollo de aplicaciones fáciles de usar y rápidas de implementar [34].

Además, después de implementar cualquier aplicación usando el entorno de programación por bloques es posible editar el código fuente con programas que permitan manipularlos, un claro ejemplo es Notepad++ en Windows, vim, vi, gedit en Linux o “Komodo” y “Bean” para MacOS.

1.3.5.2 Interfaz GNU Radio

GNU Radio presenta una interfaz que está conformada por: área o espacio de trabajo, barra de herramientas, biblioteca o librería, barra de búsqueda, ventana de variables y el panel de consola [34].

La figura 1.19 muestra la interfaz de usuario con sus elementos.

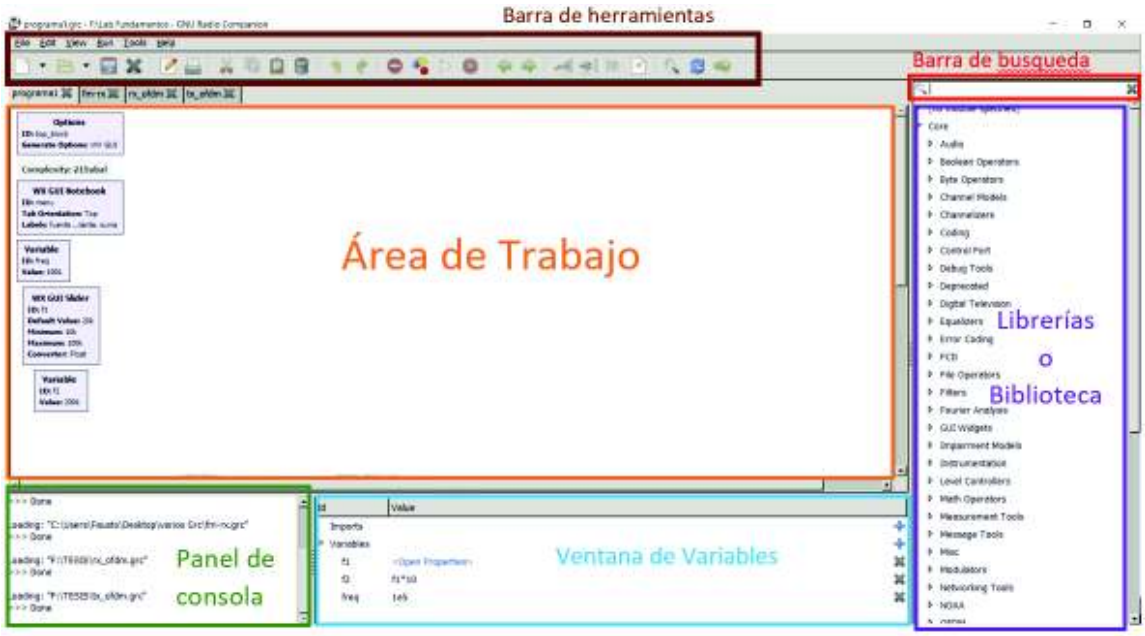

Figura 1. 19 Interfaz gráfica de GNU Radio y sus componentes

Barra de herramientas: presenta las diferentes opciones e instrumentos que se pueden utilizar en el desarrollo de aplicaciones.

Panel de consola: en este espacio se observa con detalle los mensajes que el programa envía, también funciona como depurador al momento de revisar los mensajes de error cuando un programa o aplicación no funciona.

Ventana de variables: en este espacio se pueden observar las variables que la aplicación este utilizando y también permite crear nuevas de ser necesario.

Librería: también llamada biblioteca es el espacio donde se pueden encontrar todos los bloques de herramientas que GNU Radio presenta. También se puede almacenar los bloques propios desarrollados para usarlos como herramientas comunes.

Área de trabajo: en esta área se conectan los bloques y de desarrollan las aplicaciones.

Por defecto al iniciar la aplicación aparecerá un bloque dentro del espacio de trabajo, este se llama *Options*.

El bloque *Options* permite asignar un título a la aplicación, poner una descripción de lo que hará, el autor, y también asignar el tamaño del área de trabajo. Es posible seleccionar el tipo de opciones que se generarán y utilizarán dentro de la aplicación. Para el desarrollo

de aplicaciones se tiene las siguientes opciones: WX GUI y QT GUI y para la creación de bloques propios: Hier block.

También se pueden observar varias pestañas que explican la documentación y una descripción más detallada del bloque, además de mostrar ciertos parámetros, del código de dicho bloque. El bloque "*options*" y sus propiedades se presenta en la figura 1.20.

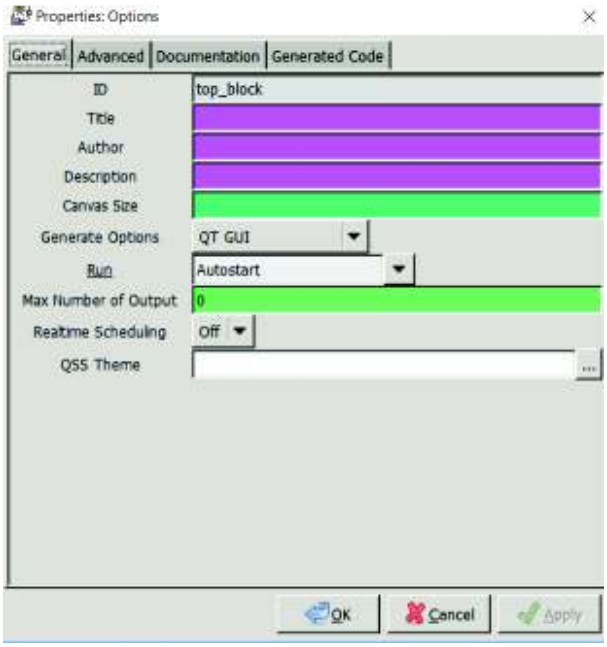

Figura 1. 20 Propiedades Bloque Options

GNU Radio puede trabajar con varios tipos de variables según los requerimientos que el sistema o aplicación necesite. Pueden ser: enteros, flotantes, complejos, caracteres, bits, entre otros. Las variables se las puede identificar por colores en la entrada y salida de los bloques, e incluso se puede asignar el tipo de variables en las cuales el bloque deberá trabajar.

En la figura 1.21 se observa los colores de las entradas y salidas de los bloques representan el tipo de datos que ingresan, por ejemplo, en el bloque "NBFM Transmit" que representa un transmisor FM de banda estrecha ingresan valores del tipo flotante y salen valores complejos, mientras que en el bloque "PSK Mod" que representa un modulador PSK ingresan valores tipo byte y después de procesarlos se entregan valores complejos.

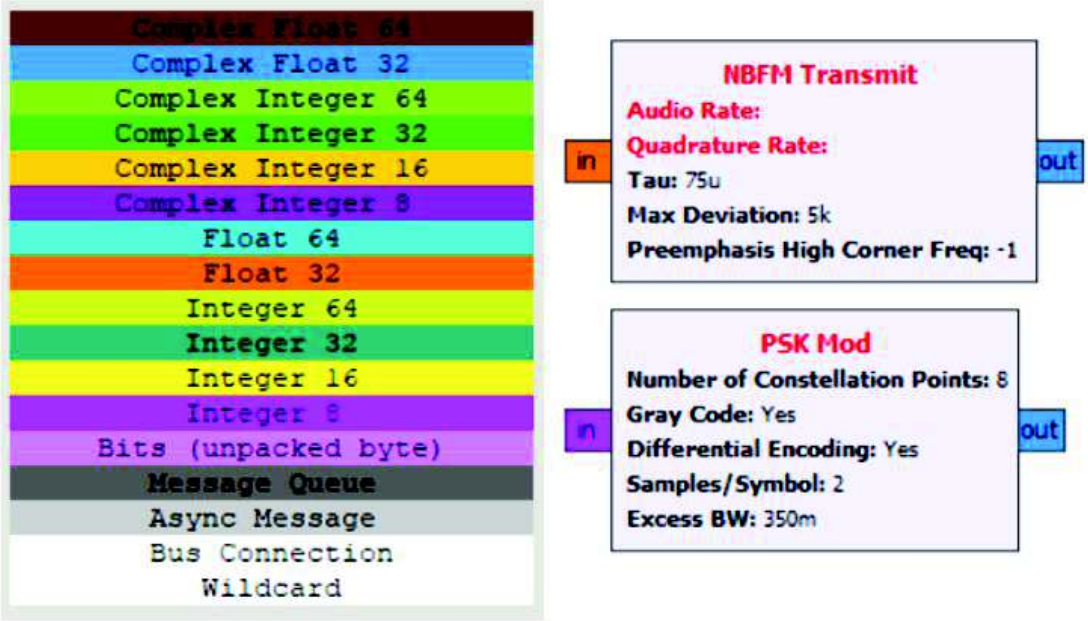

Figura 1. 21 Tipos de variables y ejemplos de bloques

Herramientas de Análisis

En GNU Radio se encuentran instrumentos que permiten hacer un análisis a las señales transmitidas y recibidas, son bloques que permiten observar el comportamiento de las señales para poder interpretar los fenómenos que existen o puedan presentarse en las aplicaciones o programas generados. Se dividen principalmente en dos grupos, herramientas WX y QT, las cuales dependen de las variables que se están utilizando.

A continuación, se describen varias herramientas WX y QT.

Herramientas WX GUI

Estas herramientas están basadas en lenguaje C++ podrán trabajar con WX options y no son compatibles con herramientas QT GUI. Estas herramientas consumen más recursos en procesamiento en comparación que herramientas QT GUI [28]. Los bloques que se pueden encontrar son:

- **WX GUI Scope Sink:** permite observar el comportamiento de las señales en el dominio del tiempo, su funcionamiento es similar al de un osciloscopio.
- **WX GUI FFT Sink:** permite observar el dominio de la frecuencia de las señales, es decir, se observa el espectro de frecuencia de la o las señales que se analicen.
- **WX GUI Constellation Sink:** es la representación gráfica de las señales complejas, es útil para el estudio y visualización de modulaciones digitales como: M-PSK, QAM, FSK, ASK, etc.
- **WX GUI Histo Sink:** esta herramienta presenta un histograma.

- **WX GUI Number Sink:** este instrumento entrega en tiempo real el valor numérico recibido transmitido de la señal, dependiendo del tipo de variable utilizado pueden ser: real, compleja, entera.
- **WX GUI Waterfall Sink:** permite observar en que rango de frecuencia se transmite alguna señal. Esta herramienta entrega un gráfico similar a los mapas de calor.

Herramientas QT GUI

Este kit de herramientas está basado en Python, consumen menos recursos en procesamiento, además, permiten el desarrollo de más herramientas visuales por parte de los usuarios. Estas herramientas están predominando en GNU Radio y se espera que en próximas versiones estas sean las únicas herramientas para análisis e interpretación de fenómenos [34]. Presentan más herramientas para visualización que WX Options, sin embargo, solamente se indican de las mayormente usadas:

- **QT GUI Constellation Sink:** esta herramienta es la equivalente a Wx Constellation sink, permite observar las constelaciones de una señal modulada digitalmente.
- **QT GUI Frequency Sink:** herramienta equivalente a la WX FFT sink, esta herramienta permite variar el número de muestras de la FFT, para observar con mayor o menor precisión la gráfica, esto implica un incremento o disminución en el procesamiento de la aplicación o programa.
- **QT GUI History Sink:** es el equivalente a WX Histo, entrega un histograma, sin embargo, es posible agregar más de una entrada para poder comparar señales si es necesario.
- **QT GUI Time Sink:** herramienta equivalente al WX Scope, permite ver el comportamiento de la señal en el dominio del tiempo. Además, se puede agregar varias entradas de esa manera se puede realizar una mejor comparación entre señales.
- **QT GUI Waterfall Sink:** el diagrama de cascada entrega una gráfica similar a un mapa de calor, muestra las frecuencias en las que exista mayor concentración de potencia en el espectro de frecuencias.
- **QT GUI Sink:** esta herramienta permite observar las 4 herramientas en una sola ventana con varias pestañas en la interfaz del usuario. Disminuye el procesamiento pues cada herramienta que no se utiliza se pone en pause mientras se utilice alguna.

2 METODOLOGÍA

En este capítulo se describe la configuración e implementación del sistema, los prototipos del transmisor y receptor OFDM, comunicación con los equipos USRP y configuración de estos equipos. El proceso de diseño e implementación se divide en varias etapas como se muestra en la figura 2.1.

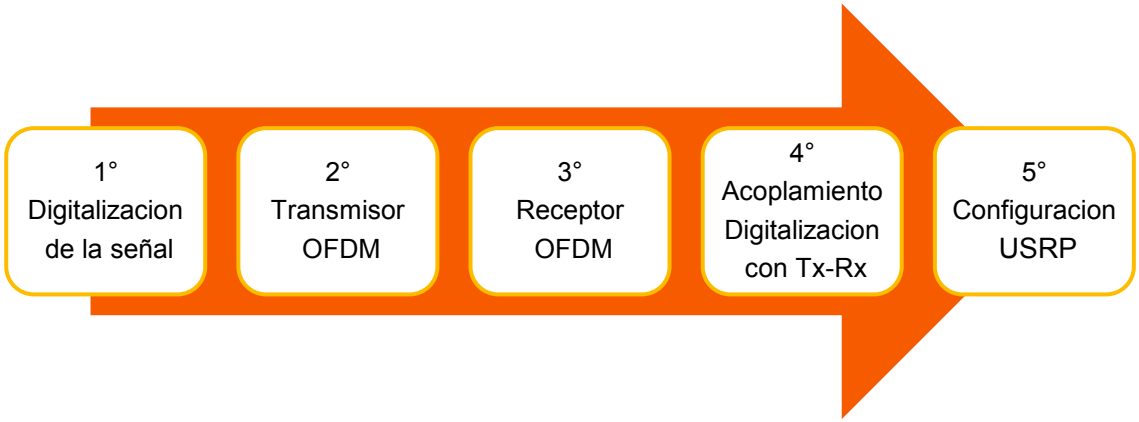

Figura 2. 1 Proceso de Implementación sistema Tx - Rx OFDM

Las primeras 3 etapas se desarrollan en diferentes áreas de trabajo, es decir, generan 3 ficheros "*.grc*" que posteriormente se unirán en un mismo archivo. Esto se realiza para trabajar simultáneamente en las etapas sin crear una dependencia entre estas. El proceso de unión o acoplamiento de estos 3 archivos se realiza configurando las variables comunes, en la cuarta etapa se realizará este acoplamiento.

El prototipo presenta una interfaz donde se observa el comportamiento de la señal en las etapas del sistema, esta interfaz también se incorpora en la etapa de acoplamiento. Finalmente, se realiza la configuración del equipo USRP con los archivos generados, en esta etapa también se especifican los valores que el equipo empleará para la transmisión y recepción. A continuación, se describen cada una de estas etapas.

2.1 Digitalización de la señal

Esta etapa procesa la información que se transmite, por ejemplo: un archivo de audio, audio en tiempo real, una señal sinusoidal o señal aleatoria, etc. Para ello se deberá escoger el tipo de bloque específico para cada tipo de información. Cada uno de estos diferentes tipos de archivos o información que ingresa al sistema lo hace de manera distinta. Por ello debe especificar y adecuar el escenario. El proceso se explica en la figura 2.2.

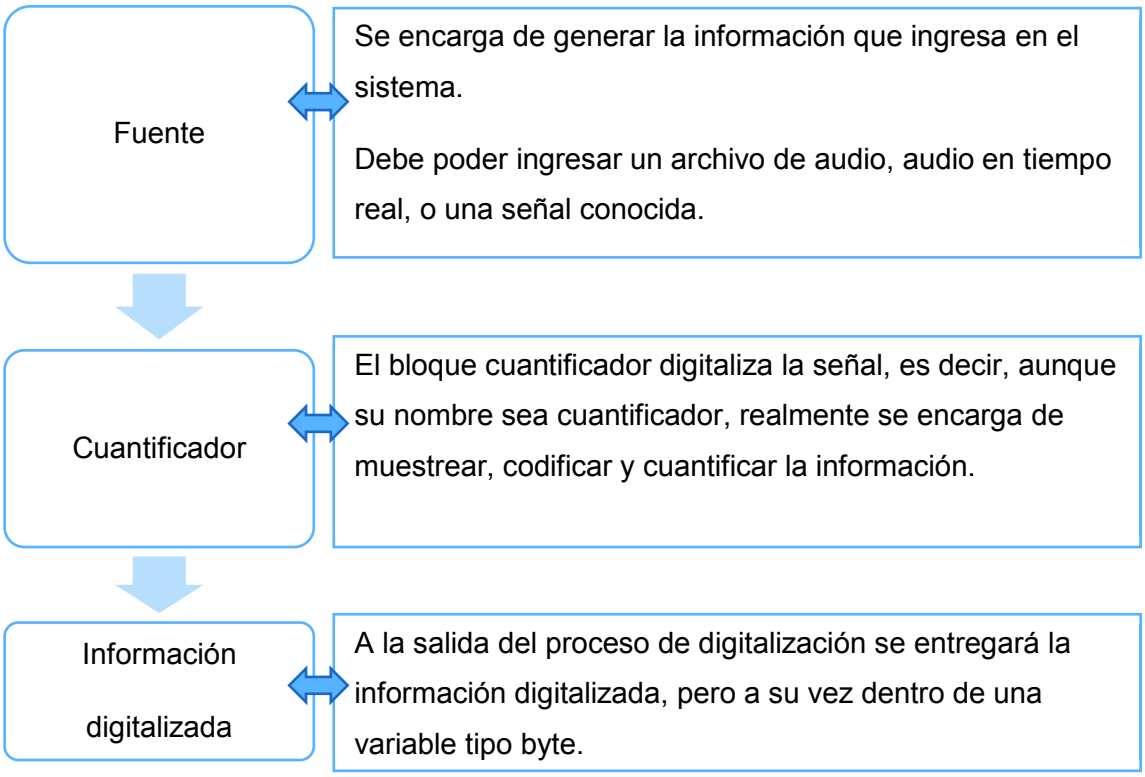

Figura 2. 2 Procesos de la etapa de digitalización de la señal

GNU Radio automáticamente genera una variable llamada "*Sample Rate*", esta variable será la frecuencia de muestreo con la que la aplicación trabaja, sin embargo, es posible variarla según los parámetros que se necesiten, cabe recalcar que se debe cumplir el criterio de Nyquist para no tener inconvenientes. Nyquist sugiere que la $f_{muestreo} \geq 2f_{maxima\_signal}$, es decir, la frecuencia de muestreo necesaria para reconstruir una señal debe ser por lo menos 2 veces mayor que la frecuencia máxima de la señal que se quiere reconstruir o el ancho de banda de la señal [35].

Por defecto GNU radio asigna a la frecuencia de muestreo un valor de 32KHz, pero se debe tomar en cuenta que a esta frecuencia el programa funciona, ejecuta los bloques y muestrea las señales de las fuentes que ingresen información, también será la frecuencia a la que los instrumentos de medición trabajarán. Dado que esta frecuencia es muy baja para trabajar con los archivos de audio (por lo general estos tienen una frecuencia de 44.1KHZ, o 48KHz) es necesario cambiar este valor. El valor con el que se trabajará será 441KHz, este valor es 10 veces mayor que la frecuencia de la señal, cumpliendo el criterio de Nyquist y siendo un valor adecuado para todas las fuentes con las cuales el prototipo trabaje, esto se observa en la figura 2.3.

Figura 2.3 Variable *samp_rate* o frecuencia de muestreo

Los bloques, y demás instrumentos en GNU Radio por defecto tienen asociada la variable "*samp_rate*" como su frecuencia de muestreo, sin embargo, es posible cambiar ese valor. No obstante, para optimizar la programación es preferible asociar la mayor cantidad de bloques a la misma variable y solamente cambiar los bloques que necesariamente deban usar un valor diferente como frecuencia de muestreo.

Para ingresar los datos se utilizan varios bloques que actúan como fuentes, algunos generan señales, otros ingresan la información asociando a archivos o ficheros. Por esta razón estos bloques se conocen como fuentes, los diferentes tipos de fuentes en GNU Radio se las encuentra en la biblioteca, pero las que se utilizarán en este proyecto son:

- **Signal Source:** genera señales conocidas como: sinusoidales, cuadradas, triangulares y dientes de sierra. Este bloque dependerá de un valor de frecuencia de la señal, amplitud y una frecuencia de muestreo que permita que la señal sea muestreada para ser representada en el programa. A la salida de este bloque se puede entregar una señal representada en diferentes tipos de variables como son: float, complex, int y short. Esto se observa en el bloque en la figura 2.4.

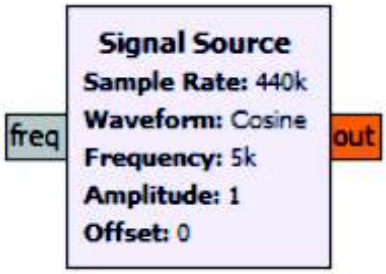

Figura 2.4 Bloque *Signal Source*

- **Random Source:** genera valores aleatorios entre un rango que se especifica, también permite insertar un número de muestras para que se generen. Este bloque tiene entre sus variables la opción de repetir las muestras, esta característica permite tener una transmisión continua de una misma cantidad de información. Se tiene algunas opciones de variables de salida como: *byte, int, short*. Este bloque se observa en la figura 2.5

Figura 2.5 Bloque *Random source*

- **Wav File Source:** este bloque permite seleccionar un archivo de audio en formato "*.wav*". En este bloque se debe especificar la ruta de la ubicación del archivo, también presenta la característica de repetición en transmisión. Adicionalmente, permite tener varias salidas, a diferencia de otras fuentes esta puede entregar más de una sola salida, pero todas las salidas salen con el mismo tipo de variable. Este bloque solamente entrega la información en una variable del tipo f*loat*. Se puede observar el bloque en la figura 2.6

Figura 2.6 Bloque *Wav File Source*

- **Audio Source:** permite la interacción entre GNU Radio y dispositivos de grabación, como el micrófono del ordenador u otro conectado. Además, permite escoger la frecuencia de muestreo con la cual ingresa la información, se puede escoger valores como: 16KHz, 22.05KHz, 24KHz, 32KHz, 44.1KHz, 48KHz, "*samp_rate*". Por defecto se encuentra seleccionado el valor de la frecuencia de muestreo (441KHz), sin embargo, si se mantiene este valor el costo en procesamiento aumenta en gran magnitud, por ello se selecciona un valor distinto menor, esta fuente solo entrega información en variables del tipo *float*. Este bloque se observa en la figura 2.7

Figura 2.7 Bloque *Audio Source*

Las fuentes pueden entregar la información en diferentes tipos de variables, sin embargo, se debe escoger un mismo tipo de variable para que el sistema trabaje adecuadamente, lo que se pretende es reducir el número de conversiones entre variables.

Dado que en su mayoría los bloques pueden trabajar con variables del tipo flotante, con excepción de la fuente aleatoria. Solamente para la fuente aleatoria se utilizará un bloque extra que permita convertir esta variable a tipo flotante. Otros tipos de variables como complejas o enteras, necesitarían de más de 2 bloques conversores, por lo que generan un procesamiento innecesario. Las fuentes con variables de salida del tipo flotante se las puede observar en la figura 2.8.

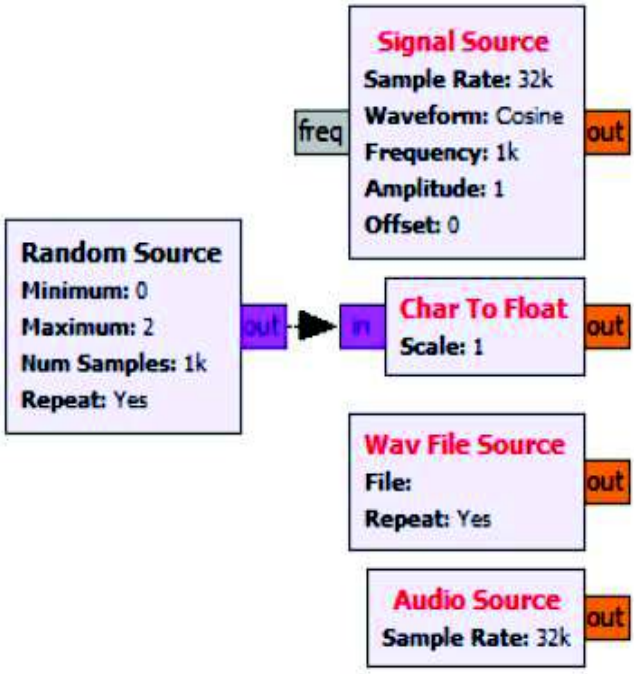

Figura 2. 8 Fuentes con variables de salida del tipo "*float".*

El prototipo puede trabajar con una fuente a la vez, por lo tanto, mientras se utiliza una fuente se deberá deshabilitar las otras. Estas fuentes se deberán conectar a un bloque cuantificador, este deberá especificar el número de bits que usará para tomar las muestras.

Se utiliza el bloque cuantificador desarrollado por [36] de la Universidad Industrial de Santander. La explicación del bloque es la siguiente: *"*Este es un cuantificador que recibe una señal tipo float, la cuantifica y entrega a tipo byte (que es lo mismo que char en c++). Funciona bien para cualquier nivel de cuantificación que esté en el rango del tipo Byte. Así, para el tipo Byte el máximo nivel de cuantización es 256 (es decir: 2^8). Los parámetros usados: Vmax: es el valor máximo de amplitud que puede llegar a alcanzar la señal entrante, bien sea en el rango de los valores negativos o en los positivos, pero aquí se registra ese valor sin signo, como un valor positivo o valor absoluto ; NivelesQ - es el número de niveles de cuantificación a usar en todo el rango dinámico de la señal*"* [36].

Este bloque fue escogido ya que permite controlar el número de bits por muestra, dependiendo de los niveles de cuantización, es decir, es posible controlar la cuantización.

La salida de este bloque será en variable del tipo byte, por lo que reduce el número de bloques necesarios en el procesamiento, se puede observar en la figura 2.9.

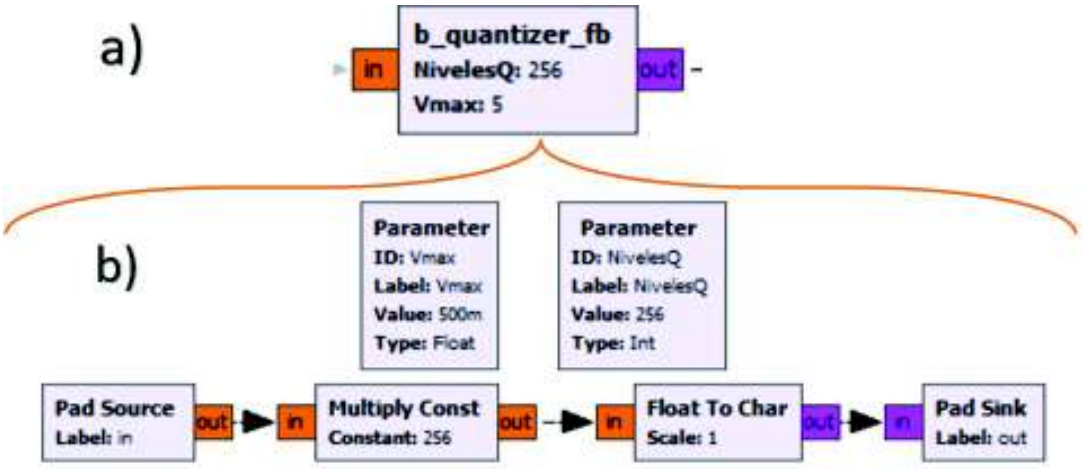

Figura 2.9 a) Bloque cuantificador embebido. b) elementos y variables del bloque cuantificador desarrollado por [36].

Las variables que se generan en este bloque son las siguientes:

- **Número de bits por muestra**: NbpS.
- **Niveles de cuantificación (NivelesQ):** dependerán de NbpS, $NivelesQ = 2^{NbpS}$.
- **Valor Máximo Amplitud Vp:** es el valor de la amplitud que el bloque cuantizador podrá aceptar. Si una señal ingresa con valor superior a este, la señal se recortará.

Para asociar las variables y sus valores es necesario que estos valores dependan de otras variables. De ser necesario se las asocia con funciones matemáticas. De este modo la variable NbpS será ingresada con un valor fijo pero la variable NivelesQ dependerá de NbpS. De este modo, si se tiene mayor cantidad de bits por muestra, los niveles de cuantificación también aumentarán. Esto se puede observar en la figura 2.10.

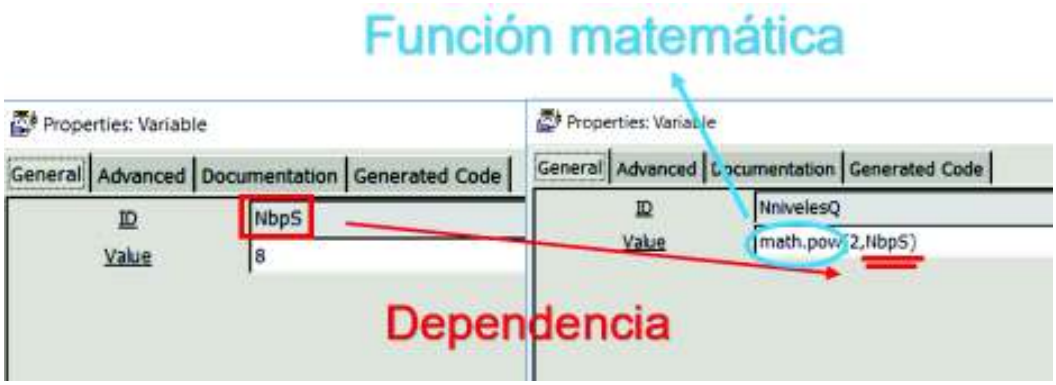

Figura 2.10 Dependencia de Variables

Para poder usar funciones matemáticas es necesario importar librerías al programa, en GNU Radio se deben insertar los bloques de importación de librerías.

Las librerías que el prototipo utiliza son [37]:

- **numPy:** extensión o librería que agrega soporte para trabajar con matrices y vectores.
- **math:** permite el acceso a funciones matemáticas.
- **cmath:** permite el acceso a funciones matemáticas para números complejos.
- **random:** permite el acceso a funciones relacionadas con números aleatorios.

Estas librerías se encuentran en forma de bloques y se las agrega desde las librerías de GNU Radio. Eso se observa en la figura 2.11.

Figura 2.11 Librerías agregadas

A continuación, se presenta en la figura 2.12, la conexión de los bloques con sus respectivas variables y librerías.

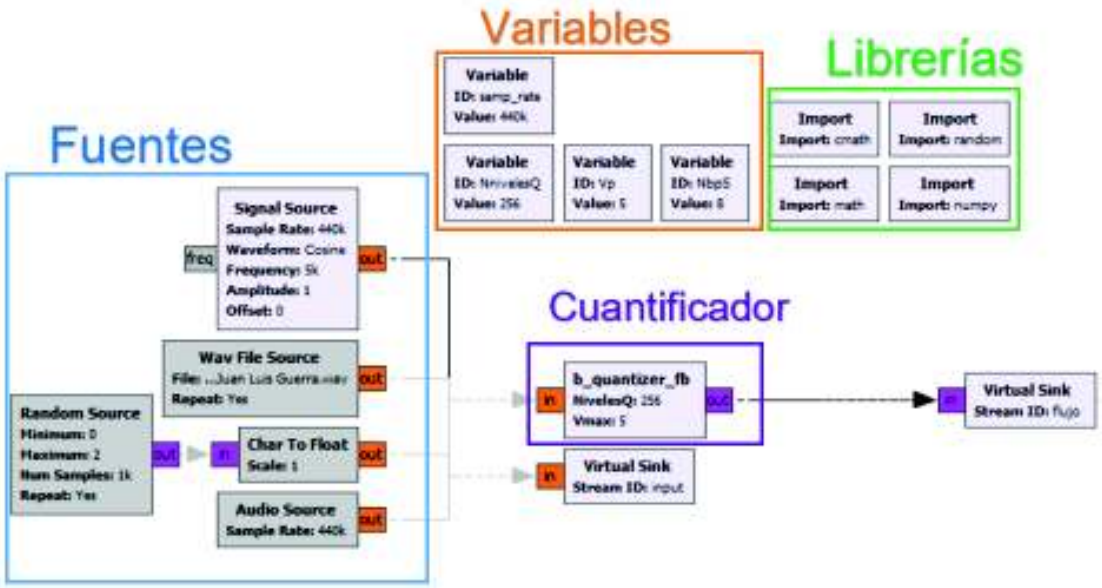

Figura 2.12 Conexión de bloques de procesamiento de la señal

Los bloques “virtual sink” son sincronizadores virtuales, se agregan para mejorar la organización al momento de analizar las señales, permitiendo así tener todos los elementos de análisis y observación de GNU Radio en un solo lugar dentro del área de trabajo.

En este punto la información se entregará al transmisor OFDM como un flujo de bits, representados con una variable del tipo byte.

2.2 Implementación del transmisor OFDM

Con los bloques mencionados en el apartado 1.3.2.2, el proceso de implementación del transmisor se detalla en la figura 2.13.

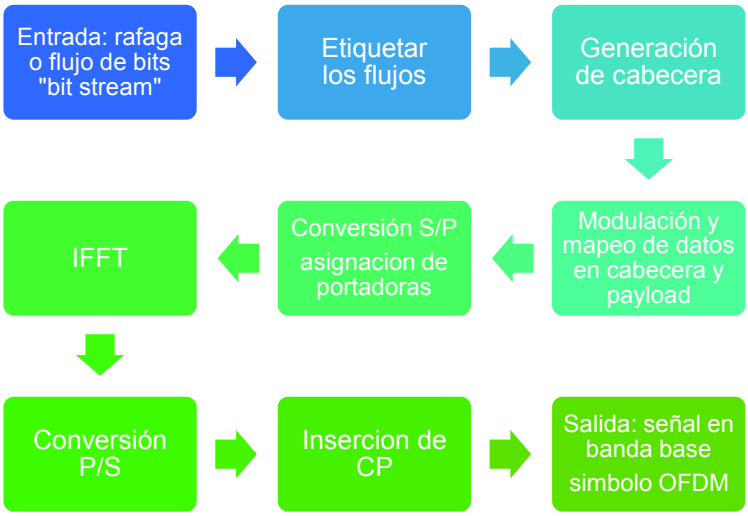

Figura 2.13 Proceso para la implementación del transmisor OFDM

Para la implementación del transmisor OFDM se toma como referencia el trabajo realizado por Bansal Kshitiz y Vishrant Tripathi en [2]. Este trabajo describe la transmisión y recepción de paquetes usando GNU Radio. Principalmente se basa en una comunicación básica de OFDM en banda base usando varias características de la capa física del estándar IEEE 802.11a en OFDM. se describen a continuación:

- Tamaño de N_{FFT}= 64
- Numero de subportadoras=52
- Subportadoras de datos= 48
- Subportadoras piloto=4
- Longitud prefijo cíclico= ¼
- Modulación cabecera: BPSK
- Modulación carga útil: QPSK

Sin embargo, aunque el estándar IEEE 802.11a trabaja en la banda ISM de 5GHz el prototipo no puede trabajar a esta frecuencia de operación, esto se debe al rango de operación de los equipos USRP, los cuales operan en la banda de 50MHz a 2.2GHz.

2.2.1 Etiquetado

La señal que ingresa ha sido previamente procesada y entrega un flujo de bits. Por esta razón, como primer paso la simulación utiliza una fuente aleatoria para emular una entrada de bits.

El etiquetado en el flujo de bits permite segmentar al flujo de bits de manera simétrica, delimitando con longitudes iguales la cantidad de información o número de bits de información en cada grupo o segmento. En esta etapa no se divide al flujo generando varios flujos, sino que se recibe un flujo de bits y se entrega un flujo de bits etiquetado cada "n" bits.

Al etiquetar el flujo los bloques posteriores del transmisor pueden armar los paquetes con el tamaño adecuado al que deberán ingresar para ser procesados, es por ello que la longitud de los paquetes "n" debe ser especificada. Es necesario etiquetar el flujo de bits en flujos de 96 bits, porque la carga útil o "*payload*" será modulada en QPSK, es decir, tomará 2 bits por símbolo, entonces a la salida de la modulación se tiene un flujo etiquetado cada 48 símbolos, mismos que posteriormente serán asignados en las subportadoras de datos. El bloque encargado de realizar el proceso de etiquetado se denomina "*Stream to Tagged Stream*", se lo observa en la figura 2.14.

Figura 2.14 Bloque para etiquetar el flujo de bits.

A continuación, del etiquetado del flujo se agrega un bloque CRC32, este bloque agrega un código de detección de errores en las etiquetas de cada flujo para la detección de los mismos en recepción. Inmediatamente se generan las cabeceras que serán insertadas en los paquetes OFDM, a la par mientras las cabeceras son generadas los flujos etiquetados son empaquetados para ser modulados.

Para definir la longitud del paquete, se crea una variable que se llama: "*packet_len*", esta variable definirá la longitud "n" de los paquetes, en este caso 96 bits.

A la salida del bloque CRC32 se produce un proceso simultaneo entre la carga útil y la generación de una cabecera que será insertada.

2.2.2 Generación de cabecera y empaquetado

La generación de cabecera está a cargo del bloque "*packet header generator*". Este bloque genera una cabecera que depende de dos variables, su longitud y su formato. La longitud de este bloque está asociada a la variable "*packet_len*" y su formato se genera automáticamente, con la variable "*packet_header_ofdm*". La segunda variable genera la cabecera de ofdm y no es posible editarla en el diagrama de bloques. Inmediatamente después de generar las cabeceras que se insertarán en los paquetes, se modulan los datos de las cabeceras.

El empaquetado depende directamente del nivel de modulación que se utilizará, porque se agrupa bits dependiendo del nivel de modulación, en el empaquetado de la carga útil se toman "*M*" bits de entrada, este grupo de bits se dividen en subgrupos de 2 bits y se entrega 1 símbolo por cada subgrupo de bits de entrada. Por ejemplo, con QPSK, el empaquetado entregará 4 pedazos de paquetes o subgrupos que serán los 4 niveles de la modulación, estos pedazos de paquetes se encuentran dentro de un paquete o grupo. En otras palabras, entrega 4 símbolos por cada 8 bits, es decir 2 bits por cada símbolo, este proceso se realiza para cada segmento del flujo que ingresa. El proceso de empaquetado es realizado por el bloque denominado "*Repack Bits*", esto se observa en la figura 2.15.

Figura 2.15 Etiquetado, generación de cabecera y empaquetado de bits de datos

En el bloque "*Repack Bits*" es necesario especificar el número de bits de entrada, como de salida, es decir, se debe especificar la cantidad de bits que ingresan y cuantos símbolos

se entregan a la salida, este empaquetamiento permite la modulación por asignación de valores según el esquema de modulación que se utilice. Esto se observa en la figura 2.16.

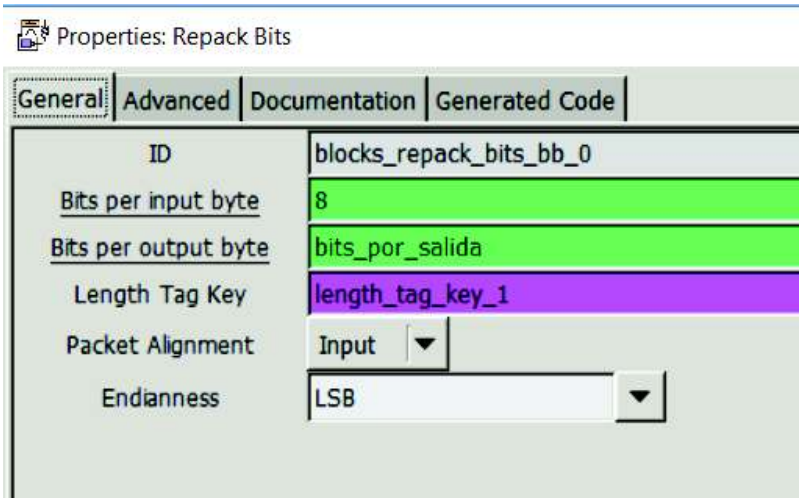

Figura 2.16 Propiedades Bloque “Repack Bits”

Mientras que el número de bits de entrada están dados por un valor fijo, los bits de salida dependen de una variable denominada “bits_por_salida”, el valor de esta variable y sus características se observan en la figura 2.17.

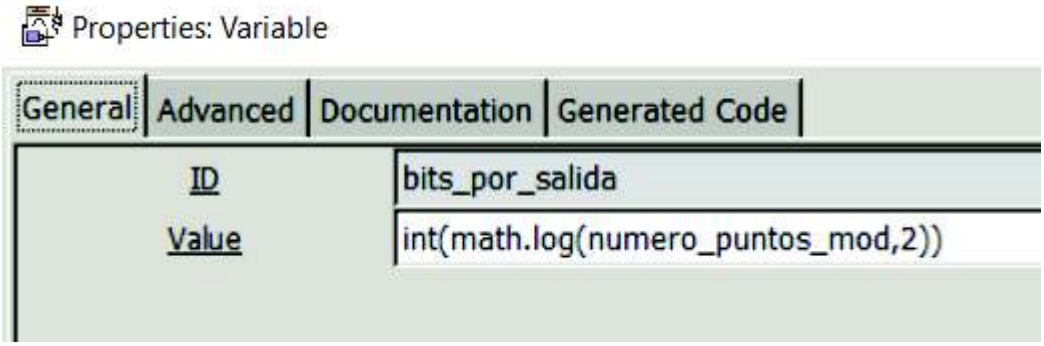

Figura 2.17 Variable "bits_por_salida"

La variable *“bits_por_salida*” depende de la variable “*numero_puntos_mod*”. La operación matemática que realiza se observa en la ecuación 2.1.

$$log_2(\text{ numero\_puntos\_mod }) = \text{bits\_por\_salida}$$

Ecuación 2. 1 Numero de bits de salida.

La ecuación 2.1 especifica que el valor de los bits de salida en el empaquetado depende del esquema de modulación, por ejemplo, si se utiliza una modulación QPSK con 4 niveles de modulación, el valor de la variable “numero_puntos_mod” será de 4, entonces la variable “bits_por_salida” entregará a la salida 1 símbolo por cada 2 bits, o lo que es lo mismo 4 subgrupos de 2 bits en cada paquete. Al hacer esto prepara la información para ser modulada en el siguiente bloque.

2.2.3 Modulación de cabecera y carga útil

Para la modulación no se utilizan los bloques "*M-PSK Mod*" o "*Constellation Mod*", de GNU radio, sino se modula asignando valores dependiendo de una tabla de modulación. Se toman muestras de M bits, y asignando a los diferentes valores permitidos según una tabla de valores de modulación (depende del esquema de modulación BPSK o QPSK), para la cabecera BPSK y para la carga útil QPSK, es por ello que el empaquetado de los flujos de bits esta enlazado con la asignación de valores para la modulación.

El proceso de asignar un valor "I & Q" a un conjunto de bits se denomina mapeo de datos, el mapeo de datos se efectua en el bloque "*Chunks to Symbols*". El proceso se realiza insertando los valores "I & Q" en una tabla y el bloque asigna estos valores a los conjuntos de bits que ingresan dependiendo del esquema de modulación que se utilice, pues cada esquema de modulación posee diferentes valores "I & Q" en sus tablas.

Para QPSK se tiene 4 posibles valores de constelación, el módulo de estos valores es 1, y deben estar separados 90º. Estos puntos se muestran a continuación: {(0,70710,0.70710); (-0.70710, 0.70710); (-0.70710,- 0.70710); (0.70710,- 0.70710)}. Para la modulación BPSK de la cabecera únicamente se tiene 2 puntos separados 180°, estos puntos son: 1,-1. Los diagramas de constelación de estas dos modulaciones se observan en la figura 2.18.

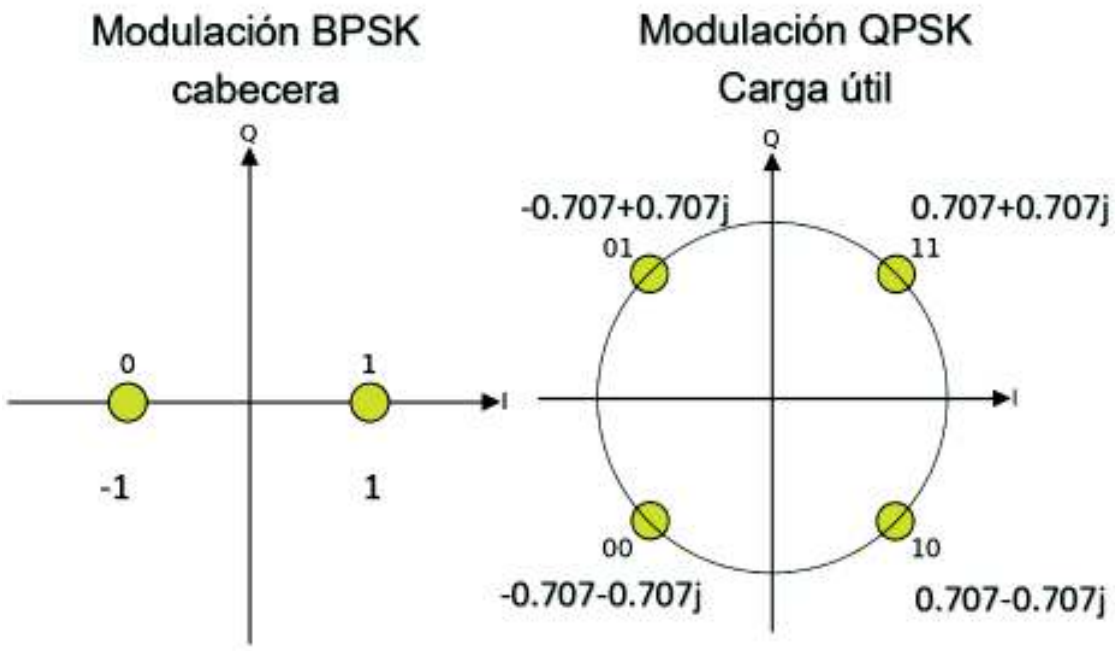

Figura 2.18 Modulación de Cabecera y Carga Útil

Para la carga útil la modulación es QPSK, 4 niveles de modulación, 2 bits se asignan a cada símbolo. Para la cabecera la modulación que se utiliza es BPSK, dos niveles de modulación. Esto se observa en la figura 2.19.

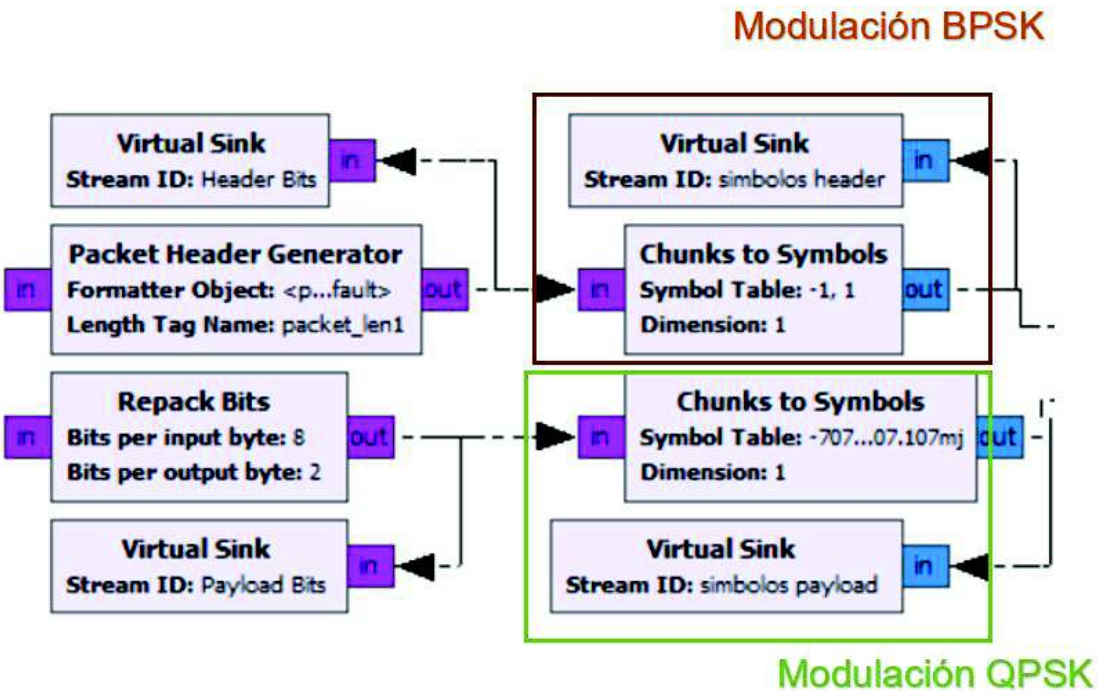

Figura 2. 19 Asignación de valores de puntos de constelación

Hasta esta instancia la información y la cabecera están moduladas, tanto en QPSK como en BPSK respectivamente. Ahora es necesario unir las cabeceras y la carga útil, es decir, insertar las cabeceras al inicio de los segmentos para formar los paquetes, para ello se utilizará el bloque "*Tagged Stream Mux*". Este bloque permite multiplexar los flujos de información que fueron etiquetados con las cabeceras generadas.

La sección que comprende la fuente de información, etiquetado del flujo, generación de cabecera, empaquetado de bits, mapeo de datos y multiplexación de flujos se denomina "*Pre-OFDM*", esto se debe a que esta información se encuentra procesada y lista para ingresar a los bloques OFDM. Además, tanto la información de la cabecera como la carga útil se encuentran moduladas y multiplexadas, pero aún se encuentra en un flujo de símbolos en serie y en el dominio de la frecuencia. Esta etapa, se observa en la figura 2.20.

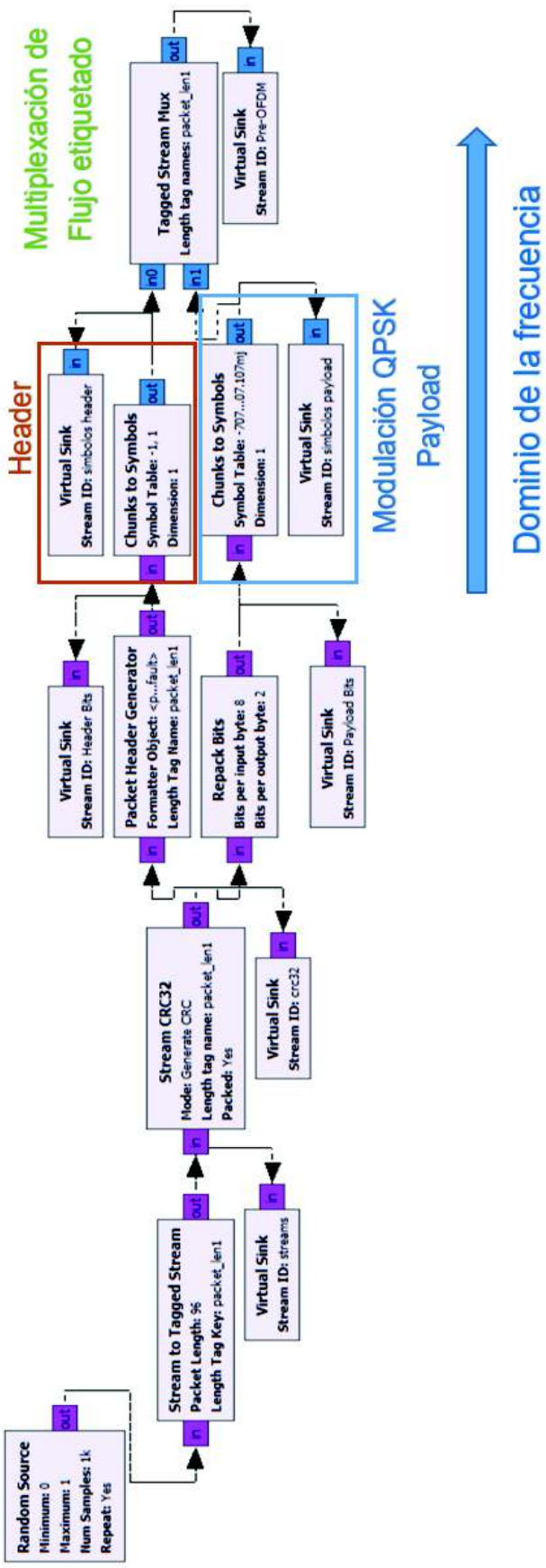

Figura 2. 20 Sección Pre-OFDM

2.2.4 Asignación de portadoras

Es el proceso en el cual se atribuye o asigna a cada subportadora la función u obligación que cumplirá en el sistema, sea: transportar datos, banda de guarda, piloto, etc. El bloque encargado de realizar este proceso es "*OFDM carrier allocator*". Este bloque asigna o entrega las funciones u obligaciones a las diferentes portadoras, también asigna los símbolos piloto y palabras de sincronismo. Se lo puede observar en la figura 2.21.

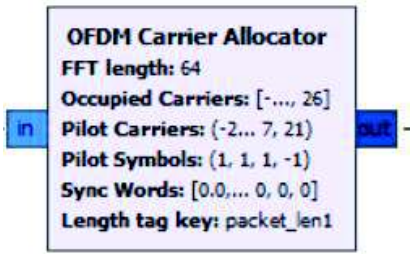

Figura 2. 21 Bloque de asignación de subportadoras

En primer lugar, el bloque dependerá del número total de subportadoras, este número se define con la variable "*fft_len*", se asigna un valor de 64 a esta variable para usar algunas características del estándar 802.11a como en [2], de esta manera se especifica que se utilizarán 64 subportadoras numeradas del -31, hasta el 32 incluyendo el cero.

Del total de 64 subportadoras:

- **Subportadoras piloto:** -21-7,-7,21, en total 4 pilotos.
- **Subportadoras de datos:** desde la -26 hasta la -22 (5), desde la -20 hasta la -8 (13), desde la -6 hasta la -1 (6), desde la 1 hasta la 6 (6), desde la 8 hasta la 20 (13), desde la 22 hasta la 26 (5). En total 48 subportadoras de datos.
- **Subportadora DC:** 0, esta es la subportadora central, esta subportadora tendrá la frecuencia central de funcionamiento del sistema asignada por el equipo USRP.
- **Subportadoras nulas:** desde la -31 hasta la -27 y desde la 27 hasta la 32, 11 subportadoras usadas como bandas de guarda.

Además de asignar las funciones a las subportadoras este bloque permite ingresar símbolos piloto y palabras de sincronismo. Los símbolos piloto son insertados en las subportadoras piloto junto con las palabras de sincronismo, esto permite la sincronización en recepción para recuperar la información.

Como varias variables y asignaciones de variables se usan tanto en transmisión como en recepción, se crean varias variables. De esta manera no es necesario ingresar

manualmente estos valores más de una vez. Estas variables son: "*occuped_carriers*", "*pilot_carriers*", "*pilot_symbols*", "*sync_word1*" y "*sync_word2*". De esta manera el bloque de asignación de portadoras y los bloques de sincronismo del receptor dependerán de estas variables.

Al ingresar los valores de: los símbolos piloto, palabras de sincronismo, portadoras piloto y portadoras de datos en las variables, solamente es necesario asociar las variables con los bloques de asignación de portadoras en transmisión y en sincronismo en recepción, de esta manera se crea una dependencia entre los bloques y estas variables. Los símbolos pilotos y palabras de sincronismo son los mismo que se utilizan en [2]. Las palabras de sincronismo contienen 64 símbolos, mismo que se utilizan para emular las 64 subportadoras. Esto se observa en la figura 2.22.

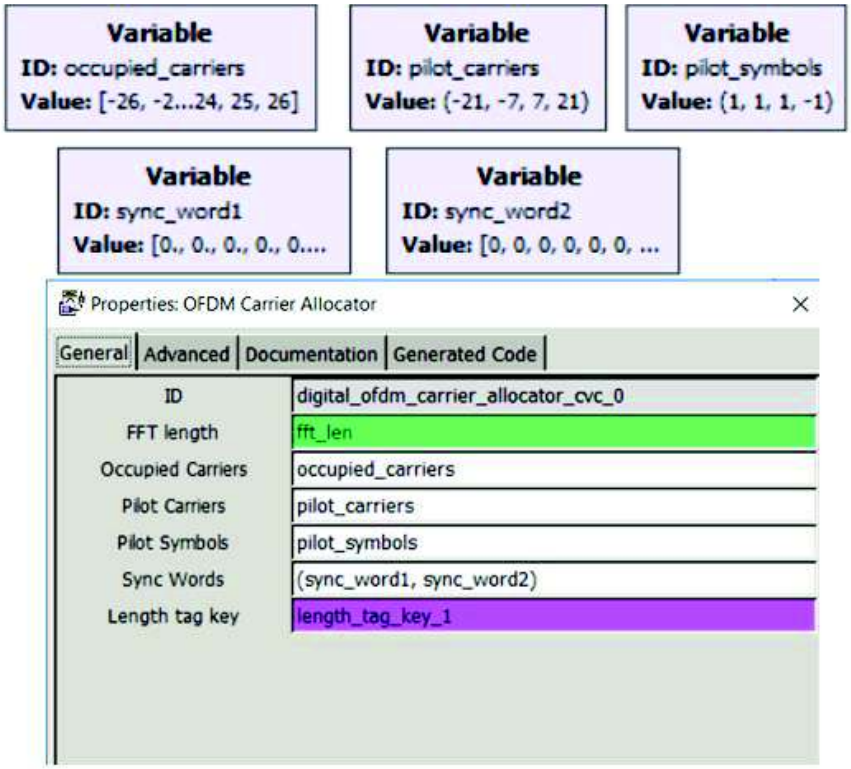

Figura 2. 22 Bloques de variables para asignación de portadoras

Los símbolos piloto y las palabras de sincronismo ayudan al bloque de sincronismo del cual se hablará posteriormente, cabe recalcar que al tomar como referencia [2] se utilizan los mismos valores en estas variables.

El bloque "*OFDM carrier allocator*" también es el encargado de realizar la conversión serie a paralelo, de esta manera inserta la información en las subportadoras de datos, enviando flujos paralelos al bloque IFFT.

2.2.5 IFFT

El bloque encargado de realizar el proceso de la IFFT se denomina “FFT”, este bloque trabaja en dos modos “forward” y “reverse”. Estos modos representan a la FFT e IFFT respectivamente. Depende de N_{FFT}, por esta razón se crea una variable llamada “*fft_len*” que representa la longitud de la FFT y a su vez el número de subportadoras que el sistema utiliza. Tanto el bloque como la variable se observan en la figura 2.23.

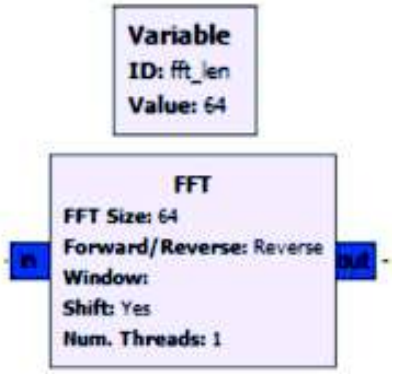

Figura 2. 23 Bloque FFT

2.2.6 Prefijo cíclico

El bloque encargado de insertar el prefijo cíclico a la señal es el bloque denominado “*OFDM Cyclic Prefier*”, este bloque depende N_{FFT}. Para calcular la longitud del prefijo cíclico en este bloque se ingresa dicho valor con una operación matemática, esto se realiza para que el valor sea dependiente de N_{FFT}. Se utiliza un valor 1/4 N_{FFT} para el prefijo cíclico. Esto se observa en la figura 2.24.

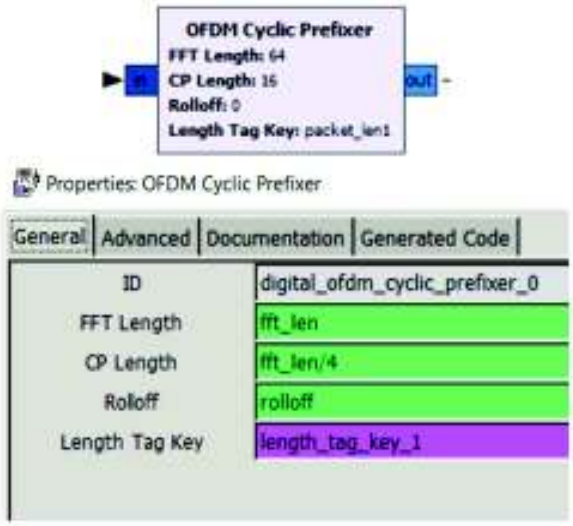

Figura 2. 24 Propiedades del bloque “*OFDM Cyclic Prefixer*”

La conexión de los bloques de la sección OFDM, se la puede observar en la figura 2.25.

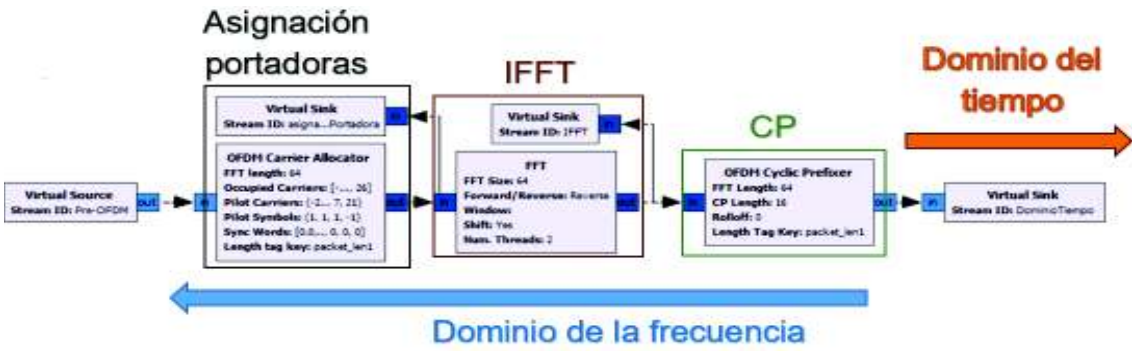

Figura 2. 25 Sección de Bloques OFDM

La señal que entrega a la salida del bloque CP se encuentra en el dominio del tiempo, en esta instancia la señal esta lista para ser enviada al equipo USRP, o en el caso de la simulación a un bloque llamado modelo de canal. En cualquier caso, esta señal ha sido procesada en el transmisor y es la que viaja en el canal.

La conexión de todos los bloques del transmisor OFDM, con sus secciones y configuraciones se observa en la figura 2.26.

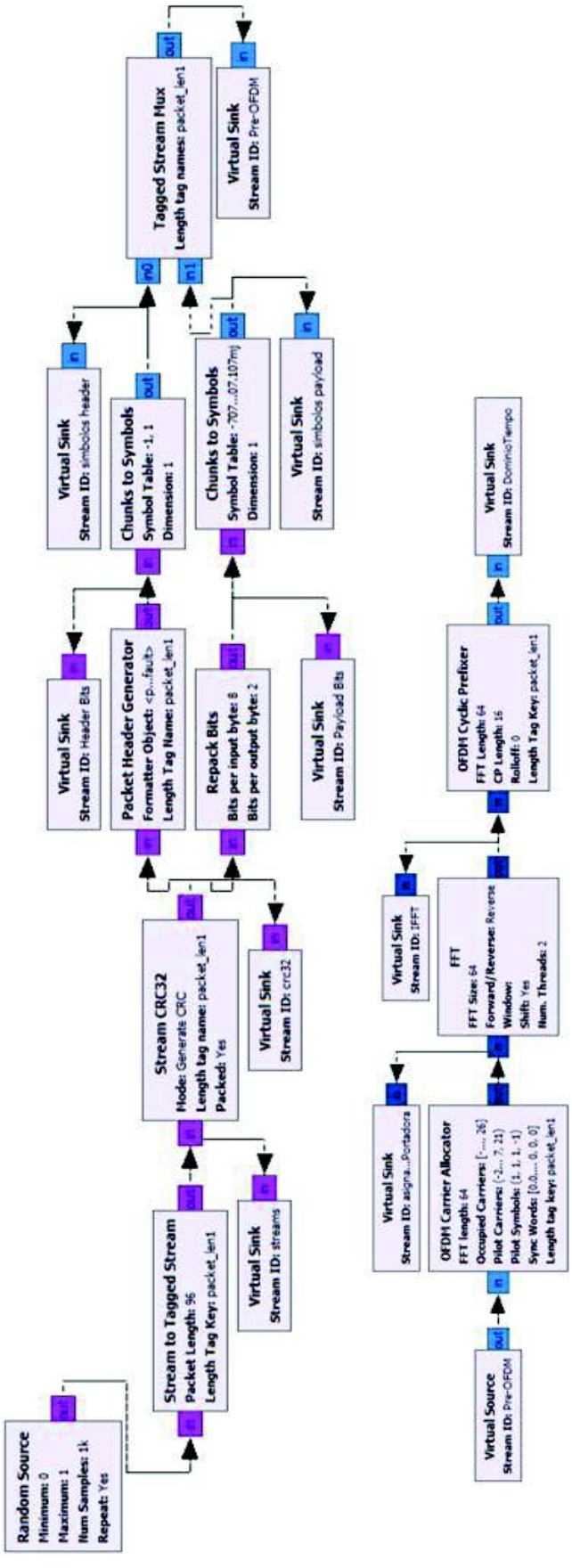

Figura 2. 26 Transmisor OFDM

2.3 Implementación del receptor OFDM

La implementación del Receptor OFDM se realiza por secciones de la misma manera que el transmisor, el proceso de la implementación del receptor se lo puede observar en la figura 2.27.

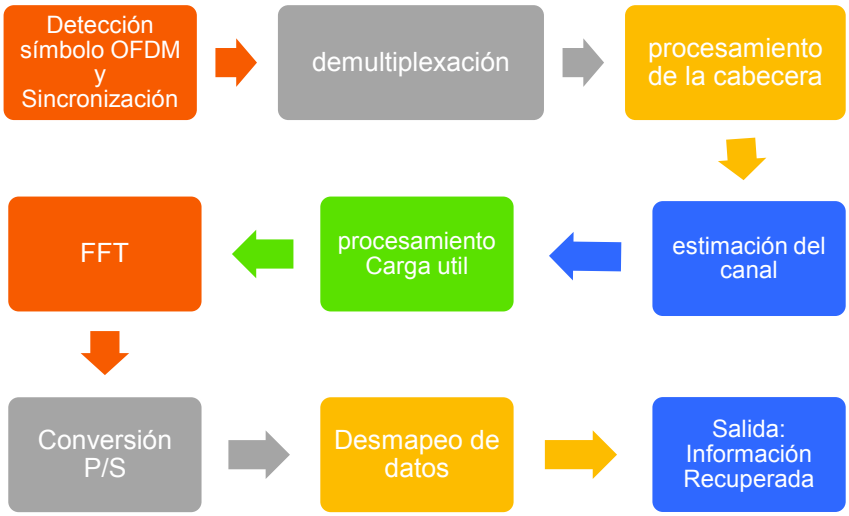

Figura 2. 27 Proceso para la implementación del Receptor OFDM

El diseño del receptor OFDM está basado en el trabajo realizado por Bansal Kshitiz y Vishrant Tripathi en [2] pero también en [21] que fue desarrollado por Yun Chiu y Dejan Markovic. Al igual que el transmisor se debe configurar al receptor para trabajar con varias características del estándar IEEE 802.11a. Para que exista compatibilidad entre transmisor y receptor es necesario especificar qué características se utilizan, estas se encuentran especificadas en la sección 2.2.

A continuación, se detalla el proceso de implementación del receptor OFDM con las diferentes secciones y bloques que lo comprenden.

2.3.1 Detección símbolo OFDM

Lo primero que se debe realizar es detectar el símbolo OFDM, para ello es necesario sincronizar al receptor de manera que pueda reconocer el símbolo, además debe detectar el desplazamiento en frecuencia. Ambos procesos son parte de esta sección.

2.3.1.1 Sincronización

La sincronización es la primera parte de la sección de detección del símbolo OFDM en el receptor. La sincronización es el proceso que se realiza en recepción para detectar la información que se recibe delimitando la o las tramas de datos, además este proceso también permite detectar un desplazamiento en frecuencia o tiempo y aplicar algún mecanismo para contrarrestar estos fenómenos. El bloque de sincronización que se utiliza

en el prototipo es el método de bloque de sincronismo basado en el algoritmo de Schmidl & Cox [38].

El sincronizador Schmidl & Cox está basado en el algoritmo del mismo nombre y se fundamenta en la búsqueda de símbolos de entrenamiento con dos mitades idénticas en el dominio del tiempo para estimar el tiempo compensado, es decir, para la recuperación del tiempo de símbolo busca un símbolo de entrenamiento o palabra de sincronismo con dos mitades idénticas en el dominio del tiempo que después de haber atravesado el canal permanecerán idénticas, la única diferencia será que habrá una diferencia de fase entre ellas ocasionada por el desplazamiento de frecuencia de la portadora [38].

Para el primer símbolo de entrenamiento o palabra de sincronismo se usan secuencias de ceros y pseudoruido ("*pseudonoise*" o PN), ceros en las frecuencias impares y una secuencia pseudoaleatoria de ruido en las frecuencias pares. Además se multiplica por $\sqrt{2}$ con el fin de mantener una energía de la señal aproximadamente constante [38]. Esto se puede observar en las figuras 2.28.

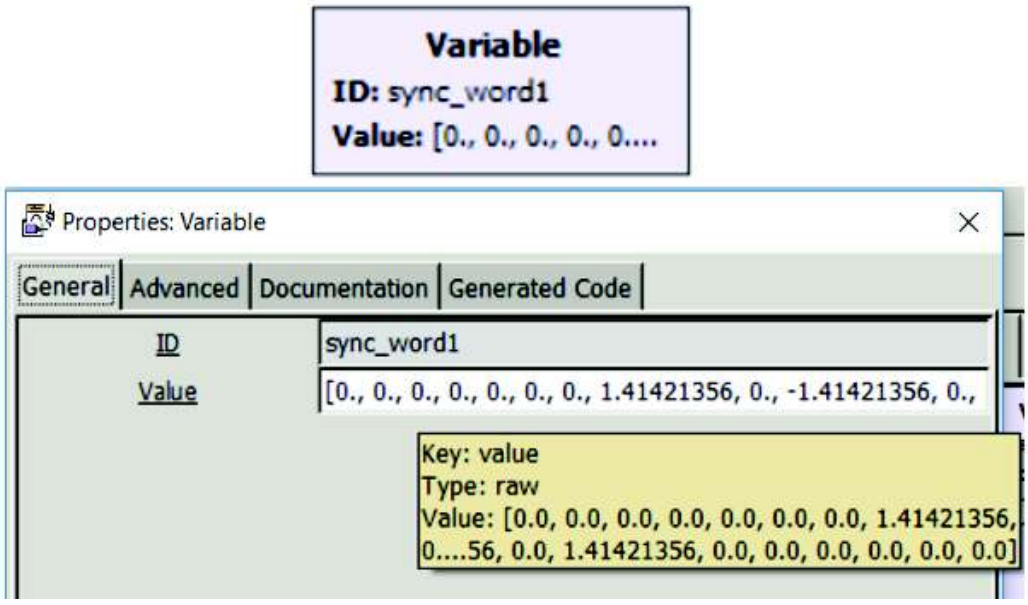

Figura 2. 28 Variable Primer símbolo de entrenamiento o palabra de Sincronismo Nº1

La primera secuencia de entrenamiento que se utiliza se observa en la figura 2.29 y se presenta de la siguiente manera:

- Bandas de guarda: verde
- Frecuencias pares: azul
- Frecuencias impares: naranja

Figura 2. 29 Primer símbolo de entrenamiento o palabra de sincronismo 1

El segundo símbolo o secuencia de entrenamiento contiene dos secuencias PN, una para las frecuencias pares y otra para las frecuencias impares, en las frecuencias impares ayuda a medir los subcanales y en las frecuencias pares ayuda a determinar el desplazamiento de frecuencia [38]. La segunda secuencia se la puede observar en la figura 2.30 y se presenta de la siguiente manera:

- Bandas de guarda: verde
- Frecuencias pares: azul
- Frecuencias impares: negro
- Frecuencia central: rojo

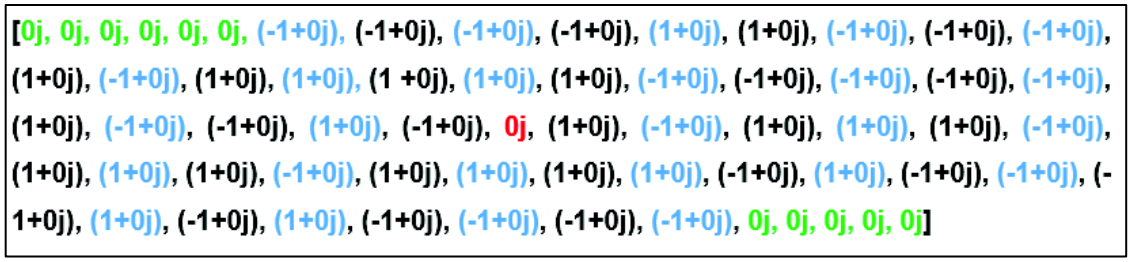

Figura 2. 30 Segundo símbolo de entrenamiento o palabra de sincronismo Nº2

Estas secuencias de entrenamiento o palabras de sincronismo son enviadas por el transmisor al inicio de la comunicación para ayudar en el sincronismo, sin embargo, también tienen otro propósito, que es ayudar en la estimación del canal, de lo cual se hablará más adelante en esta sección.

El bloque encargado de realizar el sincronismo en el prototipo es llamado: "*Schmidl & Cox OFDM synch*". Se lo puede observar en la figura 2.31.

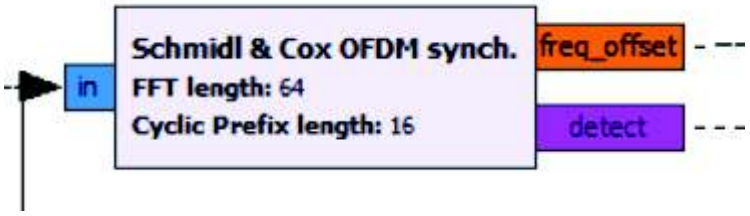

Figura 2. 31 Bloque de sincronización "*Schmidl & Cox OFDM synch*"

Este bloque no procesa la información, sino que se encarga de sincronizar a la información tanto en tiempo como en frecuencia. Este bloque tiene a la entrada la información y a la salida entrega un valor de desplazamiento en frecuencia llamado "*freq_offset*" y también una salida llamada "*detect*", esta entrega la información sobre la detección de las palabras de sincronismo o secuencias de entrenamiento. La salida "*detect*" de este bloque será la encargada de ayudar al momento de delimitar símbolos, cabeceras y "*payload*".

El bloque no entrega inmediatamente el valor la desviación de frecuencia, sino cuánto cambia la fase en función de la nueva muestra de entrada, en otras palabras, los valores cambian por cada muestra, son valores instantáneos que no se pueden utilizar, por esta razón este valor no es el que se utiliza para corregir al símbolo OFDM. La desviación en frecuencia se obtiene utilizando un modulador analógico de frecuencia, este modulador entrega el valor al cual varían las muestras de frecuencia, o en otras palabras el desplazamiento en frecuencia del símbolo. El bloque que se encarga de realizar este proceso se denomina "*Frequency Mod*", esto se observa en la figura 2.32.

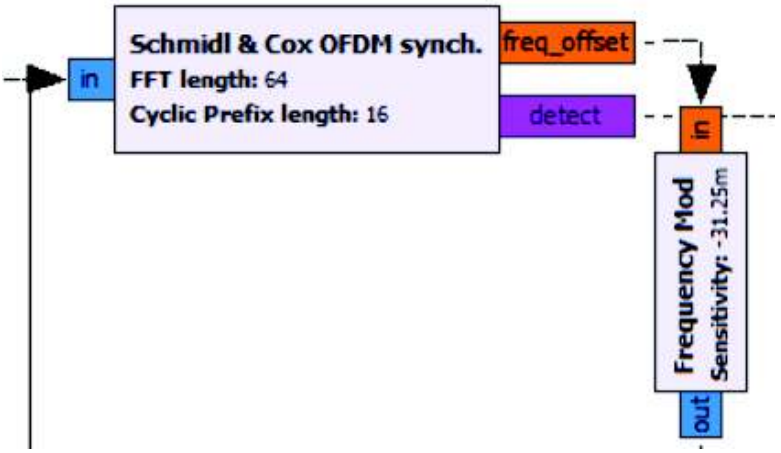

Figura 2. 32 Modulador analógico de frecuencia

El bloque de retardo o "*delay* "se agrega para que mientras se procesa un símbolo en el sincronizador a la salida sea el mismo símbolo corregido, es decir, el símbolo que ingresa en el sincronizador también se retrasa la duración de un símbolo OFDM.

Para calcular la duración del símbolo OFDM es necesario tomar en cuenta la duración del prefijo cíclico como se mencionó en la sección 1.3. Se conoce que la duración del prefijo

cíclico insertado en el intervalo de guarda es ¼ Nfft, la duración del símbolo OFDM sin el CP es 3.2 µs, por lo tanto, la duración del CP es de 0.8 µs y la duración del nuevo símbolo o símbolo extendido será 4 µs. Si se escribe lo mismo con subportadoras resulta: 64 subportadoras (N) + ¼ N, resulta 64 + 16 lo que da como resultado 80 subportadoras, por ello se ingresa este valor en el bloque "*delay*" o retraso. El proceso completo de sincronismo se lo puede observar en la figura 2.33

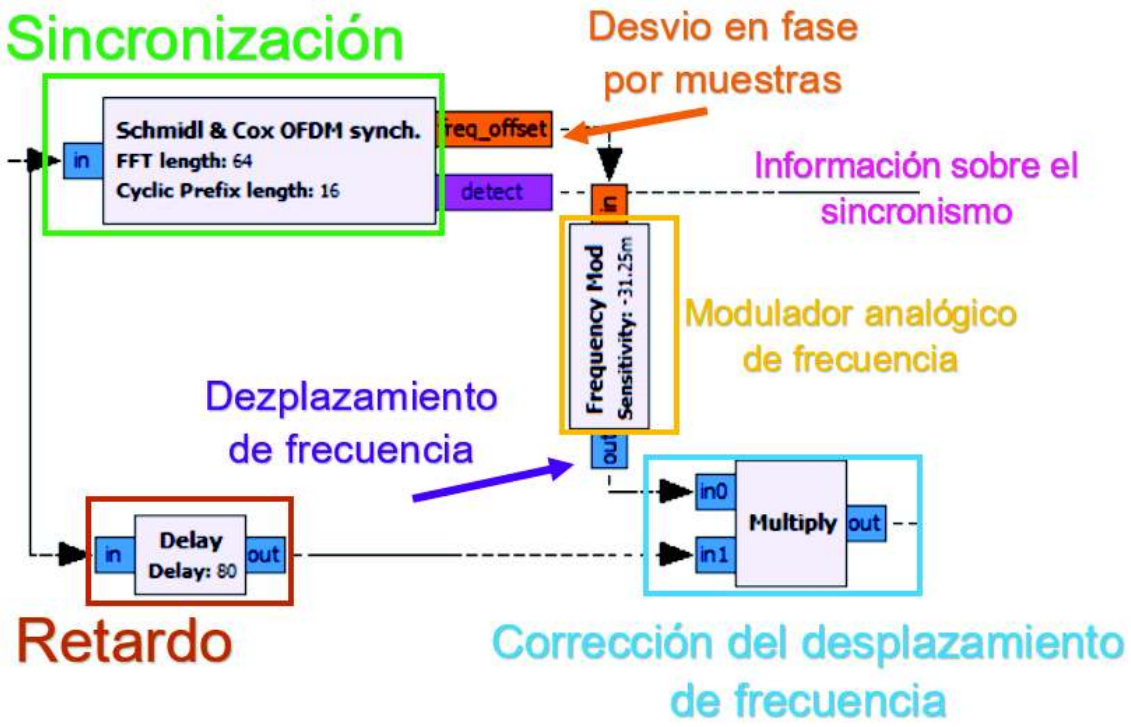

Figura 2.33 Bloques de sincronismo

Después de este procesamiento se puede observar que se tiene 2 salidas, la información de control que sale del bloque sincronizador y la información con la corrección de frecuencia. Sin embargo, la información no ha sido procesada, razón por lo cual se mantiene en el dominio del tiempo y como un flujo en serie. Sin embargo, este flujo ha sido corregido del desvío de frecuencia.

2.3.2 Demultiplexación de cabecera

A continuación, después del bloque de sincronismo se tiene el bloque demultiplexor, este bloque se encarga de demultiplexar las cabeceras de la carga útil, este bloque posee 3 entradas, los datos o información de la carga útil, los datos de control del bloque de sincronismo, y la información de control de las cabeceras después de ser procesada. A la salida entrega las cabeceras y "*payload*" respectivamente, se lo puede observar en la figura 2.34.

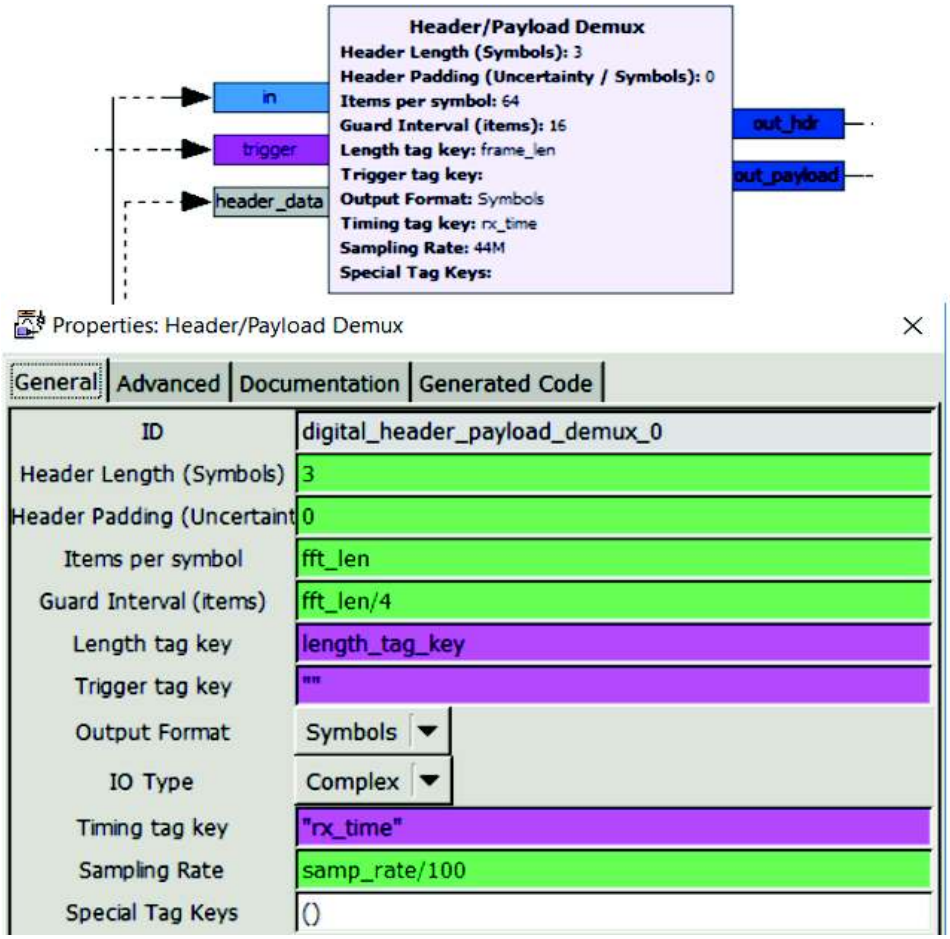

Figura 2. 34 Bloque Demultiplexor cabecera/carga útil

Este bloque depende directamente de: la información de las cabeceras y la información de control que el bloque de sincronismo entrega. Entonces, a partir de esta información es capaz de discernir entre información útil o carga útil y los datos de cabecera, procediendo así a entregar ambos valores en dos salidas independientes. Cabe recalcar que ambas salidas entregan información y esta información aún se encuentra en el dominio del tiempo y como un flujo en serie, es decir, el bloque demultiplexor entrega dos flujos de información en serie y en el dominio del tiempo, uno para cabeceras y uno para la carga útil.

Los flujos que se obtienen del multiplexor se deben procesar por separado, sin embargo, su procesamiento es similar, aunque su finalidad es distinta.

2.3.3 Procesamiento de la cabecera

El flujo de información que llega desde el bloque demultiplexor debe ser procesada por los bloques OFDM encargados del procesamiento. En primer lugar, es necesario convertir el flujo en serie a varios flujos en paralelo, pero también cambiar del dominio del tiempo al domino de la frecuencia, ambos procesos los realiza el bloque FFT. Entonces el primer bloque del procesamiento de la cabecera es el bloque FFT y se lo puede observar en la figura 2.35.

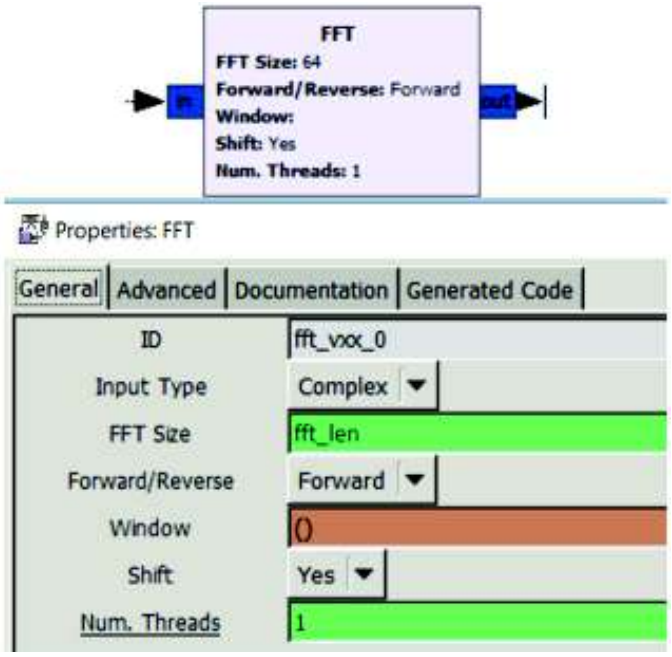

Figura 2. 35 Bloque FFT para procesamiento de la cabecera

El bloque FFT como se había indicado trabaja de dos maneras, para realizar el proceso de la FFT se selecciona el modo "*forward*", además es imprescindible especificar la longitud o tamaño de la FFT o N_{FFT}, este mismo valor había sido usado en el transmisor y se creó la variable "*fft_len*" así que se utiliza esta misma variable para especificar este valor.

Después que la información de la cabecera sale del bloque FFT como flujos en paralelo, estos ingresan al bloque de estimación de canal.

2.3.3.1 Estimación de canal

El bloque de estimación de canal se encarga de estimar el comportamiento de la señal sobre el canal y como este afecta a la señal que está atravesándolo. Esto lo realiza utilizando secuencias de entrenamiento o símbolos de entrenamiento conocidos, en el prototipo estos símbolos son las palabras de sincronismo.

De esta manera, las mismas variables que se utilizaron para las palabras de sincronismo se utilizan como símbolos de entrenamiento para el proceso de estimación (se las puede observar en las figuras 2.29 y 2.30.), el bloque encargado de realizar el proceso de estimación de canal para el sistema OFDM se denomina "*OFDM Channel Estimation*" y se lo puede observar en la figura 2.36

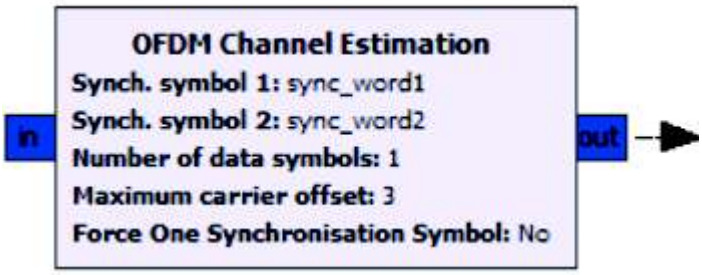

Figura 2. 36 Bloque para la estimación de canal OFDM

La información que se obtiene a partir del proceso de estimación de canal ingresa al bloque "*OFDM Frame Equalizer*" este bloque forma parte del proceso de estimación de canal. El bloque ecualizador dependerá de tanto de la longitud de la fft como del prefijo cíclico, este bloque se observa en la figura 2.37.

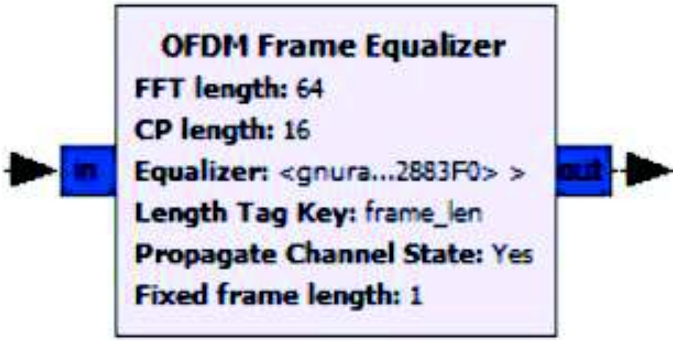

Figura 2. 37 Ecualizador OFDM

El bloque ecualizador toma la información de la estimación del canal y se encarga de ayudar a contrarrestar las anomalías del canal, envía la información de control para poder

procesar de una manera adecuada la información de la carga útil o "*payload*". Después del proceso de estimación de canal y ecualización la información de control de la cabecera se encuentra como flujos paralelos en el dominio de la frecuencia. Esta información se envía al bloque conversor P/S.

2.3.3.2 Conversión P/S

El bloque "*OFDM serializer*" es el encargado de convertir los flujos en paralelo en un solo flujo en serie. Su función es tomar la información que se insertó en las subportadoras en el proceso de asignación de subportadoras y agruparlas en un mismo flujo, es decir, toma la información de las cabeceras de las 48 subportadoras de datos que llegan como flujos paralelos y las convierte en un flujo en serie. Se observa este bloque en la figura 2.38.

Figura 2. 38 Bloque serializador o conversor P/S

Los bloques FFT, estimación de canal, ecualización y convertidor P/S, componen el procesamiento OFDM para la cabecera y se los observa en la figura 2.39.

Figura 2. 39 Procesamiento OFDM de la cabecera

La información a la salida del serializador se compone de: un flujo de información de control, en serie y en el dominio de la frecuencia. Esta información está lista para ser desmapeada o demodulada.

2.3.3.3 Demodulación de la cabecera

Para la demodulación de la cabecera se utiliza el bloque "*constellation decoder*", este bloque realiza el proceso inverso al bloque "*chunks to symbols*". Toma los símbolos y los desmapea según la tabla de modulación que se utiliza en el transmisor. En la cabecera se utiliza la tabla de demodulación para BPSK. Con el proceso inverso del mapeo llamado desmapeo, la información cambia del dominio de la frecuencia al dominio del tiempo. Este bloque se lo observa en la figura 2.40.

Figura 2. 40 Bloque decodificador de constelación

Este bloque necesita de una variable para asociar la tabla de modulación BPSK con el proceso de desmapeo, para ello se emplea la misma variable del transmisor, esta variable es "*header_mod*" y se encuentra en un bloque "*constellation object*" para la modulación y demodulación de la cabecera.

Las propiedades del bloque "*constellation object*" se las puede observar en la figura 2.41.

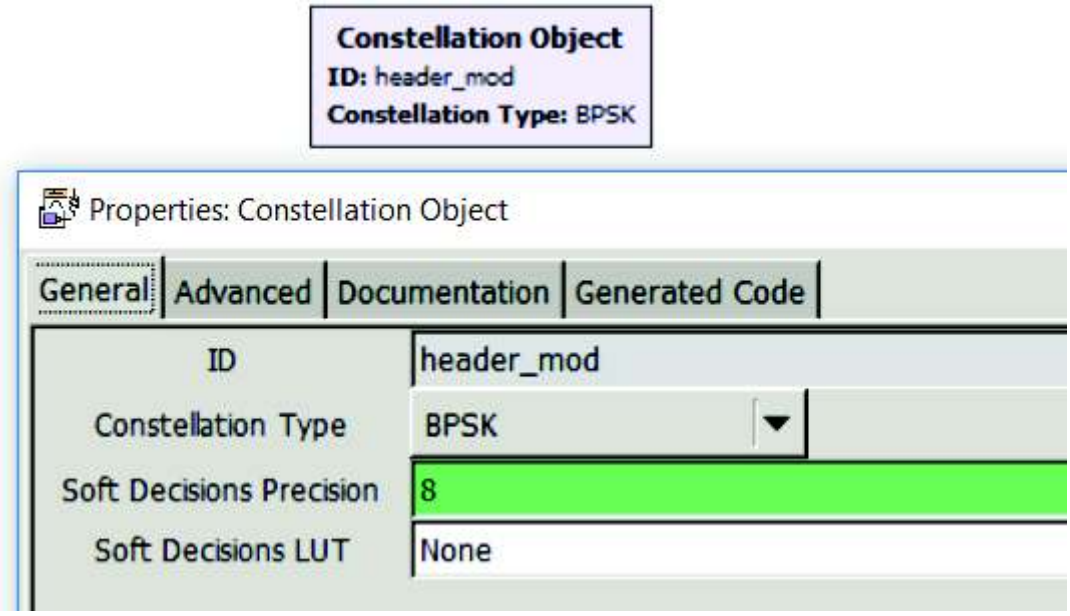

Figura 2. 41 Bloque "*Constellation Object*" y sus propiedades

Para finalizar esta etapa de procesamiento de cabecera es necesario hacer una realimentación, la información de control de las cabeceras, estimación de canal y ecualización son enviadas hacia el bloque "*Header/Payload demux*", por esta razón la información que sale del bloque demodulador debe ser separada antes de entrar al demultiplexor. El bloque que analiza la información de las cabeceras y entrega los datos al demultiplexor se llama "*Packet Header Parser*" y se lo observa en la figura 2.42.

Figura 2. 42 Bloque analizador de cabecera

Este bloque separa la información y la entrega directamente al demultiplexor con el fin de realimentar este procesamiento y aplicar las correcciones de estimación de canal y ecualización. Esta conexión se observa en la figura 2.43.

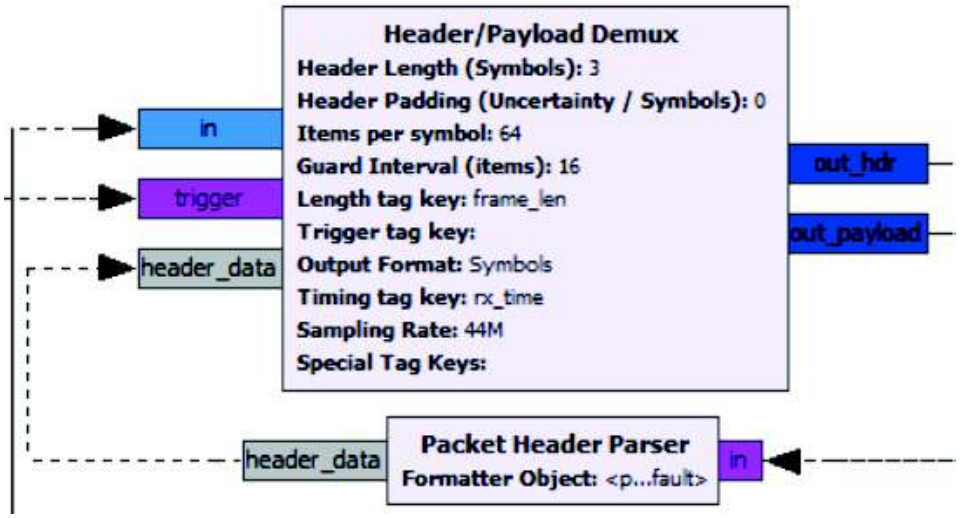

Figura 2. 43 Retroalimentación del demultiplexor con la información de la cabecera

Las etapas de procesamiento de cabecera, sincronismo y retroalimentación acopladas se observan en la figura 2.44.

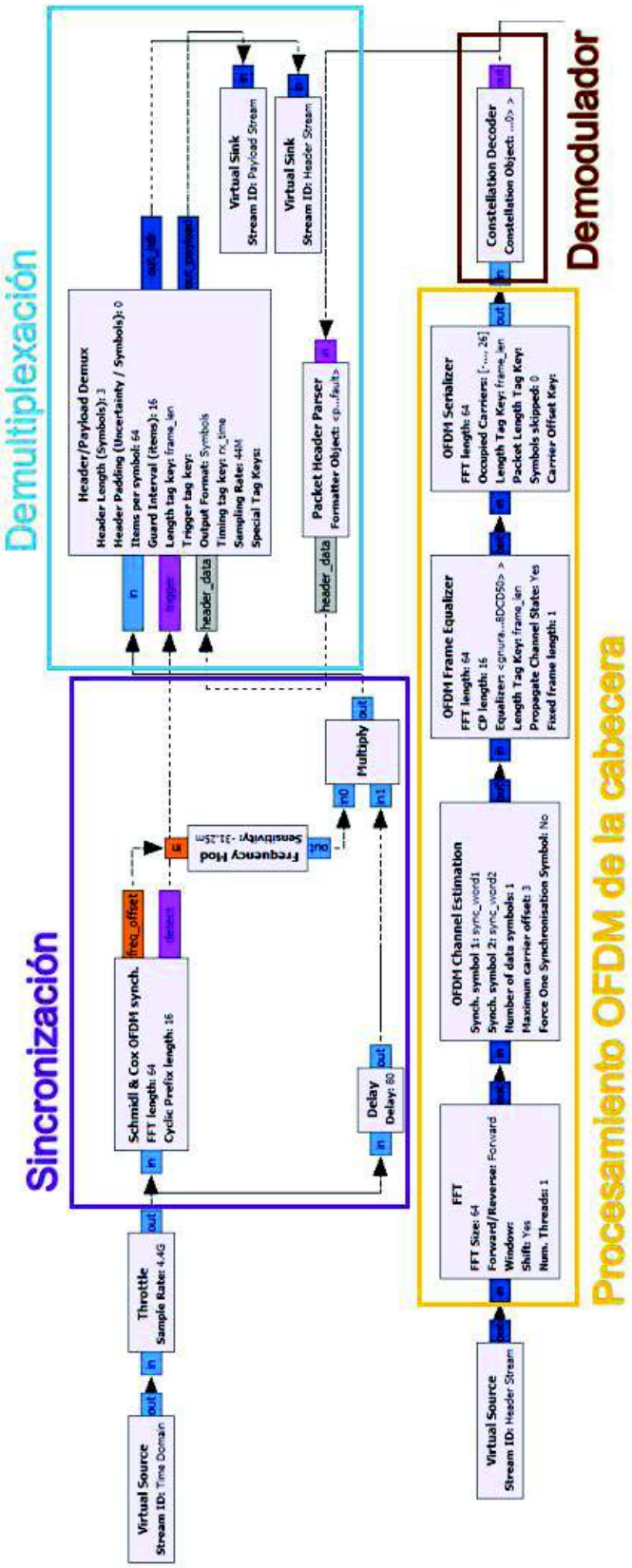

Figura 2. 44 Sincronismo, procesamiento de la cabecera y retroalimentación

La segunda salida del demultiplexor corresponde a la carga útil, a continuación, se describe el procesamiento de la carga útil.

2.3.4 Procesamiento de la Carga Útil

Dentro del procesamiento de la carga útil ese encuentra los bloques OFDM, la demodulación o desmapeo, desempaquetado, verificación de CRC en las etiquetas y recuperación de la información. El procesamiento de los bloques OFDM comienza con la conversión S/P y FFT.

2.3.4.1 FFT y conversión S/P

Después del envío de símbolos de entrenamiento, palabras de sincronismo y el proceso de estimación de canal, el sistema comienza el procesamiento de la carga útil. Este proceso comienza con una conversión S/P y uso de la FFT en el mismo bloque, similar al procesamiento de la cabecera se utiliza el bloque FFT en modo "*forward*" y depende de la variable "fft_len". Se observa este bloque en la figura 2.45.

Figura 2. 45 Bloque FFT para procesamiento de la carga útil

La salida se conecta al bloque ecualizador.

2.3.4.2 Ecualizador

Posterior al bloque de la FFT se conecta un bloque ecualizador para mitigar las anomalías de canal, se utiliza la información sobre la estimación de canal que se obtuvo en el procesamiento de la cabecera realimentado al demultiplexador.

Aunque es un bloque similar al bloque utilizado en la cabecera, la variable de ecualizador es diferente y depende directamente de una variable tipo objeto que llama al ecualizador propio de carga útil de OFDM [2]. El ecualizador se lo observa en la figura 2.46.

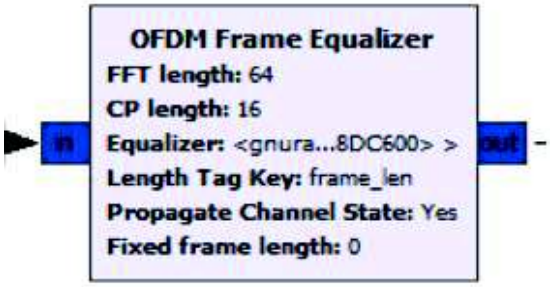

Figura 2. 46 Ecualizador para la carga útil.

La información que entra en el bloque FFT es un flujo en serie que se convierte en varios flujos en paralelo y posteriormente son ecualizados, ahora nuevamente es necesario convertir estos flujos a uno solo en serie.

2.3.4.3 Conversor P/S

El bloque "*OFDM serializer*" es el encargado de convertir los flujos de símbolos a un solo flujo en paralelo tomado la información de cada subportadora de datos y entrega este flujo de símbolos en serie listo para ser demodulado. Se observa este bloque en la figura 2.47.

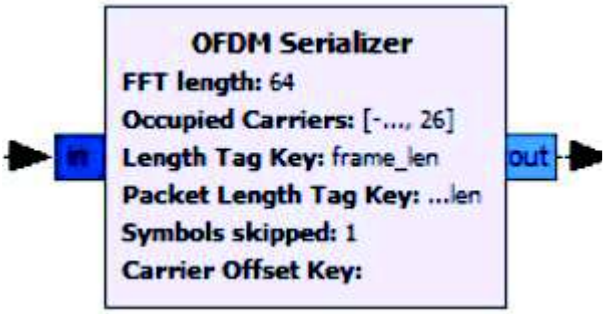

Figura 2. 47 Conversor P/S de la carga útil

Los bloques OFDM del procesamiento de la carga útil conectados, se los observa en la figura 2.48.

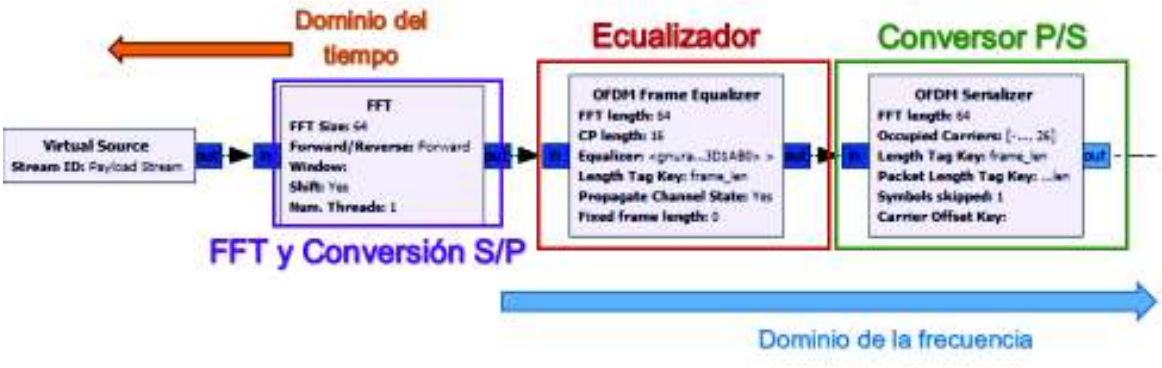

Figura 2. 48 Bloques procesamiento OFDM de la Carga Útil

2.3.4.4 Demodulación

El flujo de símbolos que los bloques OFDM entregan están preparados para ser desmapeados, se utiliza el bloque "*Constellation Decoder*" y se asocia a una variable de tipo objeto llamada "*constellation object*" que asocia la tabla de desmapeo con los simbolos que ingresan. La carga útil utiliza modulación QPSK, por lo tanto, es necesario especificar el tipo de modulación en el bloque "*constellation object*". Se observa estos bloques en la figura 2.49.

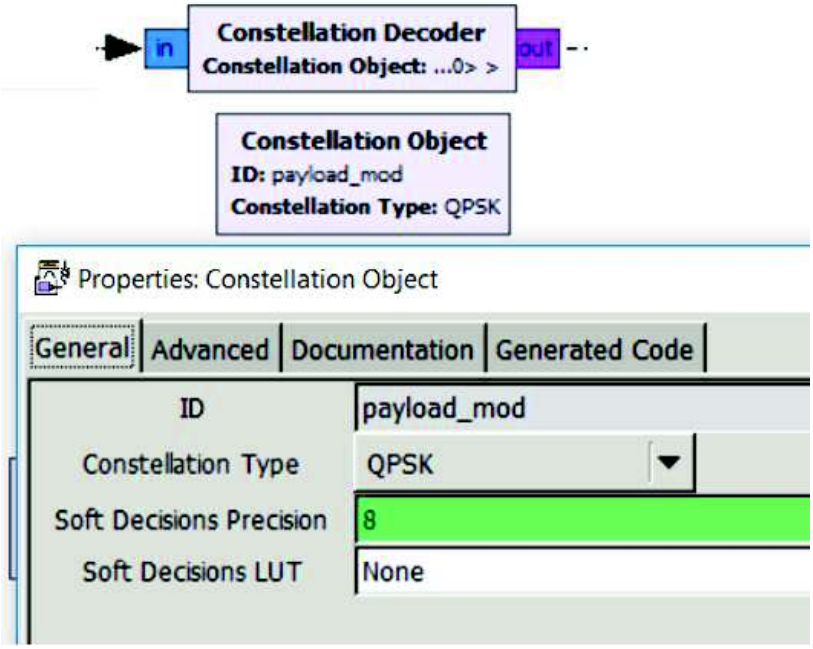

Figura 2. 49 Bloques Demoduladores de la carga Útil

Después del proceso de demodulación de los símbolos, se entrega un flujo de bits en serie empaquetados y etiquetados en el dominio del tiempo.

2.3.4.5 Desempaquetado

En el transmisor se empaquetó o agrupó a los bits en un grupo de 8 bits con 4 subgrupos de 2 bits, cada subgrupo representaba un símbolo demapeado, en el proceso inverso se eliminan los subgrupos y grupos de bits, generando un flujo desempaquetado. Para este proceso se utiliza el mismo bloque "*Repack bits*" especificando la entrada y salida. Este bloque y sus propiedades se observan en la figura 2.50.

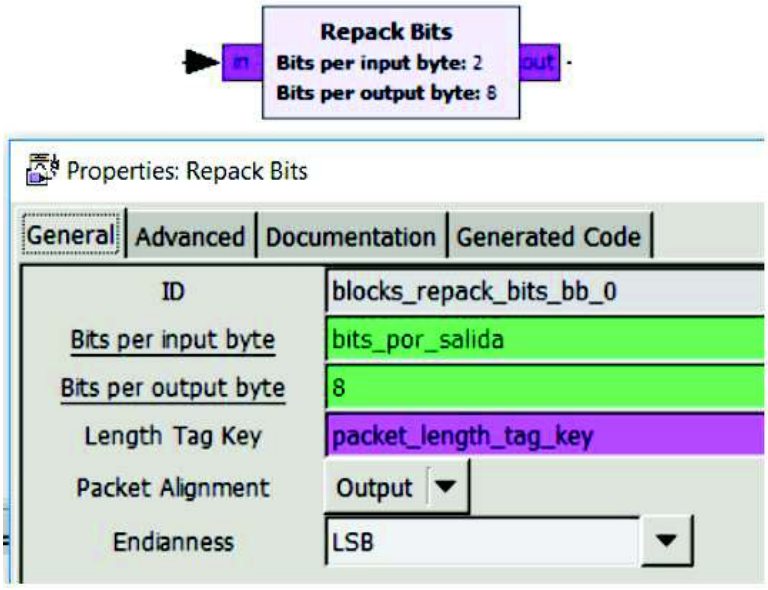

Figura 2. 50 Bloque de desempaquetado de bits

La entrada del bloque de desempaquetado está asociada a la misma variable que fue creada en el transmisor para poder usar una sola variable para ambos procesos. Esta variable a su vez dependía de otras variables que se indicaron en la sección 2.2, en el proceso de empaquetado, se las puede observar en la figura 2.17 de la misma sección.

2.3.4.6 Verificación de CRC32

Esta verificación se la realiza en las etiquetas insertadas en transmisión, para ello se utiliza el mismo bloque "*Stream CRC32*", sin embargo, en recepción el bloque está en el modo inverso, no genera sino revisa el CRC generado en transmisión. Este bloque se observa en la figura 2.51.

Figura 2. 51 Verificación CRC32

Antes de presentar la información ya recuperada, es necesario conectar un boque para convertir el tipo de señal que entrega el bloque CRC32. El bloque encargado de realizar esta conversión es el bloque "*char to float*". La conexión de los bloques de demodulación, desempaquetado, verificación de CRC y conversión de variable se los observa en la figura 2.52.

Figura 2. 52 Demodulación, desempaquetado y verificación de CRC en la carga útil.

La información que entrega el bloque de verificación de CRC es la información recuperada, solamente deberá ser presentada. Dependiendo del tipo de información que haya ingresado, en otras palabras, depende de la fuente de entrada.

El receptor implementado con sus etapas acopladas se puede observar en la figura 2.53.

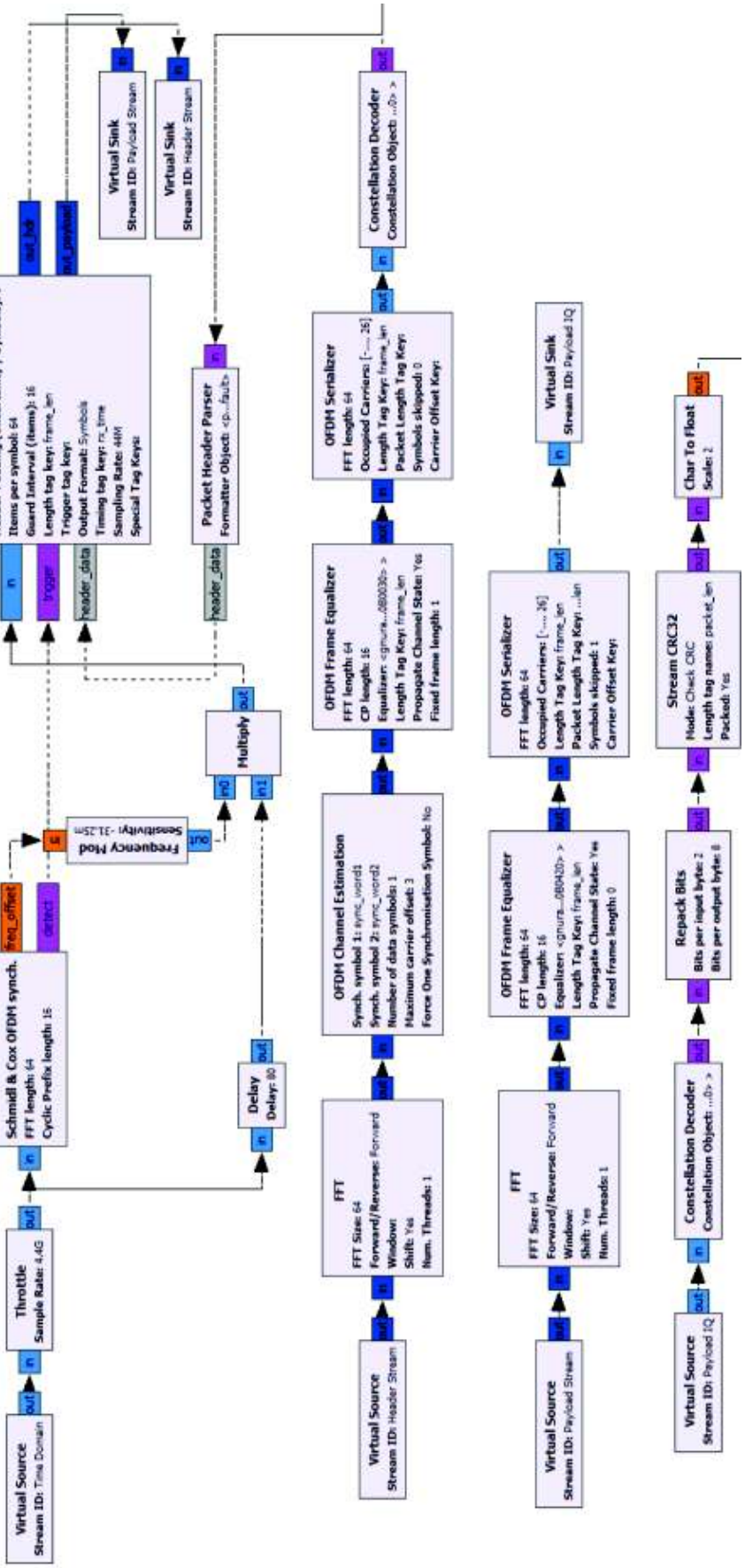

Figura 2. 53 Receptor OFDM implementado en GNU Radio

2.4 Acoplamiento entre etapas

En esta sección se indica el proceso de adecuación y acoplamiento de las etapas descritas en las secciones de digitalización, transmisor y receptor OFDM y los archivos "*.grc*" donde se encuentran. El acoplamiento es necesario tanto para las simulaciones como las pruebas, además permite optimizar el código que se genera. Al final de esta etapa se espera tener 2 archivos "*.grc*", un archivo en el cual se observe todas las etapas de transmisión y otro para las etapas en recepción, estos mismo archivos se pueden usar tanto para simulación como transmisión con los equipos USRP.

2.4.1 Acoplamiento Digitalización-transmisor

Con este acoplamiento se espera combinar la etapa de digitalización con la etapa del transmisor. En la sección de digitalización de la señal el cuantificador entrega información a manera de bits, como en el transmisor la información debe ingresar de manera de bits, existe compatibilidad entre estas etapas al conectar, sin embargo, hay que tomar en cuenta variables como:

- Frecuencia de muestreo
- Variables de cuantificación y modulación
- Librerías

GNU Radio por defecto genera una frecuencia de muestreo a la que los bloques de los programas se ejecutarán, por esta razón ambos archivos "grc" funcionan a frecuencias de muestreo diferentes. Para solucionar este inconveniente se toma como base la frecuencia de la señal que ingresa, es necesario cumplir con el criterio de Nyquist para esta frecuencia, por ello se especifica la variable "*samp rate*" como la frecuencia de muestreo a la cual el programa se ejecuta. Se crea la variable "*samp_rate_audio*" para ser la frecuencia de muestro a la cual la señal de audio se reproducirá.

Se separan estos valores para poder variar ciertos bloques necesarios y no en todos los bloques del sistema. El valor de la frecuencia de audio depende del archivo que ingresa, por este motivo se usan archivos con frecuencias de 44.1kHz, esto significa que la variable "samp_rate_audio" tendrá la frecuencia de 44.1kHz y la variable general "*samp_rate*" deberá cumplir con el criterio de Nyquist, se establece la frecuencia "*samp_rate*" con 441Khz como se indicó en la sección 2.1. Se observan estas variables en la figura 2.54.

Figura 2. 54 Variables de frecuencia

Las variables que se emplean en la digitalización y cuantificación no dependen de factores del transmisor, por esta razón no es necesario asociar estas variables sino solamente copiarlas. Lo que se debe tomar en cuenta es que algunas de estas variables dependían de la frecuencia de muestreo, razón por lo cual se debe buscar y actualizar el valor de la frecuencia de muestreo adecuada.

Las variables de la modulación no dependen de la frecuencia de muestreo, pero tienen dependencia entre ellas como se describió en la sección 2.2.3. Todas las variables se las puede observar en la figura 2.55.

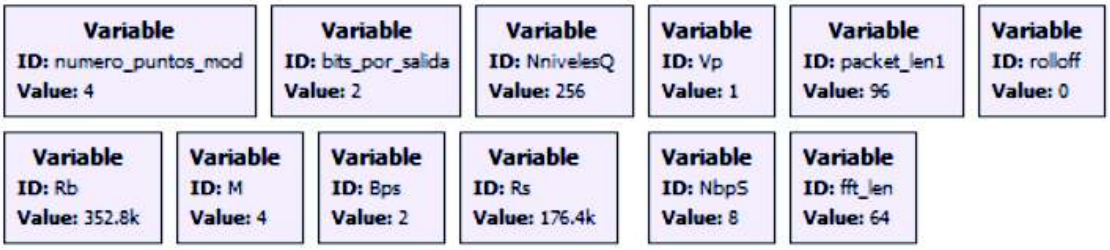

Figura 2. 55 Variables de Digitalización y Modulación

Para las librerías, es necesario importar todas las que se usan en ambas etapas, esto permite el funcionamiento del sistema en conjunto. Muchas librerías que se usan en ambas etapas son las mismas, así que no es necesario copiarlas dos veces. Las librerías usadas se observan en la figura 2.56.

Figura 2. 56 Librerías del Transmisor

En conjunto todas las variables para el prototipo de transmisor OFDM incluyendo las variables de procesamiento OFDM se las puede observar en la figura 2.57.

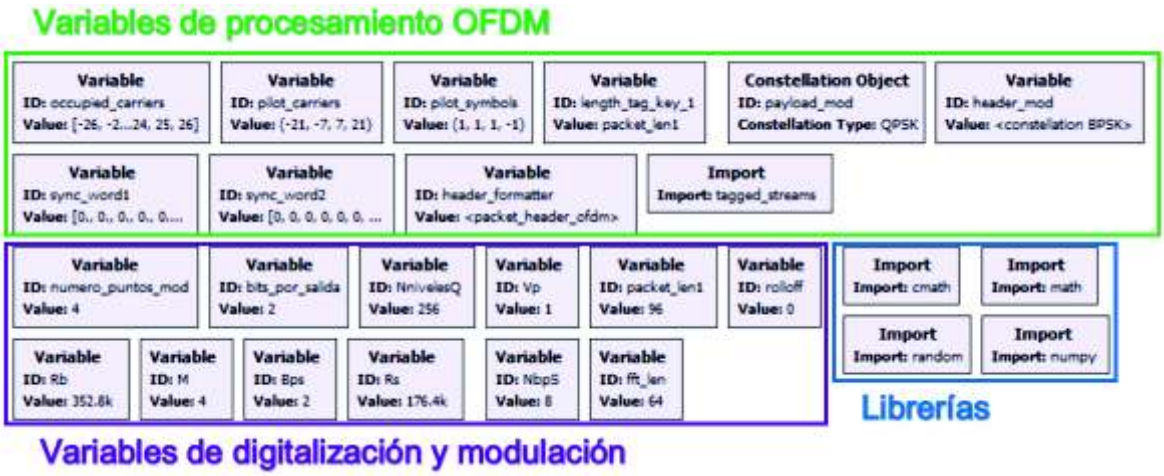

Figura 2. 57 Variables Transmisor OFDM

El Acoplamiento del prototipo de transmisor OFDM con digitalización, se lo observa en la figura 2.58.

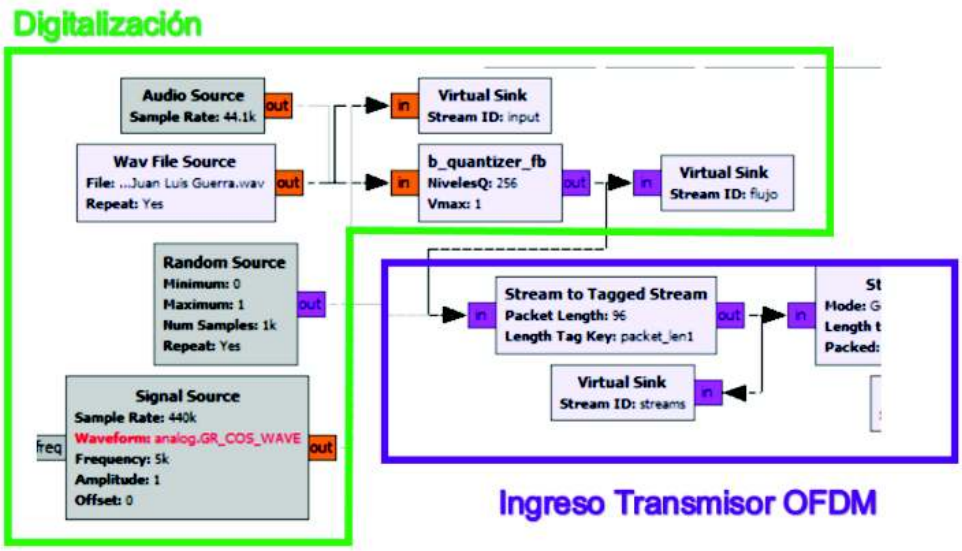

Figura 2. 58 Acoplamiento del transmisor OFDM con digitalización

2.4.2 Acoplamiento Receptor – Recuperación de la señal

En acoplamiento se lo realiza entre receptor y la recuperación de la información de digitalización. A la salida del receptor en el bloque CRC32 se entrega la información como un flujo de bits en una variable del tipo "*byte*", para recuperar la información los datos se deben entregar con una variable del tipo "*float*" o flotante, por este motivo este tipo de información no puede directamente reproducirse pues los tipos de variables no son

compatibles. Es necesario incorporar un bloque de conversión de variable del tipo "*char to float*", se observa este bloque en la figura 2.59.

Figura 2. 59 Bloque convertidor de variable "*Char to Float*"

Sin embargo, solamente cambiar el tipo de variable no permite reproducir directamente la señal de audio. Este flujo debe ser normalizado y asociado a las variables de cuantización o digitalización del transmisor. Este proceso se realiza multiplicando por un valor normalizado, se puede observar en la figura 2.60.

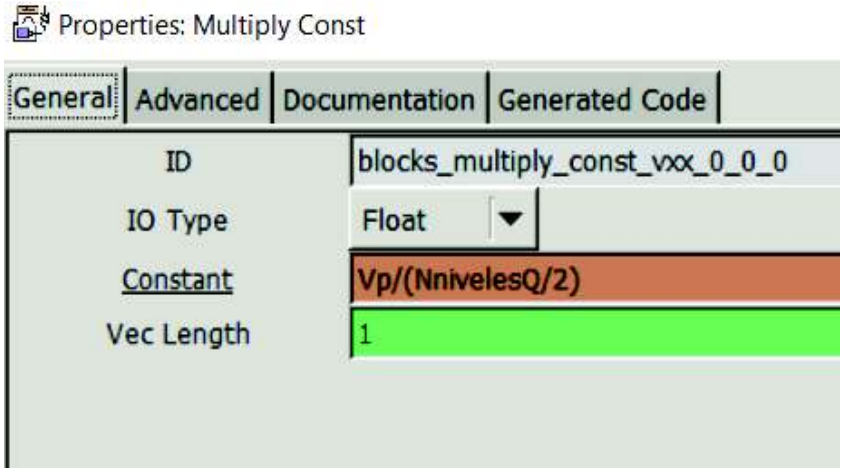

Figura 2. 60 Bloque multiplicación de constante para normalizar el flujo

Las variables "Vp" y "*NnivelesQ*", permiten normalizar el flujo y lo asocian con los niveles de cuantificación. Como se usan las mismas variables de digitalización del transmisor, también se toma en cuenta la frecuencia de muestreo a la cual funcionan ambos archivos. Además, las librerías usadas son las mismas, así que no es necesario importar librerías adicionales.

Al igual que en transmisor, se crean dos variables para la frecuencia de muestreo, "*samp_rate*" y "*samp_rate_audio*", estas variables tendrán los mismos valores y realizan las mismas funciones.

La conexión de los bloques del receptor acoplado con la digitalización se lo observa en la figura 2.61.

Figura 2. 61 Receptor OFDM con recuperación de la información

2.5 Configuración Equipos USRP

Al ejecutar los programas generados en computadora, los equipos USRP se encargan de comunicar la información del software con los elementos de hardware para implementar una transmisión real. Los bloques que permiten esta conexión entre los equipos y los programas del transmisor y receptor son los bloques UHD. Se observa estos bloques en la figura 2.62.

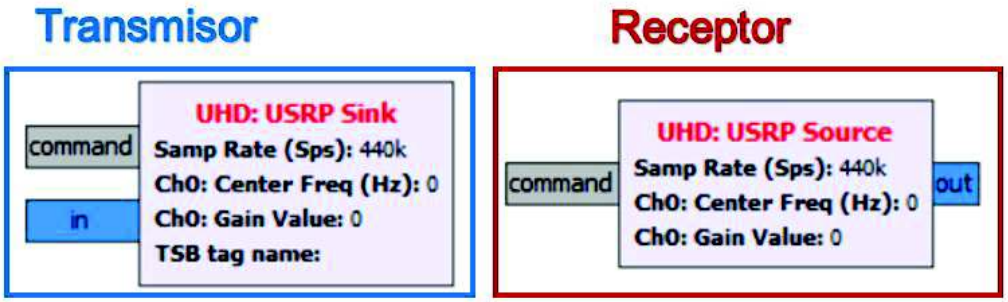

Figura 2. 62 Bloques UHD-USRP de configuración

Estos bloques se conectan en el transmisor y receptor respectivamente, en el transmisor el bloque "*UHD: USRP Sink*" se conecta a la salida del procesamiento. En el receptor se conecta el bloque "*UHD: USRP Source*" al comienzo del procesamiento.

Al conectar estos bloques en los programas de transmisor y receptor se establece la comunicación con los equipos, al levantar esta comunicación y correr los programas las simulaciones se convierten en transmisiones en tiempo real. Estas conexiones se observan en las figuras 2.63 y 2.64 respectivamente.

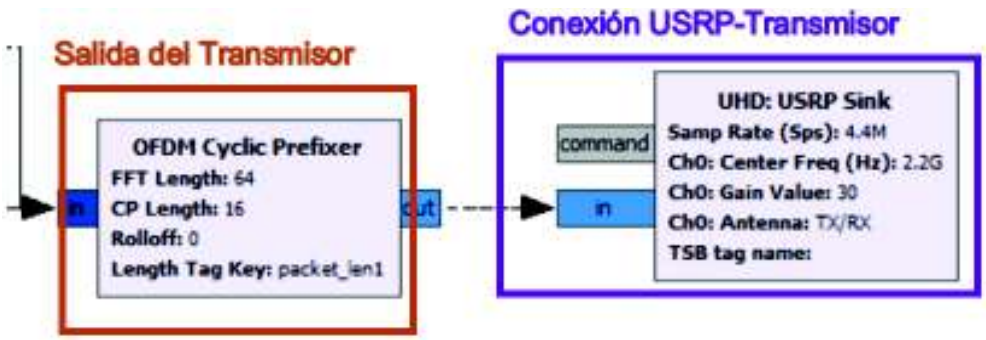

Figura 2. 63 Conexión del transmisor OFDM con equipo USRP

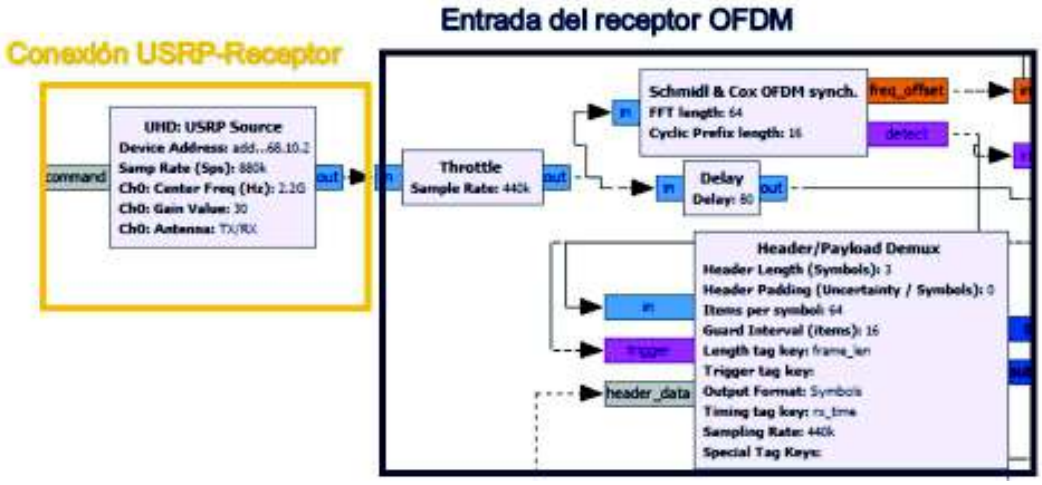

Figura 2. 64 Conexión del receptor OFDM con equipo USRP

2.5.1 Configuración del equipo USRP en el transmisor y Receptor

Para la configuración del equipo USRP en el transmisor es necesario establecer los parámetros como: conectividad con el ordenador, frecuencia de transmisión, frecuencia de muestreo del USRP, ganancia de la antena, entre otros. En esta sección describen cada uno de estos parámetros.

Conectividad

En primer lugar, es necesario conectar el equipo USRP con el ordenador con un cable *"Gigabit Ethernet*", se utiliza un cable categoría 6 o categoría 6a. Al conectar al ordenador se debe establecer una comunicación interna entre el ordenador y el equipo USRP mediante una red local.

Es necesario configurar el puerto "*Gigabit Ethernet*" del ordenador como "*Gateway*". Se observa en la figura 2.65.

Figura 2. 65 Configuración Dirección IP GW del ordenador

Después de la configuración de la dirección del puerto del ordenador, es necesario configurar la dirección del puerto del equipo USRP, esa configuración se la realiza mediante el bloque UHD en GNU Radio. Esta configuración se la observa en la figura 2.66.

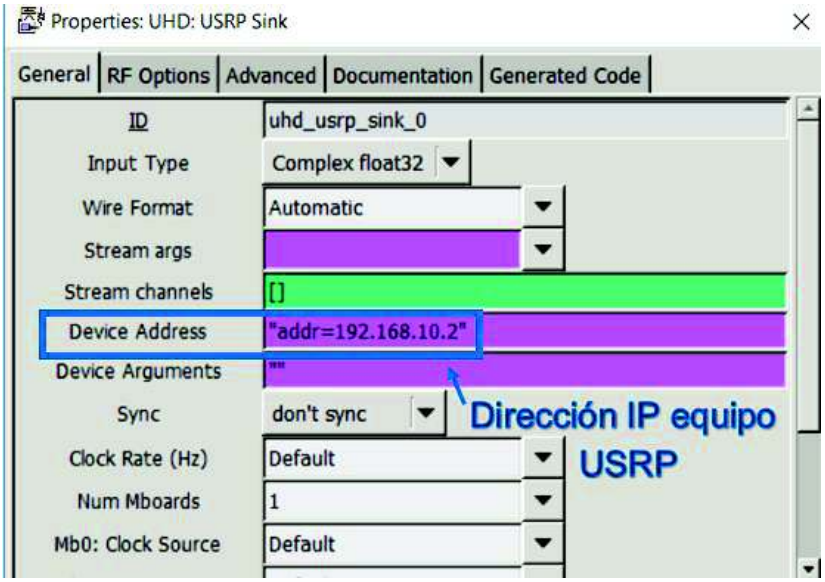

Figura 2. 66 Dirección IP del equipo USRP

Frecuencia de Transmisión

En GNU Radio la información se encuentra en banda base lista para ser transmitida, para poder transportar esta información debemos seleccionar una banda de transmisión y una frecuencia para la portadora central.

Consideraciones:

- Las bandas ISM son 2.4GHz y 5.1GHz.
- La frecuencia de operación de los equipos USRP es de 50MHz a 2.2GHz.
- Se implementan las características del estándar IEEE 802.11a que se mencionó en la sección 2.2, la frecuencia de operación de este estándar es a 5.1GHz.

Tomando en cuenta que el equipo USRP tiene limitación en su frecuencia de operación no es posible implementar la transmisión en ninguna banda ISM. Se selecciona la banda de 2.1GHz para transmitir sin trabajar en la frecuencia máxima de operación del equipo.

Las mismas consideraciones se toman en el programa del prototipo en recepción, es necesario sintonizar los equipos para poder establecer la comunicación.

Antena

Para seleccionar una antena de transmisión se debe tomar en cuenta la frecuencia de transmisión con la que se establece la comunicación. Se dispone de tres opciones de antenas:

- **VERT 400:** antena multibanda, antena tribanda vertical. 144MHz, 400MHZ, 1200MHz
- **VERT 900:** antena de banda doble o "*dual band*", 824-960 MHz, 1710-1990 MHz.
- **VERT 2450:** antena vertical doble banda "dual band", 2.4-2.5 GHz y 4.9 -5.9 GHz.

Estas antenas se las puede observar en la figura 2.67.

Figura 2. 67 Tipos de Antenas Disponibles

Para el funcionamiento del prototipo se selecciona la antena VERT 900, esta antena es seleccionada porque su frecuencia de operación es la más cercana a la frecuencia de transmisión seleccionada para el prototipo.

El equipo USRP tiene dos puertos coaxiales SMA, las antenas se conectan en estos puertos coaxiales, pero se debe seleccionar el puerto dependiendo del uso que se da a la antena. El puerto 1 TX1/RX1 funciona como puerto tanto de transmisión como recepción, y el puerto RX2 solamente funciona como puerto de recepción.

La conexión de la antena con el equipo se puede observar en la figura 2.68.

Figura 2. 68 Antena VERT 900 conectada al equipo

Ganancia

La ganancia del equipo se debe fijar de acuerdo a sus características. Las especificaciones de los equipos USRP soportan de 0 a 31.5 dB, para controlar la ganancia del equipo no se especifica un valor estático, sino uno variable con ayuda del bloque "*qt gui range*".

El bloque "*qt gui range*" crea variables que pueden cambiar su valor en tiempo real, es necesario especificar un valor inicial, máximo, un valor predeterminado y un valor de pasos entre valores. La variable que se crea se denomina "*gain*". Estas propiedades se las puede observar en la figura 2.69.

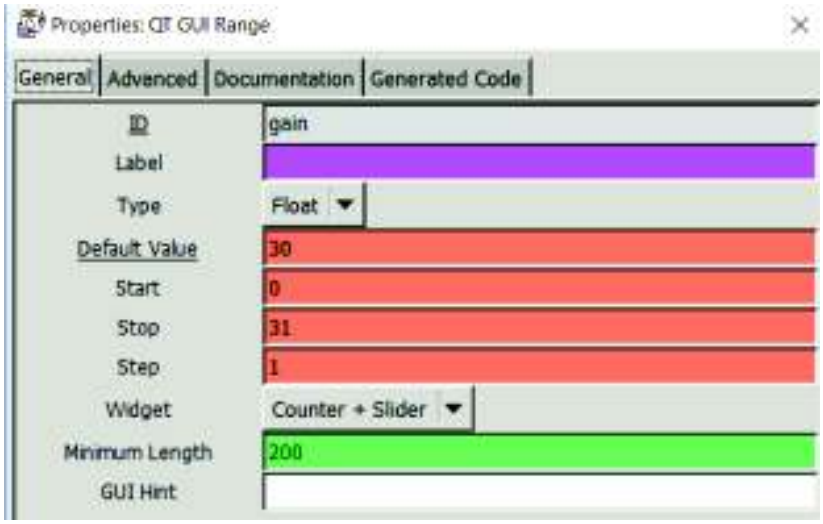

Figura 2. 69 Propiedades bloque *"QT GUI Range*" para controlar la ganancia

Frecuencia de muestreo del equipo USRP

La frecuencia de muestreo del equipo USRP es la frecuencia a la cual la tarjeta madre toma las muestras por segundo de la señal en banda base que se genera en el software, esta frecuencia de muestreo se define por "*Sps*" que significa "*Samples per second*" o muestras por segundo. Esta frecuencia tiene como límite el procesamiento de la tarjeta FPGA.

Es necesario tomar en cuenta los valores que puede tomar la frecuencia de muestreo de la tarjeta FPGA, se debe procurar no exceder las frecuencias de muestreo. Un valor alto de frecuencia de muestreo incurre en un alto nivel de consumo de recursos en procesamiento, tanto de la FPGA como del procesador del ordenador y la memora ram. Sin contar con el costo de procesamiento para la observación de cada etapa del sistema.

Por esta razón el valor de la "*Sps*" en los equipos FPGA será definido por: 10*samp_rate, es decir, 10 veces el valor de la frecuencia de muestreo. En banda base esta frecuencia está dentro del rango de funcionamiento de los equipos y no consume demasiados

recursos al momento del procesamiento. Cabe recordar que los conversores A/D y D/A de los USRP ya están configurados dentro del equipo y no es necesario configurarlos.

Se puede observar este valor configurado en el equipo USRP en la figura 2.70.

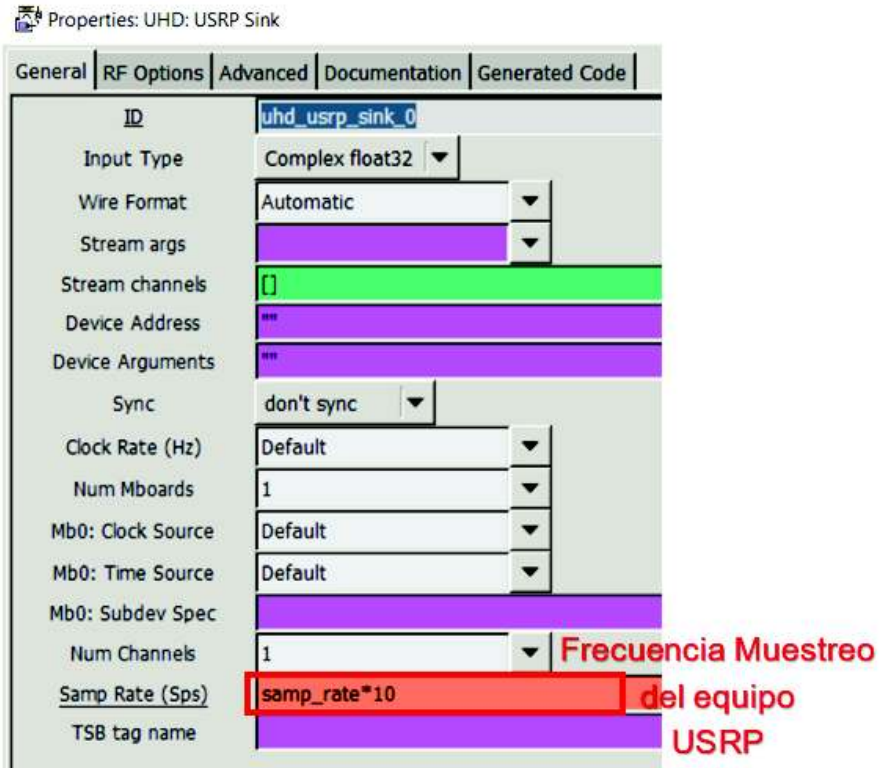

Figura 2. 70 Frecuencia de Muestro del Equipo USRP

En las propiedades del bloque "*UHD USRP Sink*" se puede observar una pestaña para configurar las opciones de Radiofrecuencia que han sido descritas, la configuración de todos estos valores se los puede observar en la figura 2.71.

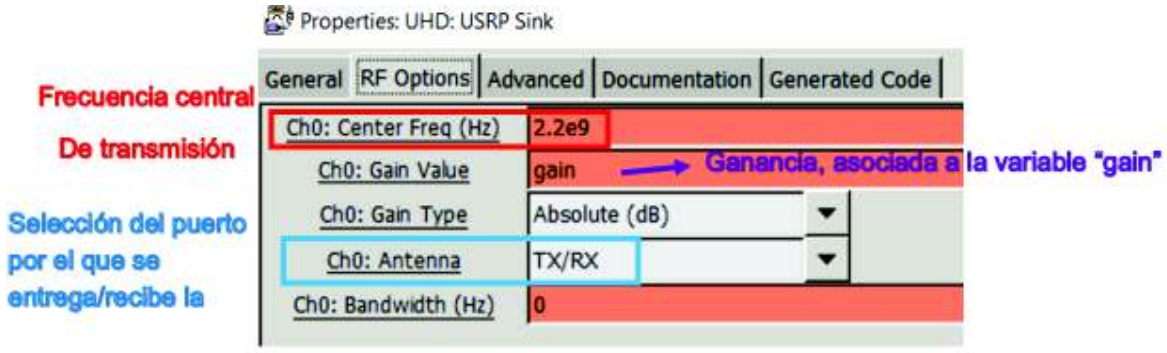

Figura 2. 71 Variables de Radiofrecuencia del equipo USRP

En recepción el bloque "*UHD USRP Source*" tiene las mismas opciones, asi que se debe configurar todo de la misma manera.

2.6 Configuración de archivos complementarios para las pruebas

Los archivos que se crearon para las implementaciones de los prototipos de transmisor – receptor tienen bloques equivalentes en GNU Radio, así que para probar su funcionamiento es necesario implementar otros dos archivos ".*grc*" para posteriormente poder verificar la compatibilidad y funcionamiento.

2.6.1 Configuración del Transmisor OFDM modelo en GNU Radio

El bloque que GNU Radio presenta en sus librerías se denomina: "*OFDM Transmitter*", el cual tiene intrínsecamente incorporados los bloques que se utilizan en el prototipo de transmisor OFDM, es necesario configurar este bloque con los mismos valores mencionados y explicados en la sección 2.2. Estos bloques realizan las mismas funciones que los prototipos implementados sin embargo no es posible observar su procesamiento interno. Este bloque y sus opciones se los puede observar en la figura 2.72.

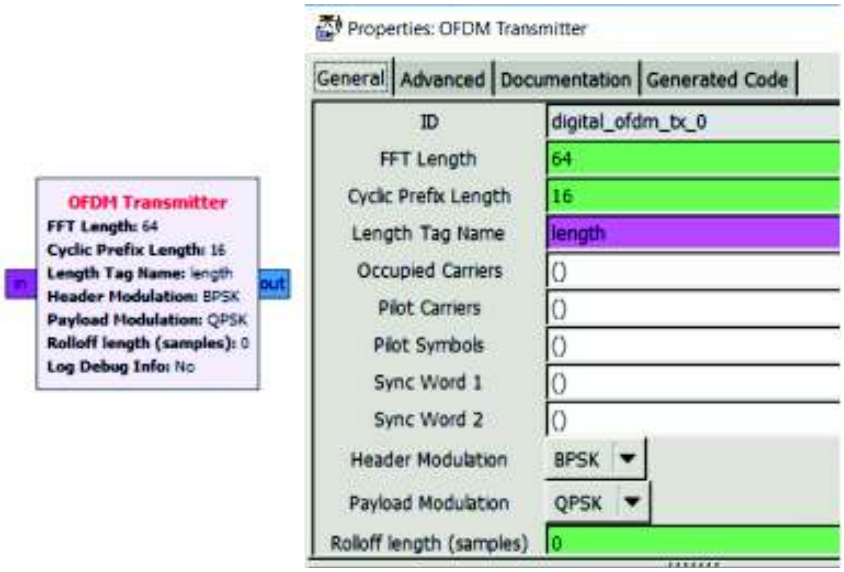

Figura 2. 72 Bloque "*OFDM Transmitter*" y sus propiedades

Es necesario tomar en cuenta que este transmisor ocupa las variables que fueron creadas anteriormente, de esta manera se configuran las variables de portadoras piloto, datos, palabras de sincronismo y tipos de modulaciones.

La implementación del transmisor OFDM con el bloque "*OFDM Transmitter"* se lo observa en la figura 2.73.

Figura 2. 73 Transmisor OFDM con el bloque "*OFDM Transmitter*"

2.6.2 Configuración del Receptor OFDM modelo en GNU Radio

El bloque de receptor OFDM modelo en GNU radio se llama: "*OFDM Receiver*", tiene internamente integrados los bloques detallados en la sección 2.3 este bloque utiliza las variables creadas y descritas en las secciones 2.2 y 2.3, este bloque y sus propiedades se observan en la figura 2.74.

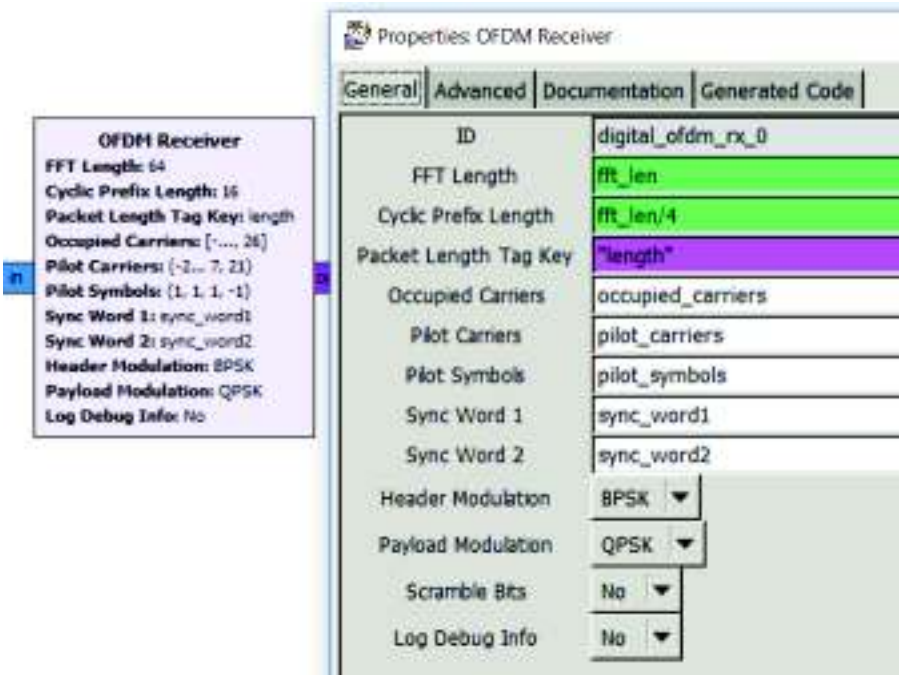

Figura 2. 74 Bloque "*OFDM receiver*" y sus propiedades

Es posible observar las configuraciones y como se asocian los valores a las variables creadas en secciones anteriores, este bloque y su conexión e implementación se lo observa en la figura 2.75.

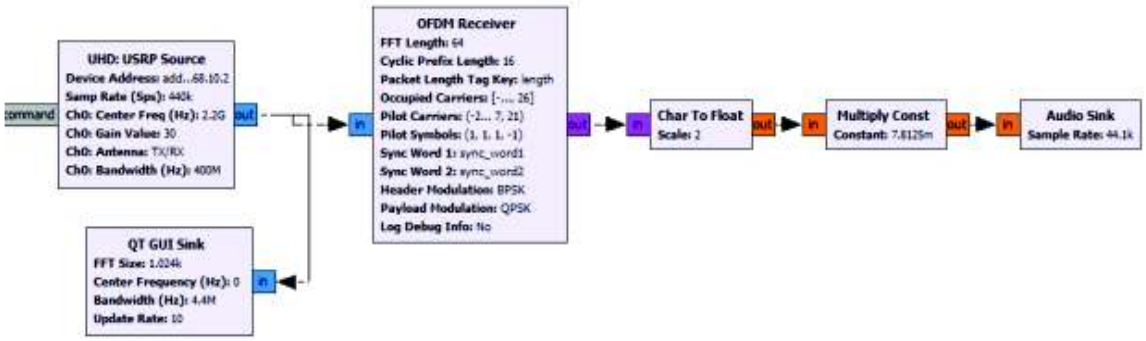

Figura 2. 75 Receptor OFDM con el bloque "*OFDM receiver*"

3 RESULTADOS Y DISCUSIÓN

En esta sección se presentan las pruebas de funcionamiento, estas pruebas se realizan para verificar el funcionamiento y la compatibilidad de los prototipos. Para comprobar el funcionamiento de los prototipos en primer lugar se realizan pruebas mediante simulaciones en GNU Radio en un mismo computador. Las simulaciones buscan verificar que no existan problemas en la configuración e implementación de los programas generados en los archivos ".grc", en estos archivos se encuentran los programas de transmisor y receptor.

Las simulaciones a su vez permiten verificar la compatibilidad entre los archivos transmisor y receptor implementados que muestran el procesamiento OFDM por etapas con los archivos transmisor y receptor modelos, mismos que GNU Radio proporciona en su biblioteca y tienen el procesamiento OFDM como un proceso cerrado. Tras comprobar la compatibilidad y funcionamiento de los programas desarrollados en GNU Radio se procede a conectar los equipos USRP a las computadoras para comenzar con las pruebas de transmisión en tiempo real.

Las pruebas de transmisión en tiempo real de los prototipos transmisor y receptor tienen como finalidad comprobar la conexión y la configuración entre los equipos USRP y los programas desarrollados, cabe recalcar que se utilizan los mismos archivos que en las simulaciones. Sin embargo, se habilita la conexión entre los programas con los equipos USRP.

Al habilitar la comunicación entre los programas generados y los equipos USRP las simulaciones se convierten en un sistema de comunicaciones real. Este procedimiento permite levantar un sistema de software definido por radio, por lo tanto, a la vez que el programa es ejecutado se levanta un sistema de comunicaciones que es capaz de mostrar el procesamiento en tiempo real por etapas de la capa física.

Por otra parte, también se realizan pruebas de comunicación en tiempo real del sistema transmisor – receptor conectados a los equipos USRP, a través de estas pruebas se busca verificar la compatibilidad de los equipos USRP y los archivos generados en GNU Radio, comprobando a su vez la compatibilidad entre transmisores, receptores y su correcta ejecución.

Las pruebas de transmisión en tiempo real se realizan en un ambiente controlado, un cuarto cerrado o "*in-door*", en este cuarto se realizan 2 pruebas, la primera prueba se realiza para comprobar una correcta configuración entre los equipos USRP y las computadoras, la configuración abarca conectividad entre el equipo USRP y el computador, drivers

actualizados y verificación de los parámetros RF. Los USRP transmisor y receptor respectivamente se encuentran conectados entre sí a través de un medio guiado. La segunda prueba de transmisión se realiza usando al aire como un medio no guiado, para esto se transmite a 3 y 6 metros de distancia entre transmisor y receptor. Estas pruebas muestran el funcionamiento del sistema transmisor – receptor implementados, además verifican que el sistema de comunicación se levante y muestre un procesamiento por etapas en la capa física de OFDM en tiempo real.

Para las pruebas de funcionamiento se utiliza un archivo de audio "*.wav*", también se usan dos esquemas para transmisor y dos para receptor, mismos que fueron implementados en GNU Radio y se encuentran en archivos "*.grc*", estos archivos se describen a continuación:

- Transmisor OFDM implementado
- Receptor OFDM implementado
- Transmisor OFDM modelo
- Receptor OFDM modelo

3.1 Simulaciones

Como se describió previamente estas pruebas permiten verificar la compatibilidad, el funcionamiento y correcta configuración de los archivos generados (".grc") en GNU Radio. Para comprobar este funcionamiento se realizan cuatro pruebas, la primera es la prueba que verifica la compatibilidad entre transmisor modelo y receptor modelo, la segunda prueba verifica compatibilidad entre el transmisor implementado y el receptor modelo, la tercera prueba permite verificar la compatibilidad entre el transmisor modelo y el receptor implementado. Finalmente, se procede a verificar la compatibilidad y funcionamiento de los prototipos transmisor y receptor implementados.

3.1.1 Prueba de compatibilidad entre el Transmisor modelo y el receptor modelo

Esta prueba permite observar el comportamiento de los bloques modelos de GNU radio (transmisor OFDM y receptor OFDM), comprobar su correcta configuración, verificar si las variables creadas se encuentran bien configuradas y finalmente permite observar a la etapa de digitalización de la señal conectada a los bloques modelo. Estos bloques son como cajas cerradas en las cuales no es posible observar el procesamiento interno, pero es

posible ver la señal resultado del procesamiento OFDM, también permite comprobar si las variables creadas como: palabras de sincronismo, símbolos piloto, portadoras de datos, portadoras piloto, entre otras. Se encuentran bien configuradas y son totalmente funcionales. Al ejecutar esta simulación se observan los siguientes resultados:

La señal de audio que ingresa al sistema se la puede observar en la figura 3.1

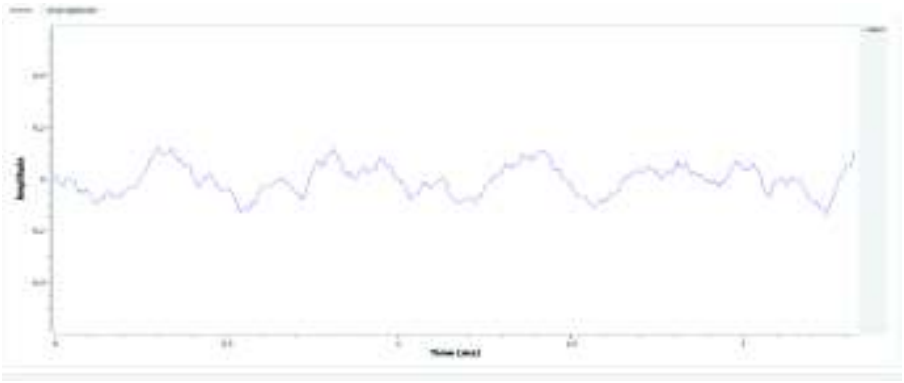

Figura 3. 1 Señal de audio sin procesar

Posteriormente, se observa la digitalización del archivo de audio, este proceso se observa en la figura 3.2.

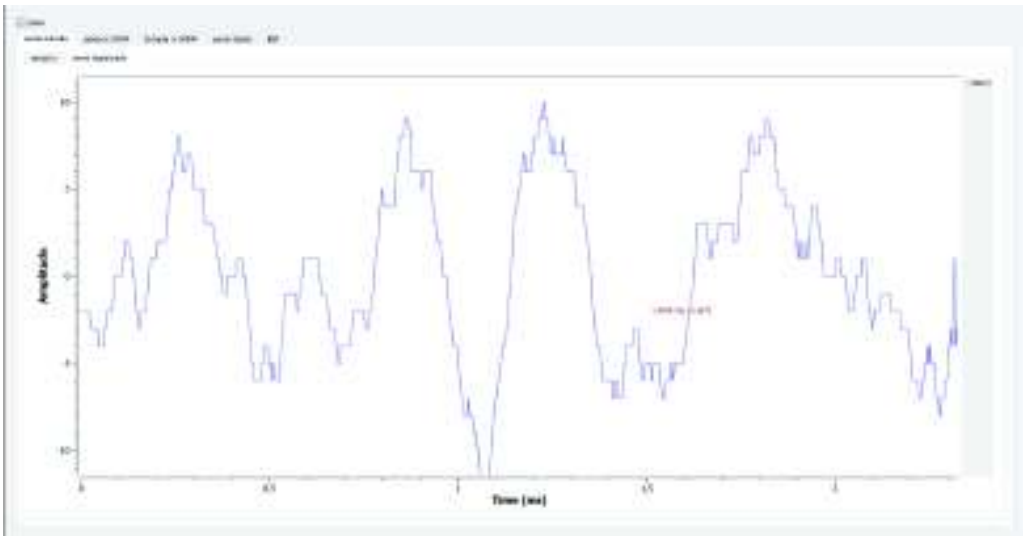

Figura 3. 2 Digitalización de la señal

La digitalización entrega la señal al bloque transmisor OFDM. Los bloques modelo de GNU Radio no permiten observar el proceso interno de un transmisor OFDM, por lo tanto, la señal que sale del "*OFDM Transmitter*" ya está procesada, se observa en la figura 3.3.

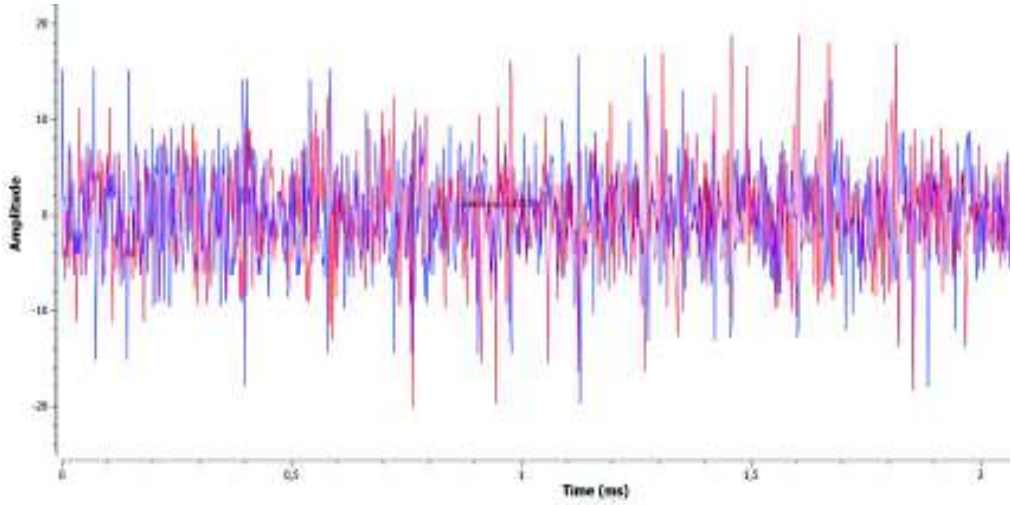

Figura 3. 3 Señal OFDM en el dominio del tiempo

La señal que el transmisor entrega está lista para viajar en el canal, también se puede observar en el dominio del tiempo los picos (estos picos afectan en el PAPR de OFDM). Por otra parte, el espectro de esta señal se puede observar en la figura 3.4.

Figura 3. 4 Señal OFDM en frecuencia

Es posible observar la señal con un espectro ensanchado, este espectro es característico de OFDM, se puede observar la transmisión multiportadora como una transmisión de banda ancha. Además, aumentando el tamaño de las muestras FFT en los instrumentos de medición se puede observar el espectro de las subportadoras piloto.

La señal que ingresa al receptor OFDM se observa en la figura 3.5. Para esta simulación el canal solamente atenúa la señal transmitida con a un factor de “0.05”, este es un valor al azar asignado por el usuario pues el objetivo de esta simulación no es estudiar el canal, sino verificar la compatibilidad de los archivos y su funcionamiento.

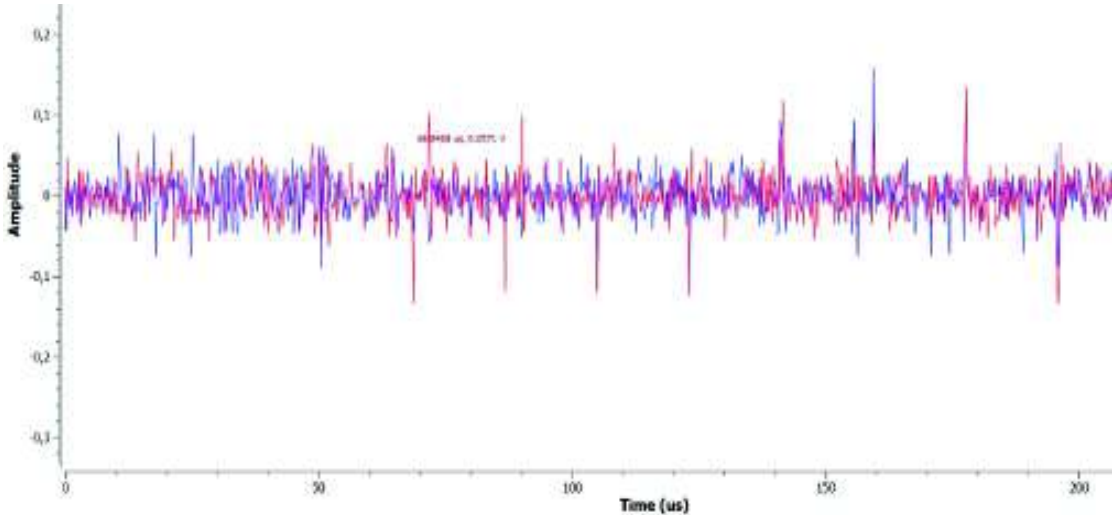

Figura 3. 5 Señal OFDM Recibida

La señal recibida que se aprecia en la figura 3.5 es la señal atenuada por el canal, el espectro de esta señal se observa en la figura 3.6.

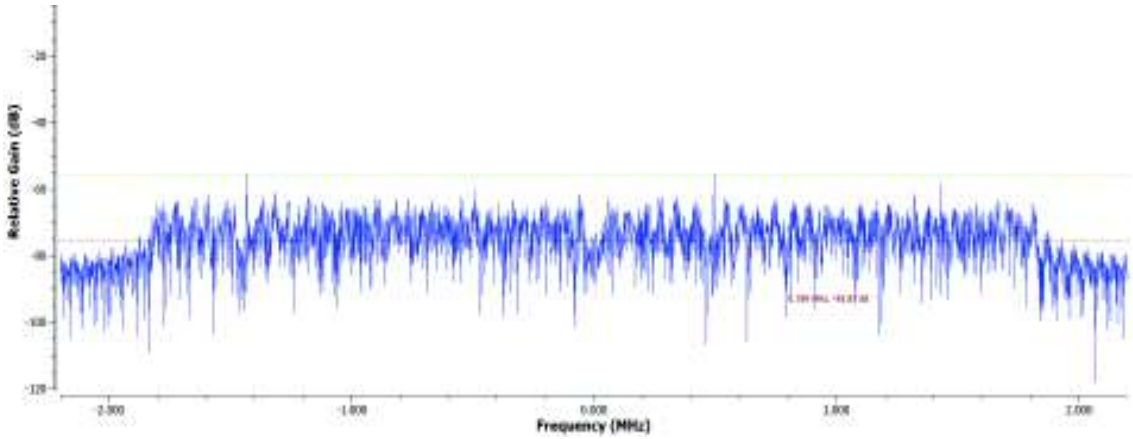

Figura 3. 6 Espectro de la señal OFDM recibida**.**

La atenuación que afecta al canal se observa por el nivel de potencia, sin embargo, es posible apreciar la señal y su ancho de banda.

Pese a que es una simulación, GNU radio muestra los resultados en tiempo real, razón por lo cual es posible observar los paquetes que ingresaron al bloque "*OFDM receiver*", así mismo este bloque se encarga automáticamente de sincronizar la señal y detectar el inicio y fin de cada símbolo OFDM. Esto se observa en la figura 3.7.

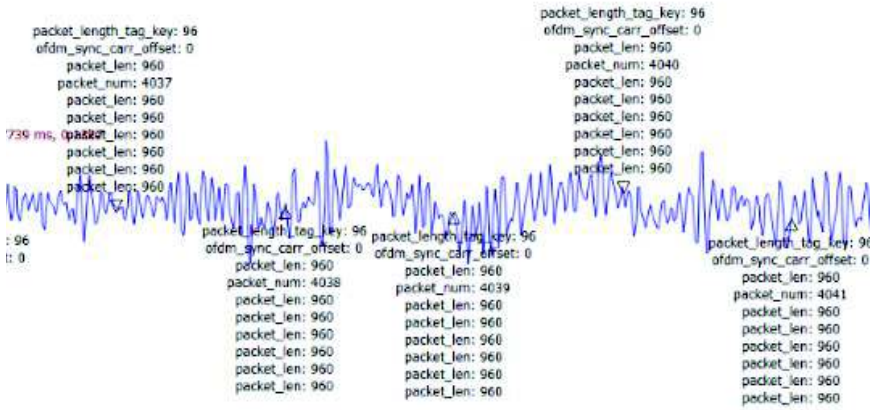

Figura 3. 7 Señal recuperada en el receptor

A la vez que se observa la señal recuperada también se aprecia los paquetes OFDM, y los inicios de cada uno. Esta prueba muestra un funcionamiento completo y una correcta configuración de los bloques "*OFDM Transmitter*" y "*OFDM receiver*". Además, las variables creadas funcionan correctamente, no presentan problemas de compatibilidad.

3.1.2 Prueba de compatibilidad entre el transmisor implementado y el receptor modelo

Esta prueba además de compatibilidad verifica la conexión y configuración de los bloques que conforman el transmisor OFDM implementado, este prototipo muestra cada etapa de procesamiento de OFDM en transmisión, es decir, el procesamiento de la capa física. Además, permite de verificar la correcta configuración de las variables.

Este prototipo al ser ejecutado recibe información a manera de archivo de audio y entrega una señal OFDM en el dominio del tiempo, esta señal viaja por un canal simulado con atenuación. En recepción los bloques del receptor entregan la señal recuperada en forma de señal audible, permitiendo escuchar el audio y observar en el tiempo esta señal.

La señal de audio que ingresa al transmisor se puede observar en la figura 3.8.

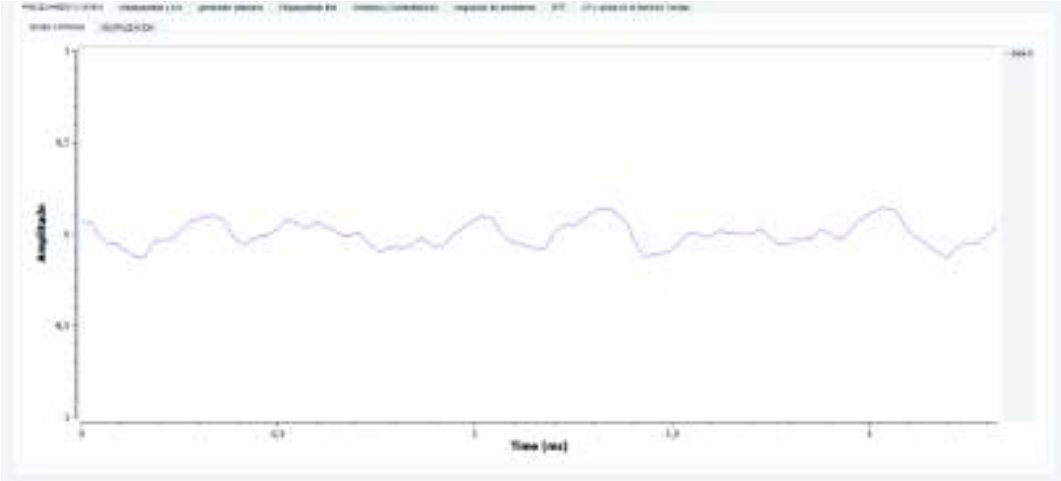

Figura 3. 8 Señal de entrada del transmisor OFDM

La digitalización de la señal que ingresa al sistema se observa en la figura 3.9.

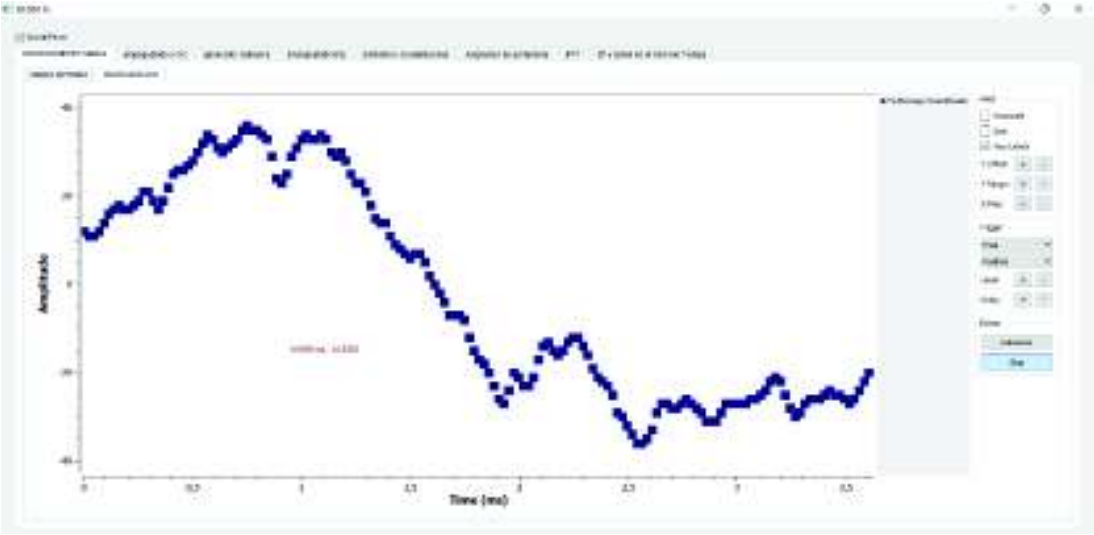

Figura 3. 9 Digitalización de la señal de entrada

El flujo que se genera a la salida del cuantificador debe ser etiquetado con la longitud de cada paquete. Para ello se agrega una etiqueta, esta etiqueta delimita la duración de cada paquete, y se la puede observar en la figura 3.10.

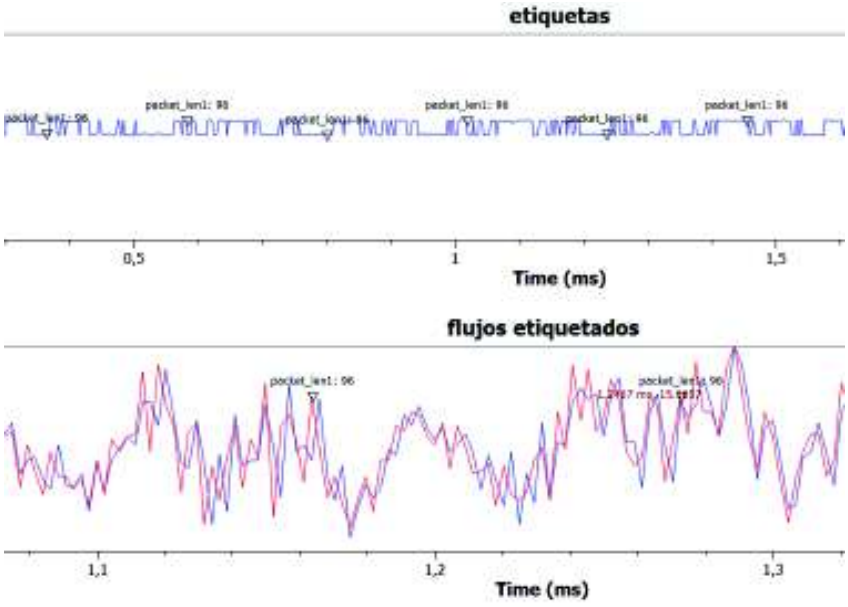

Figura 3. 10 Flujo de bits etiquetado

La longitud de cada paquete es de 96 bits y las etiquetas proveen esa información, por esta razón GNU Radio muestra las etiquetas sobre la señal de tiempo.

Posterior al etiquetado se agrega el CRC32, este código se agrega en cada etiqueta y se puede observar en la figura 3.11, como el tamaño de la longitud cambió después de agregar el CRC32 a los paquetes, las etiquetas cambian el valor de 96 bytes a 100 bytes.

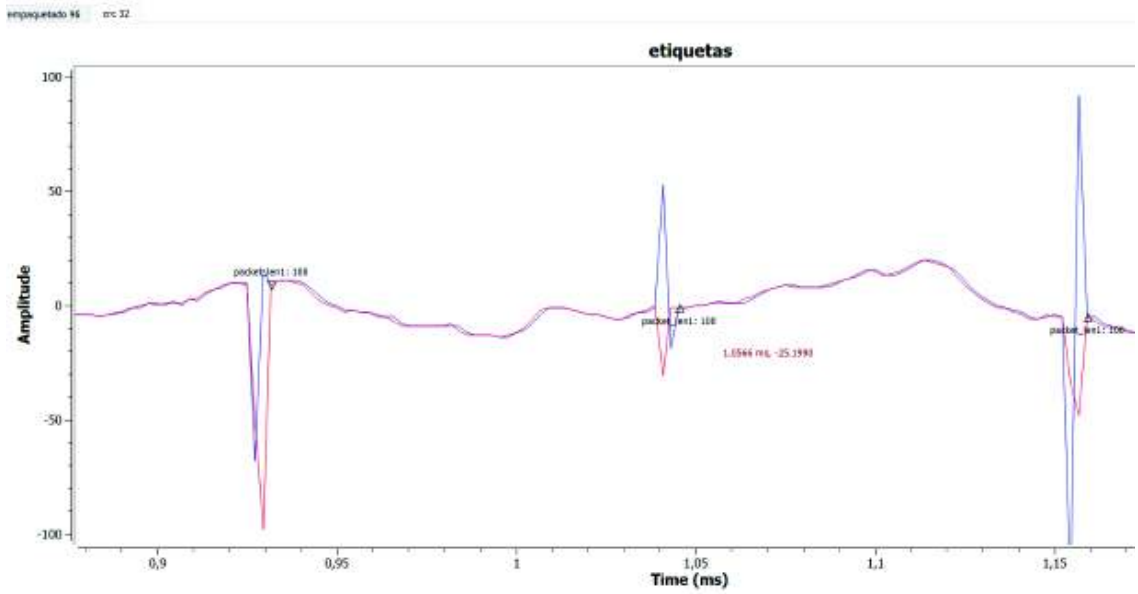

Figura 3. 11 Inserción del código CRC32

A continuación, el sistema genera una cabecera y al mismo tiempo prepara la carga útil para ser modulada, la cabecera generada es digital y su informacion viene dada por 0 y 1. El proceso de generación de cabecera se observa en la figura 3.12.

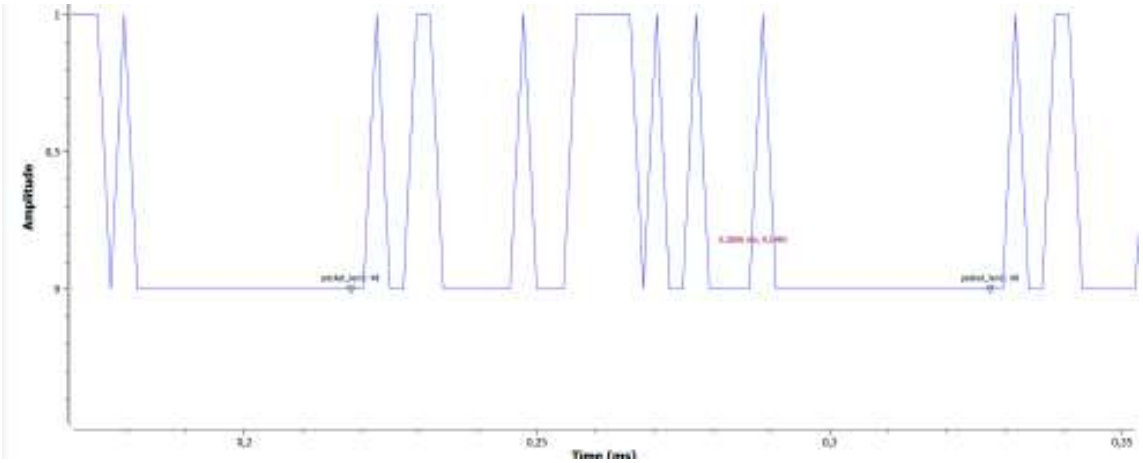

Figura 3. 12 Cabecera generada

Mientras las cabeceras son generadas el proceso de empaquetado de la carga útil convierte a la señal que ingresa en una señal digital de 4 niveles. Esto se observa en la figura 3.13.

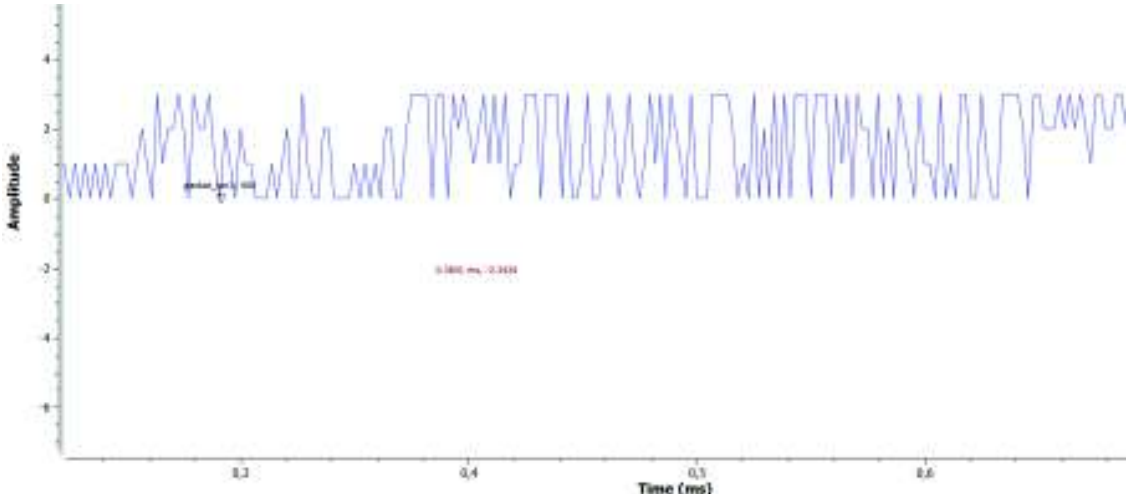

Figura 3. 13 Empaquetado de bits

A continuación del empaquetado la señal debe ser modulada. Como se describió en la sección 2.2.3 la modulación de cabecera y carga útil se realiza simultáneamente, cada modulación en bloques del mismo tipo, pero distintos. La modulación BPSK de la cabecera se puede observar en la figura 3.14.

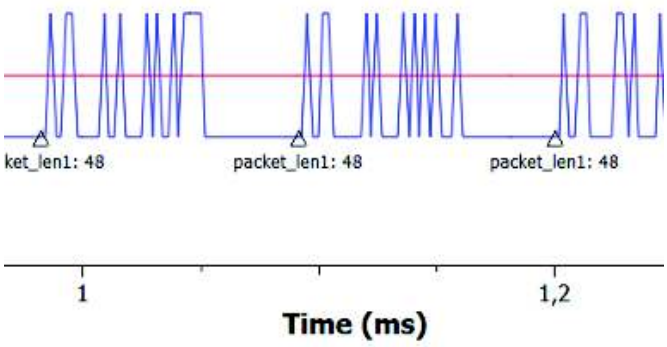

Figura 3. 14 Modulación de la cabecera en BPSK

La señal BPSK de la cabecera en el dominio del tiempo lleva la información de la longitud de los paquetes. Para verificar su modulación es posible observar el diagrama de constelación en la figura 3.15.

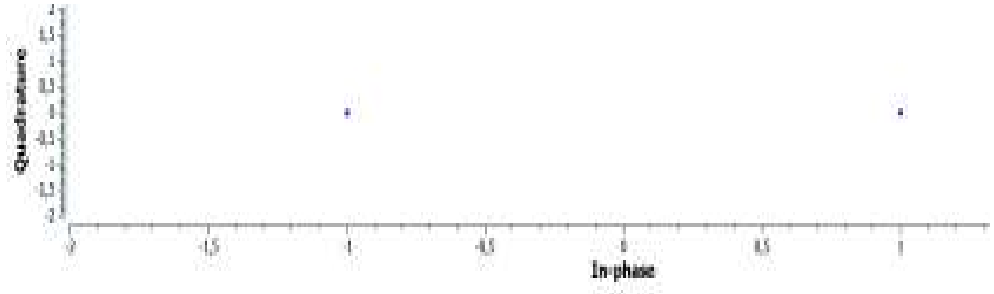

Figura 3. 15 Diagrama de constelación BPSK

De igual manera, es posible observar la modulación de la carga útil en la figura 3.16.

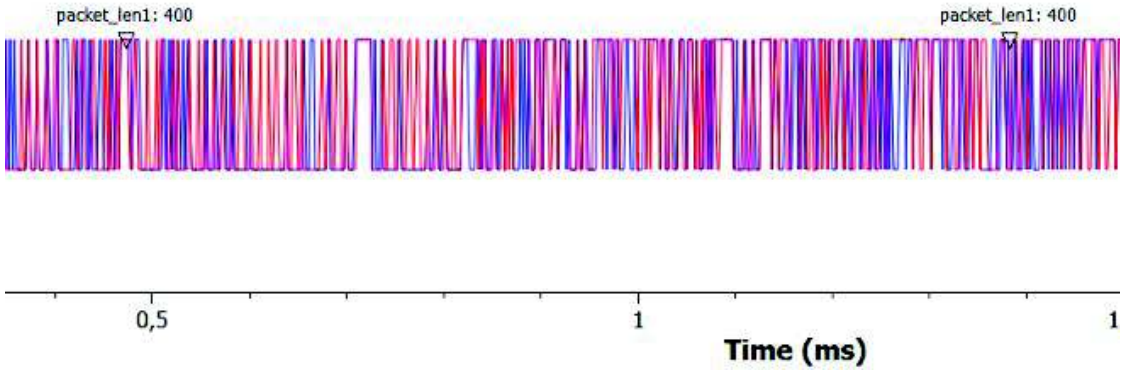

Figura 3. 16 Modulación de la carga útil

Es posible observar la longitud de los paquetes que fueron ensamblados en etapas anteriores a la modulación, para verificar la modulación QPSK de la carga útil se observa el diagrama de constelación generado por este bloque en la figura 3.17.

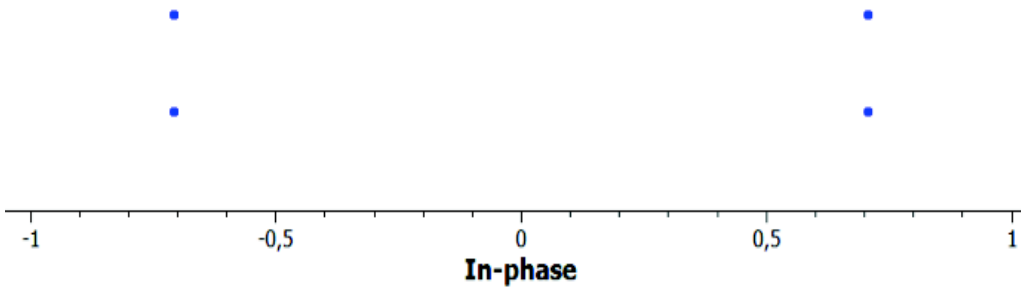

Figura 3. 17 Diagrama de constelación QPSK de la carga útil.

Posterior a las modulaciones ambas señales son multiplexadas, al multiplexar las señales se genera un flujo de símbolos en serie, este flujo se observa en la figura 3.18.

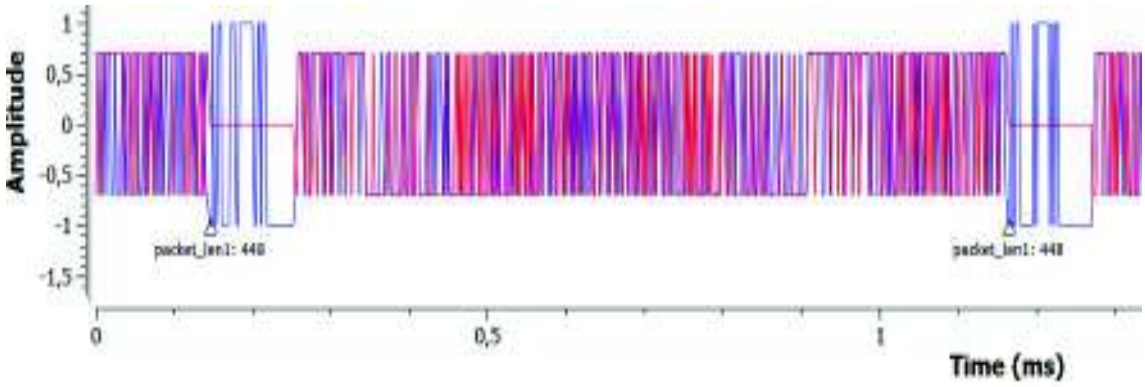

Figura 3. 18 Flujo de símbolos en serie resultado de la multiplexación de cabecera y carga útil

Junto con el flujo es posible observar la longitud de cada símbolo, además es posible observar claramente las cabeceras y carga útil. Al multiplexar las señales su diagrama de constelación también es multiplexado, esto se observa en la figura 3.19.

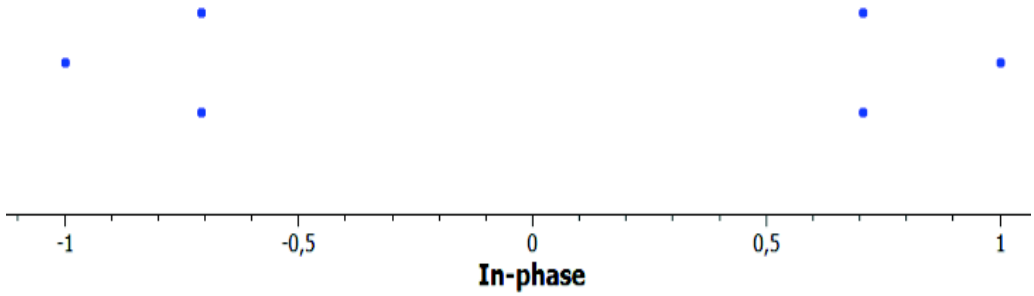

Figura 3. 19 Diagrama de constelación del flujo de símbolos multiplexado

La etapa pre-OFDM se completa con el multiplexado del flujo de símbolos en serie, este flujo ingresa a los bloques OFDM para su procesamiento. La figura 3.20 presenta el espectro de frecuencia de la señal OFDM a la salida de la IFFT. Se puede apreciar el ancho de banda ocupado por la señal transmita y en los extremos las bandas de guarda.

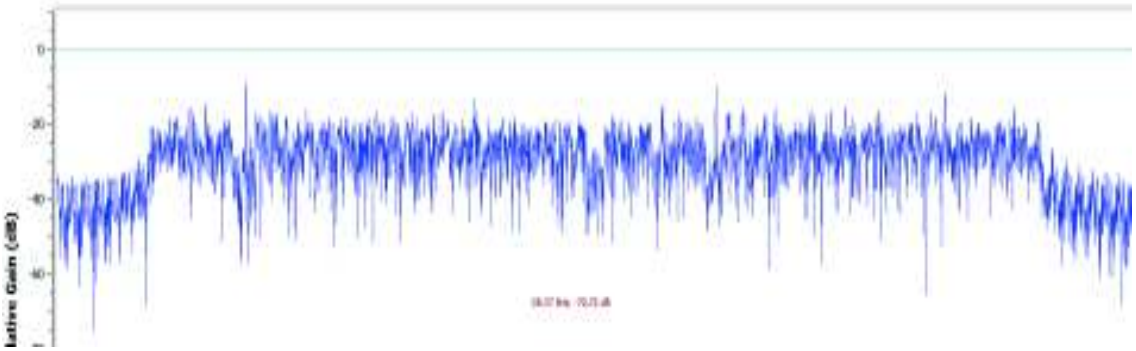

Figura 3. 20 Señal con el espectro ensanchado

A continuación de la IFFT se agrega un CP a la señal para ser transmitida, al insertar el CP se completa la señal en el dominio del tiempo que será enviada. Esta señal se observa en la figura 3.21.

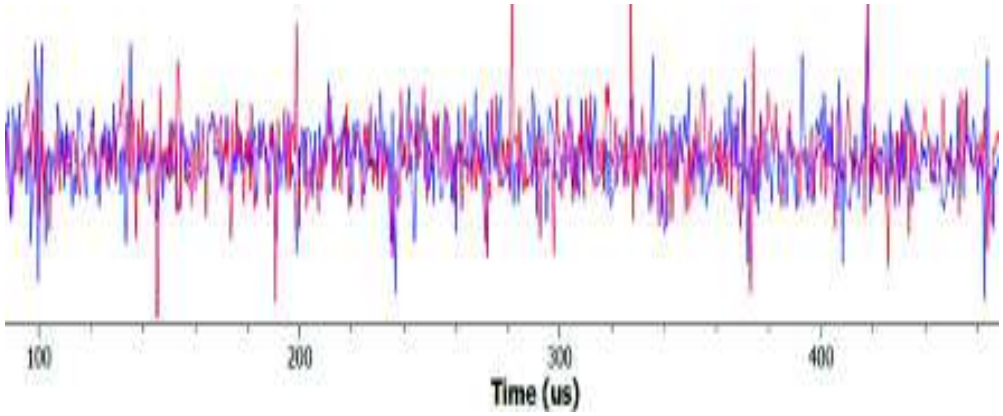

Figura 3. 21 Señal transmitida en el dominio del tiempo

La señal transmitida en el dominio de la frecuencia se presenta en la figura 3.22. Esta señal es similar a la obtenida por el transmisor modelo (figura 3.4).

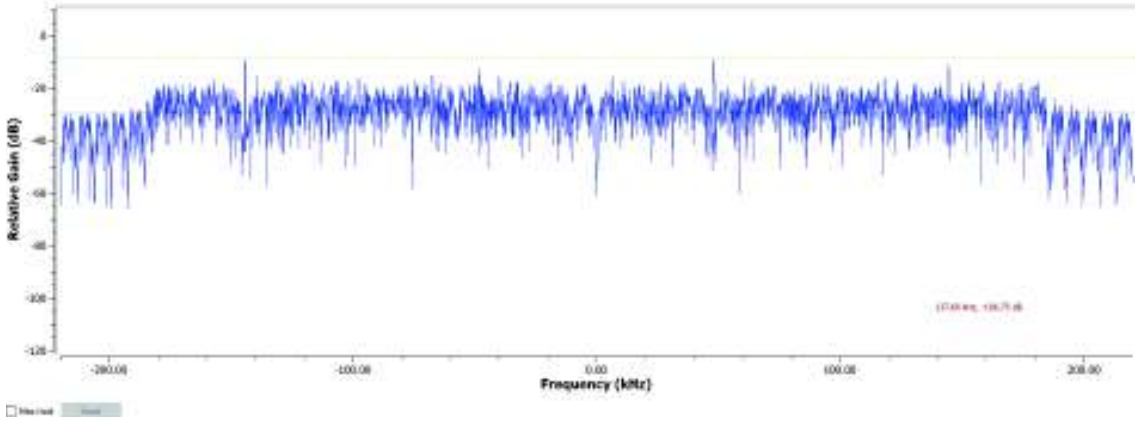

Figura 3. 22 Espectro de la señal a transmitir

Al receptor llega la señal atenuada, se utiliza el mismo canal simple, con atenuación que se utilizó en la prueba anterior. Es posible observar el espectro de la señal recibida en la figura 3.23.

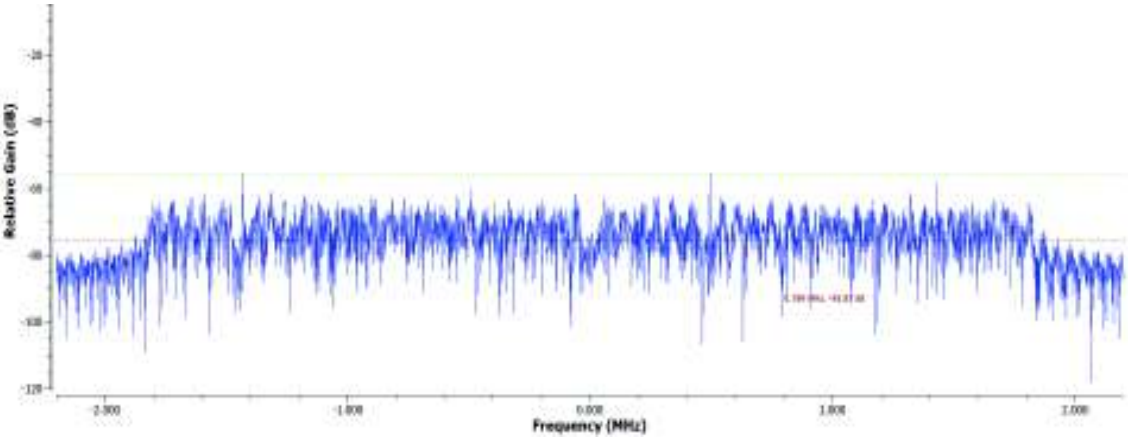

Figura 3. 23 Espectro de la señal recibida

Al ser el receptor modelo no es posible observar el procesamiento interno, razón por lo cual solamente es posible observar la señal recuperada. En la figura 3.24 se observa esta señal en el dominio del tiempo.

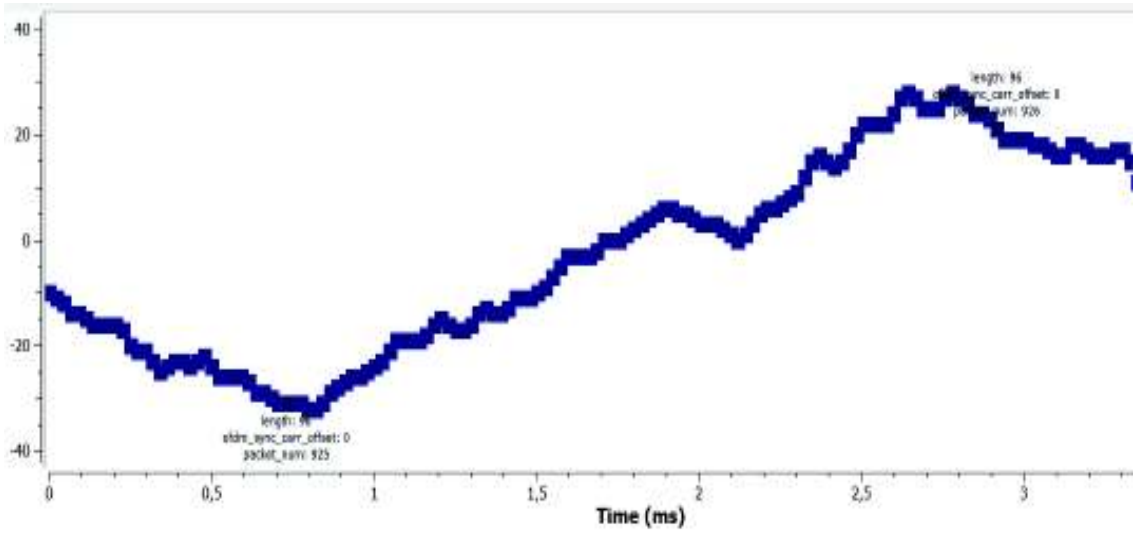

Figura 3. 24 Señal recuperada en recepción

En recepción es posible observar el número del paquete que llega y también su longitud. Se muestra la señal de mejor manera en la figura 3.25.

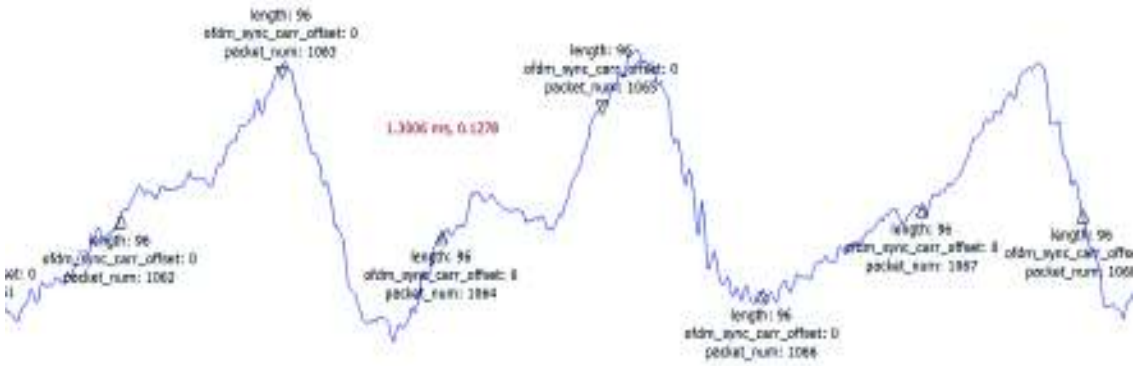

Figura 3. 25 Señal recuperada en recepción

En la figura 3.25 es posible visualizar la señal recuperada y el número de los paquetes que se ensamblaron. Al observar de esta manera, esta señal se asemeja a la señal de audio que ingresa (figura 3.8).

La simulación permite observar la compatibilidad de ambos archivos, el transmisor implementado y el receptor modelo, sin embargo, el procesamiento del transmisor es mucho mayor, esto se debe al procesamiento de interacción con el usuario al observar cada etapa de procesamiento. Incluso este procesamiento puede saturar la memoria RAM del ordenador donde se ejecute el programa.

Además, el procesamiento que se muestra es en tiempo real, razón por la cual, a la vez que se escucha el audio de la señal recibida, es posible observar el procesamiento en cada etapa. Como el procesamiento es rápido y no permite observar cada etapa es necesario configurar los instrumentos de medición para que actualicen los valores en intervalos de tiempo más largos, permitiendo observar mejor el procesamiento en las etapas, esto a su vez implica mayor procesamiento.

Para no saturar los procesos del CPU se agregan bloques “aceleradores” que GNU Radio proporciona en sus librerías. Este bloque ayuda a contrarrestar el elevado procesamiento en la CPU, sin embargo, estos bloques no permiten disminuir la utilización de memoria RAM.

Este inconveniente de saturación de memoria RAM y alto uso del procesador del ordenador no ocurre con la simulación que se realizó usando el transmisor y receptor modelo, esto se debe a que los bloques modelo de GNU Radio, no permiten observar el procesamiento interno.

3.1.3 Prueba de compatibilidad entre el transmisor modelo y el receptor implementado

Esta prueba permite observar el comportamiento del receptor implementado frente a un transmisor modelo configurado en GNU radio, se utiliza el mismo archivo de audio y la etapa de digitalización de pruebas anteriores. La señal que ingresa al transmisor se observa en la figura 3.26.

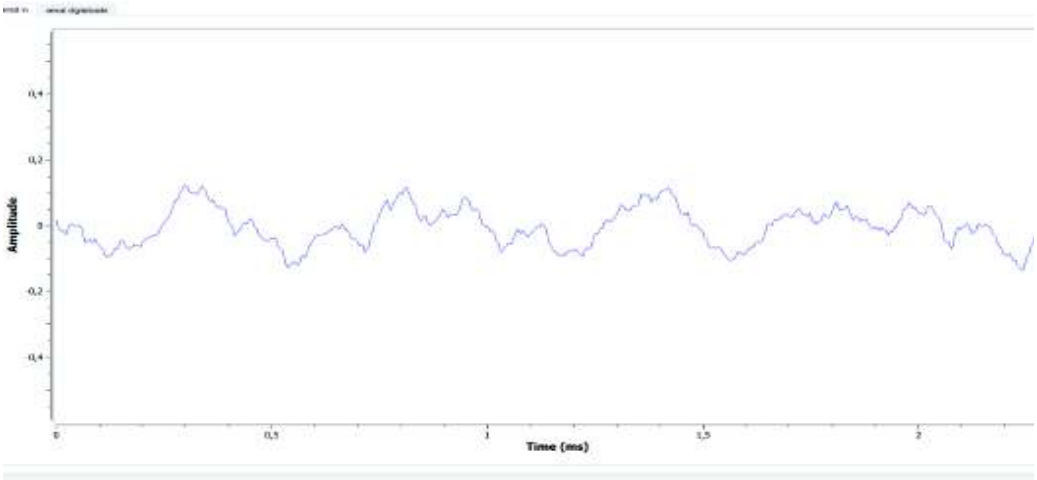

Figura 3. 26 Señal de entrada

La señal que ingresa es la misma que en la primera prueba pues es el mismo programa. La digitalización de la señal se puede observar en la figura 3.27.

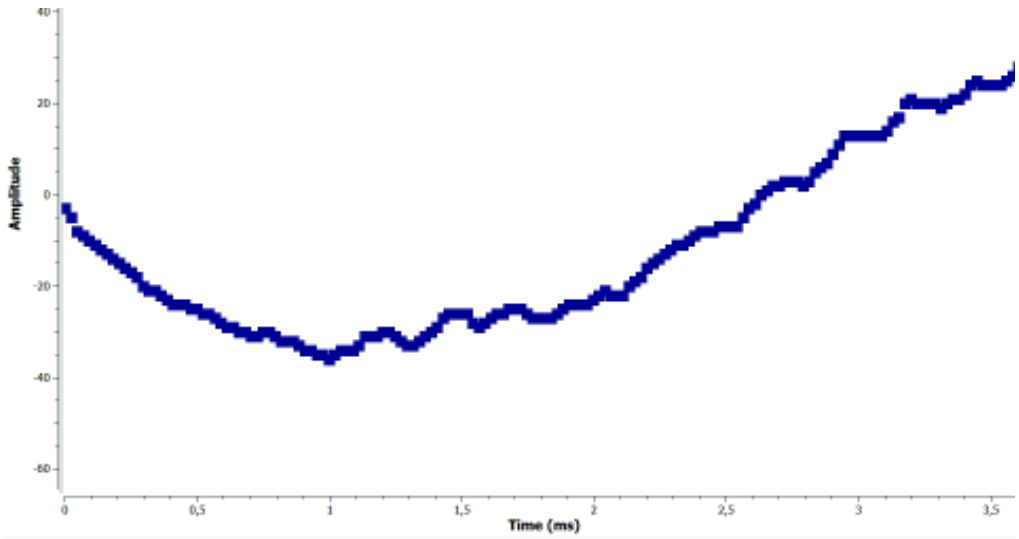

Figura 3. 27 Digitalización de la señal

De igual manera como la primea prueba, el bloque transmisor modelo no permite observar el procesamiento interno, por esta razón la señal que se muestra en la figura 3.28 es la señal OFDM que se transmite en el canal, la cual esta lista para transmitirse.

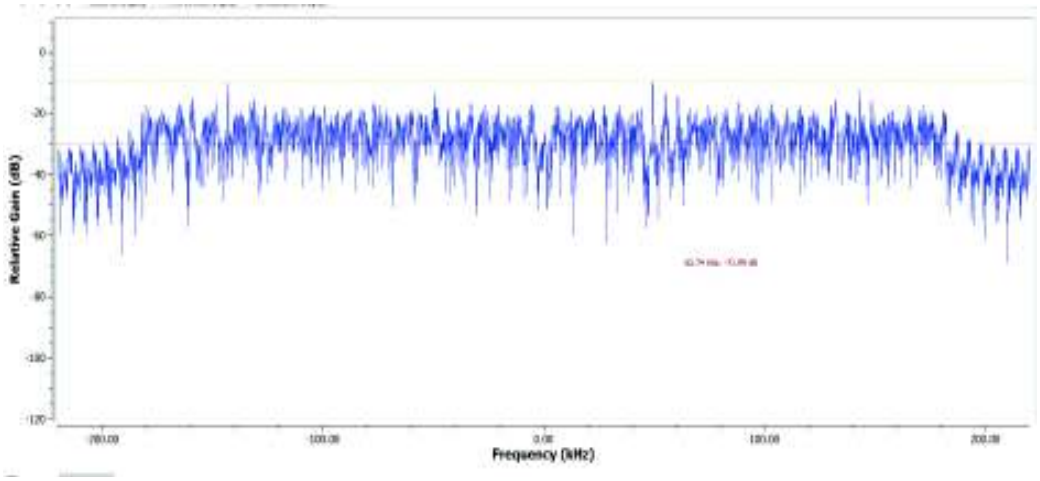

Figura 3. 28 Espectro en frecuencia de la señal a transmitir**.**

La señal que sale del transmisor pasa por el canal con atenuación, el mismo que se utilizó en las pruebas anteriores. La señal que llega al receptor se puede observar en la figura 3.29.

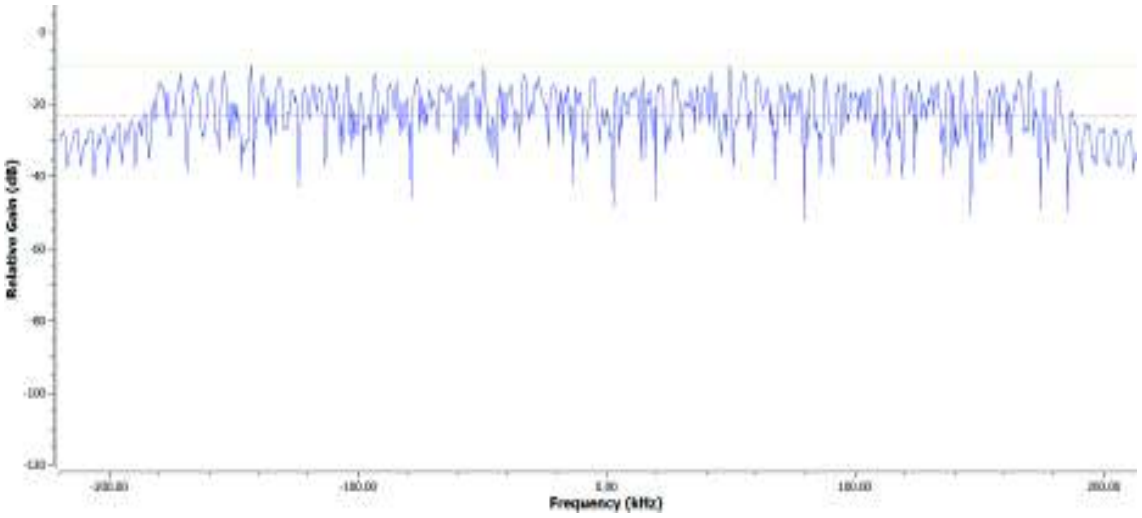

Figura 3. 29 Señal que llega al receptor

El nivel de potencia en la señal de recepción es menor, esto se debe al canal de transmisión que tiene atenuación. Como primera fase en recepción es necesario sincronizar la señal que llega, El desfase en frecuencia se observa en la figura 3.30.

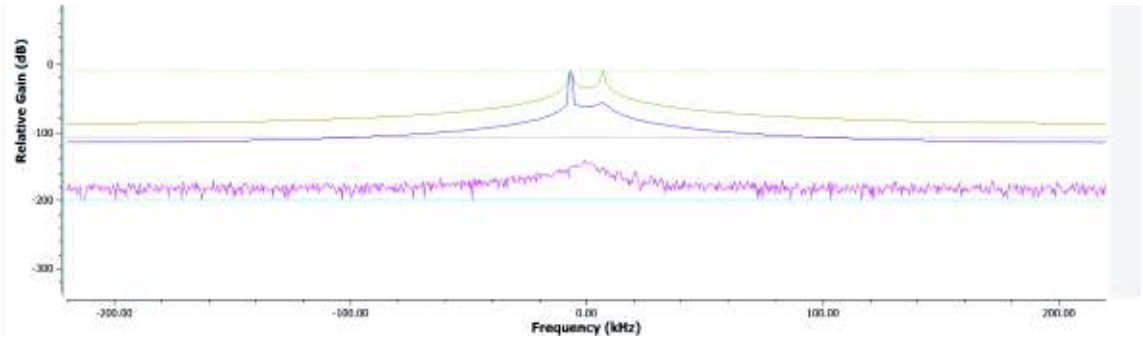

Figura 3. 30 Desfase de frecuencia generado en el modulador analógico.

En la figura 3.30 se observa la amplitud de la señal de desfase en frecuencia que se genera en el bloque sincronismo, esta señal se genera al tomar valores instantáneos cuando las muestras llegan al receptor, sin embargo, las muestras instantáneas tienen cambios rápidos que no son posibles mostrar, el bloque modulador toma las muestras instantáneas y las normaliza generando una señal que es posible analizar y corregir, es por esta razón que el modulador de frecuencia es parte del bloque sincronismo. Aunque el grafico de la figura 3.30 entregue valores con la frecuencia o como frecuencia central, lo que realmente se interpreta es que de la frecuencia central a la cual se transmita habrá un desfase como se muestra en los picos de la señal azul de la figura 3.30. la señal de color verde es el desfase máximo y la rosada es el desfase mínimo.

Adicionalmente, el bloque de sincronismo envía la información de detección de cabecera al bloque multiplexor, de esta manera este bloque puede clasificar la información de la

cabecera y carga útil. La información de detección de cabecera se envía como una señal de pulsos, esta señal se observa en la figura 3.31.

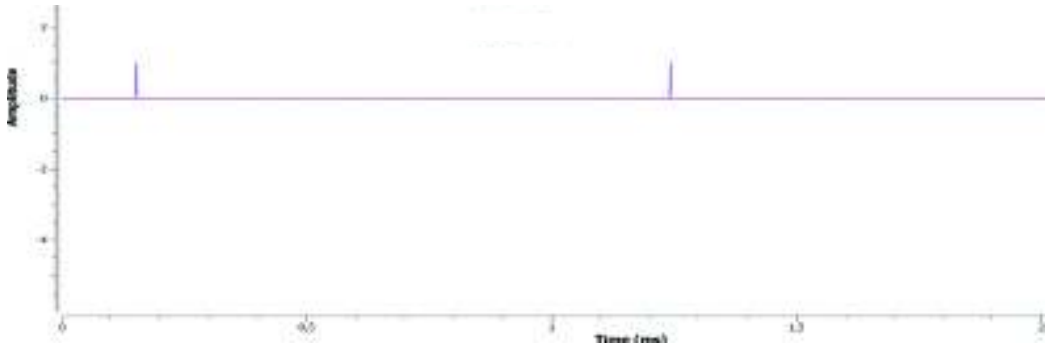

Figura 3. 31 Información de detección

Por su parte el demultiplexor toma la información del bloque sincronizador (sincronismo y detección) y la señal de información, entonces entrega dos flujos, el de la cabecera y el de la carga útil. El flujo de la información de la cabecera se puede observar en la figura 3.32.

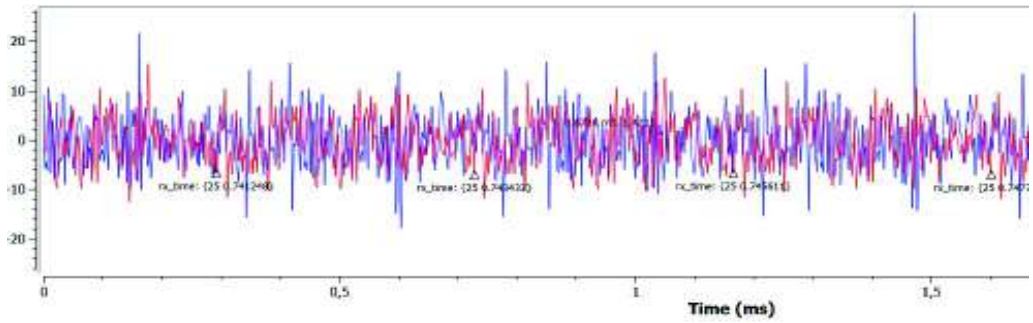

Figura 3. 32 Flujo de símbolos de la cabecera

El flujo ahora se encuentra delimitado, esta información se debe a la información de sincronización que el multiplexor utiliza. El espectro de frecuencia del flujo de símbolos de la cabecera se muestra en la figura 3.33.

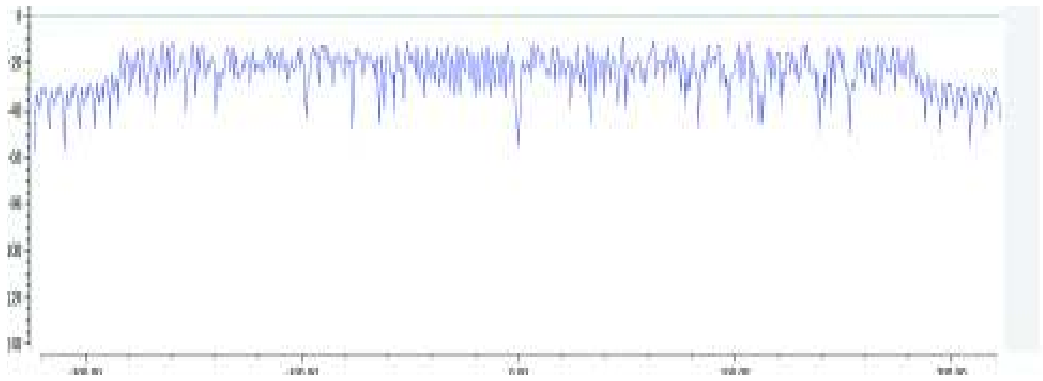

Figura 3. 33 Espectro de frecuencia del flujo de símbolos de la cabecera

El flujo de símbolos de la carga útil se puede observar en la figura 3.34. Este flujo de la carga útil lleva información de control como: sincronismo, longitud del paquete, numero del paquete, longitud de la trama, tiempo de llegada, misma que se encuentra en forma de etiquetas. GNU radio muestra esta información al inicio de cada paquete como se observa en esta figura, esta información se agrega en el demultiplexor.

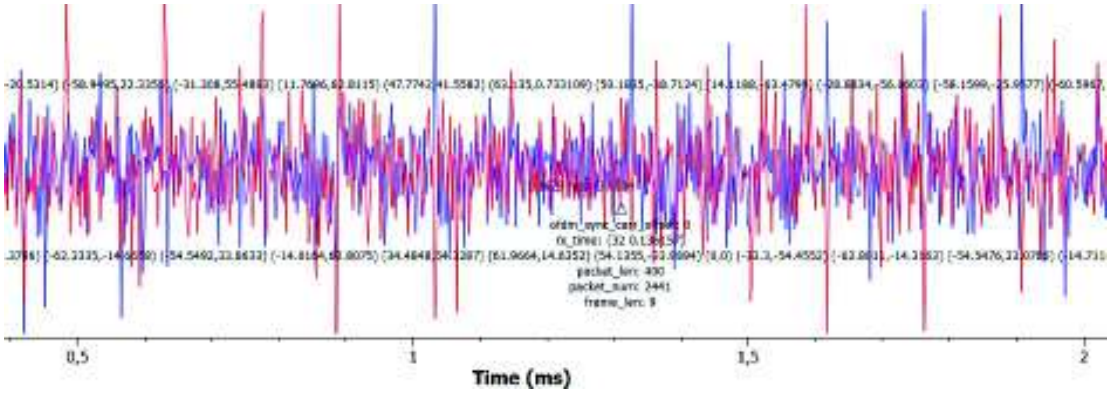

Figura 3. 34 Flujo de símbolos de la carga útil

El espectro de frecuencia de este flujo se observa en la figura 3.35.

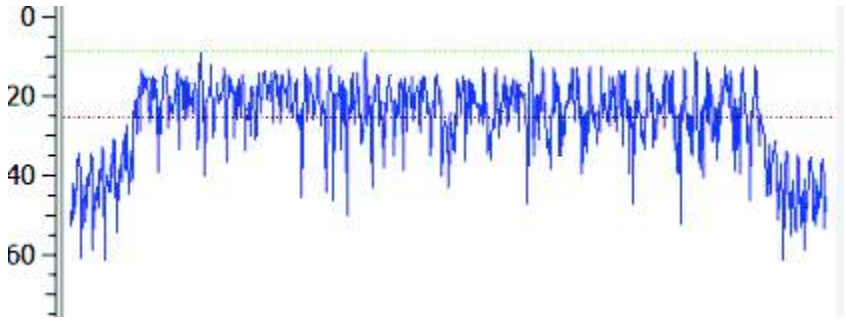

Figura 3. 35 Espectro de frecuencia del flujo de símbolos de la carga útil.

El procesamiento de los flujos de la cabecera y carga útil se realiza mediante dos procesos paralelos, es decir, se realizan procesamientos simultáneos para ambos flujos en bloques separados. En primer lugar, se muestra el procesamiento del flujo de la cabecera.

Después del procesamiento OFDM, el flujo de la cabecera que se entrega al demodulador se puede observar en la figura 3.36 y el diagrama de constelación de dicho flujo se observa en la figura 3.37.

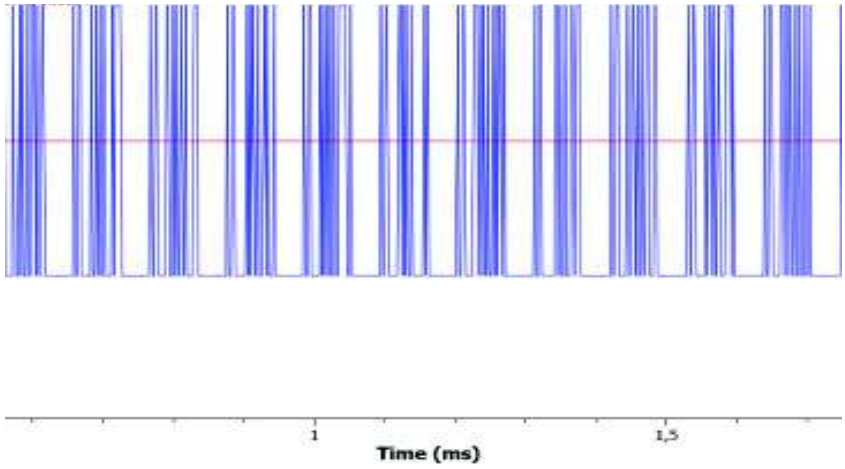

Figura 3. 36 Información de la cabecera a la salida del procesamiento OFDM

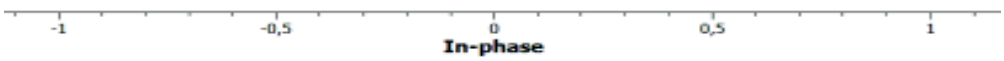

Figura 3. 37 Diagrama de Constelación del flujo de símbolos de la cabecera

El proceso de demodulación del flujo BPSK de la cabecera entrega un flujo de bits que se observa en la figura 3.38.

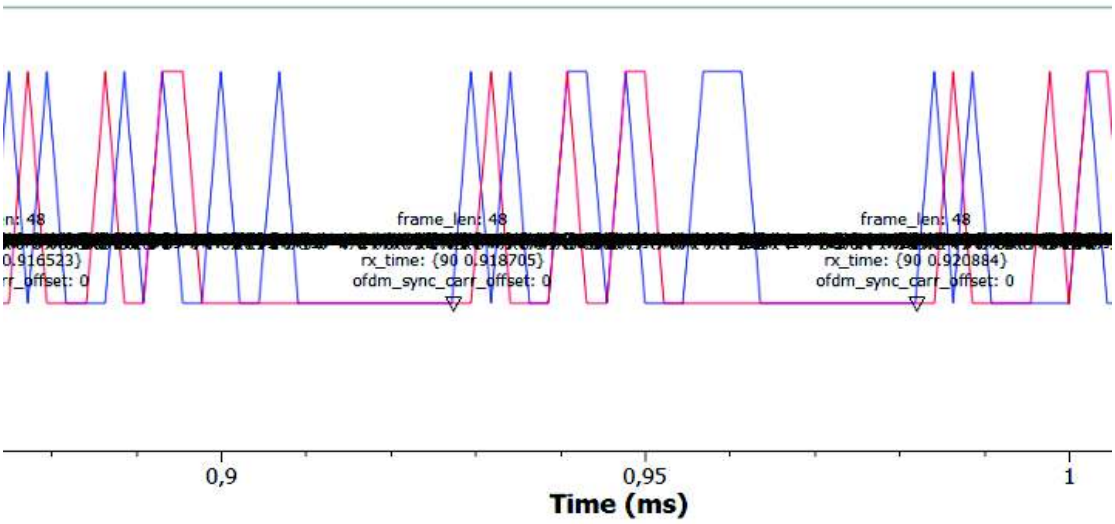

Figura 3. 38 Flujo de bits de la cabecera demodulados

El flujo de bits de la cabecera permite informar al bloque demultiplexor lo referente a la estimación de canal, ecualización y la información que las cabeceras transportan, eso lo hace mediante la realimentación. Cabe recalcar que GNU Radio en la visualización no permite retirar las etiquetas de información que se agregan, es por ello que ciertos valores no se pueden apreciar con claridad.

El segundo flujo que el demultiplexor entrega es el flujo de símbolos de la carga útil, este también lleva información de control en forma de etiquetas, el flujo que sale del multiplexor y se observa en la figura 3.34 se entrega al conjunto de bloques OFDM para procesarlo. Posterior a este procesamiento se entrega un flujo de datos que se muestra en la figura 3.39.

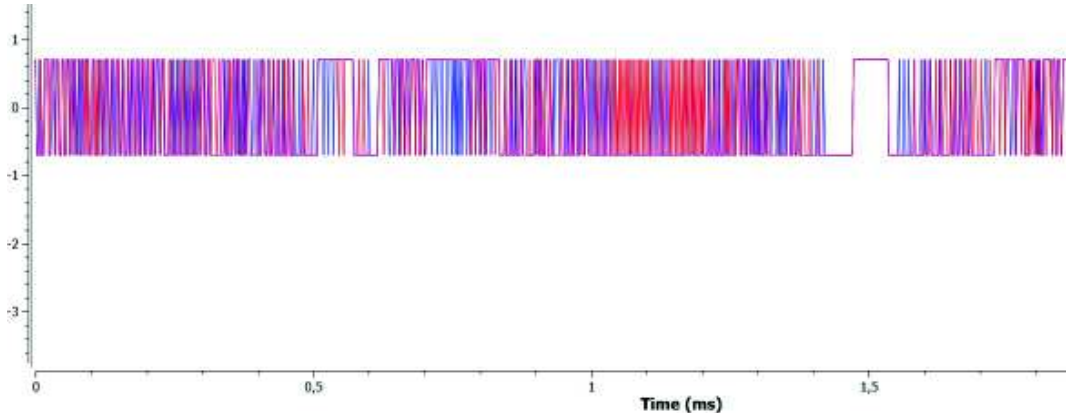

Figura 3. 39 Flujo de símbolos de la carga útil listo para ser demodulado

El flujo de la cabecera se encuentra modulado en QPSK, por esta razón se puede observar los 4 puntos en su diagrama de constelación de la figura 3.40.

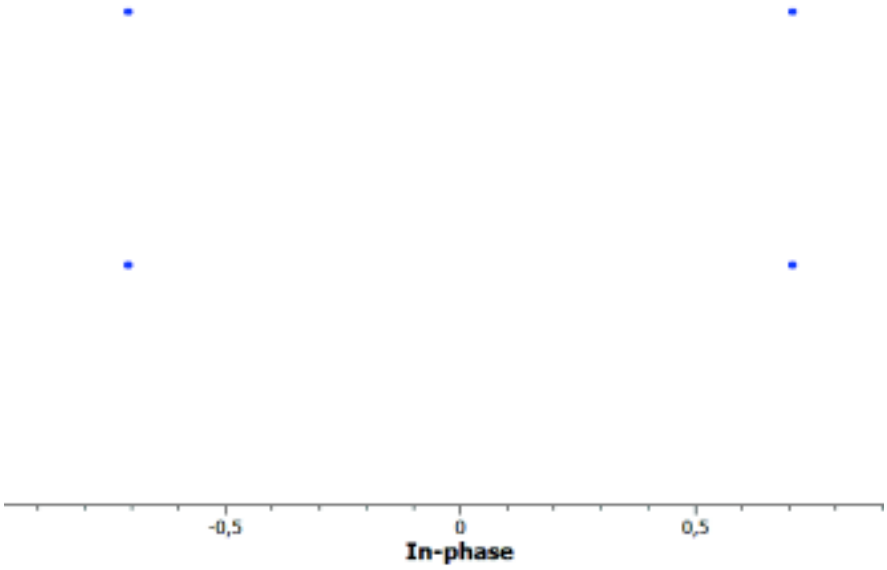

Figura 3. 40 Diagrama de constelación del flujo de símbolos de la carga útil

El flujo demodulado de la cabecera se lo puede observar en la figura 3.41.

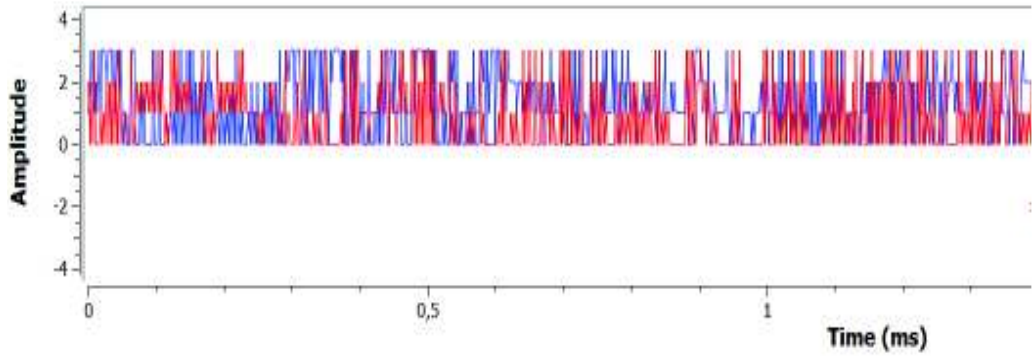

Figura 3. 41 Flujo de la cabecera demodulado

Posterior a la demodulación de la información los paquetes deben ser desempaquetados, este proceso entrega un flujo de bits etiquetado. Este flujo se observa en la figura 3.42. sin embargo, la información que GNU Radio muestra en los instrumentos de visualización es tan extensa que no es posible discriminar muy bien, esto se debe a que GNU radio trata de mostrar toda la información posible en la señal. Para contrarrestar este inconveniente se utilizan otros instrumentos de visualización.

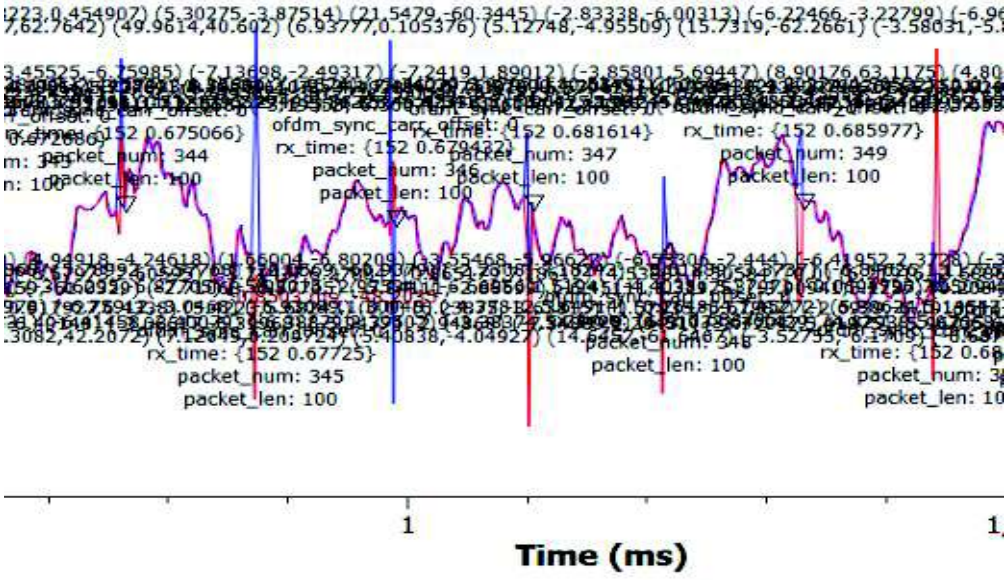

Figura 3. 42 Flujo de bits desempaquetados.

Como se indicó anteriormente, la información que entrega el instrumento de observación de GNU radio muestra muchos valores innecesarios que no son posibles remover, sin embargo, en toda esa información se puede apreciar la longitud de los paquetes, numeración de los paquetes, tiempo de arribo del paquete y los valores en ese instante de la señal. La misma señal se observa en la figura 3.43, pero sin toda la información que GNU radio entrega.

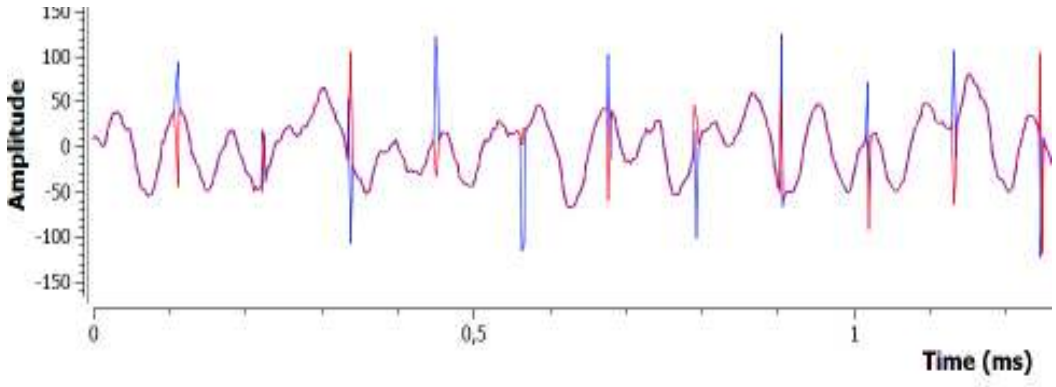

Figura 3. 43 Flujo de bits desempaquetados

Este flujo de información se encuentra etiquetado, razón por lo cual se debe verificar el CRC32. Tras la revisión del CRC32 se entrega un flujo que se muestra en la figura 3.44.

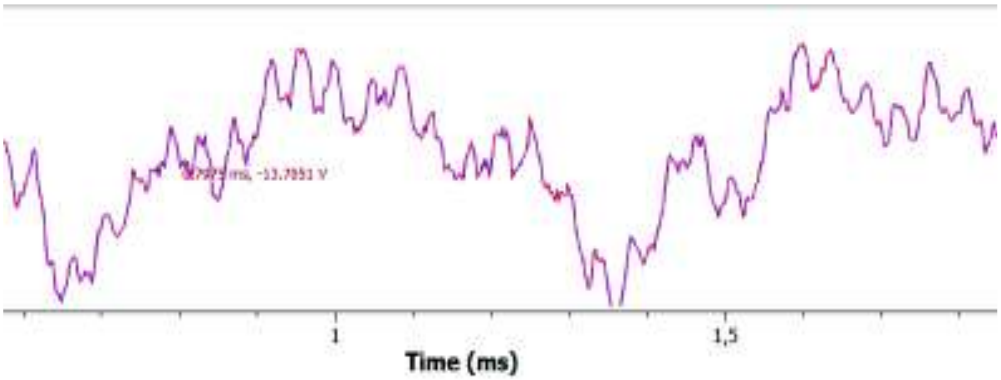

Figura 3. 44 Flujo de bits desempaquetados

La información se debe normalizar, esto se especificó en la sección 2.4.2, a la salida de este proceso la información recuperada se observa en la figura 3.45.

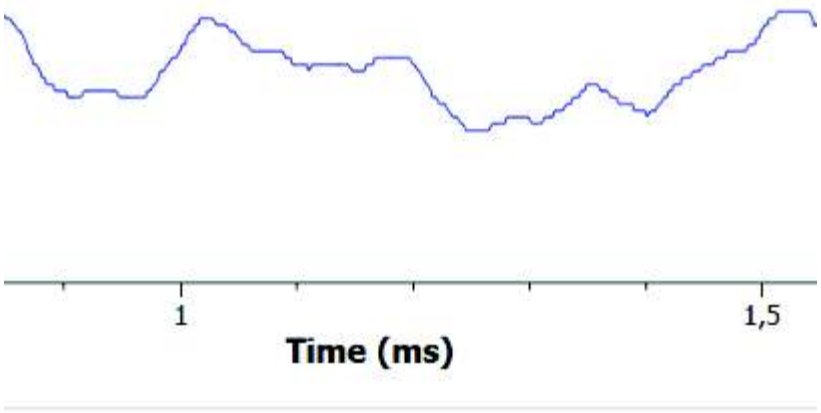

Figura 3. 45 Señal en recepción de la información recuperada.

La compatibilidad de los archivos receptor implementado y transmisor modelo resulta satisfactoria, sin embargo, esta simulación tiene inconvenientes con el procesamiento del computador, esto se debe al consumo de recursos del computador y la RAM debido al procesamiento por etapas y visualización de las mismas. Además, para poder observar las etapas es necesario disminuir la velocidad de actualización de los datos en los bloques de visualización, esto genera mayor procesamiento y consumo de recursos. A pesar de estos inconvenientes, el audio del archivo es recuperado.

La siguiente prueba de compatibilidad se realiza entre el transmisor y receptor que fueron implementados, reemplazando los bloques de transmisor y receptor modelo. Hasta este punto ambos prototipos funcionan con los modelos que GNU Radio provee, de esta manera se muestra independencia entre Transmisor y receptor.

3.1.4 Simulación de la compatibilidad del transmisor implementado con el receptor implementado

En esta prueba se simula la compatibilidad de los prototipos transmisor y receptor, ambos prototipos consumen muchos recursos de procesamiento del ordenador para visualizar las etapas de procesamiento de la señal. El archivo de audio que funciona como señal de entrada se observa en la figura 3.46.

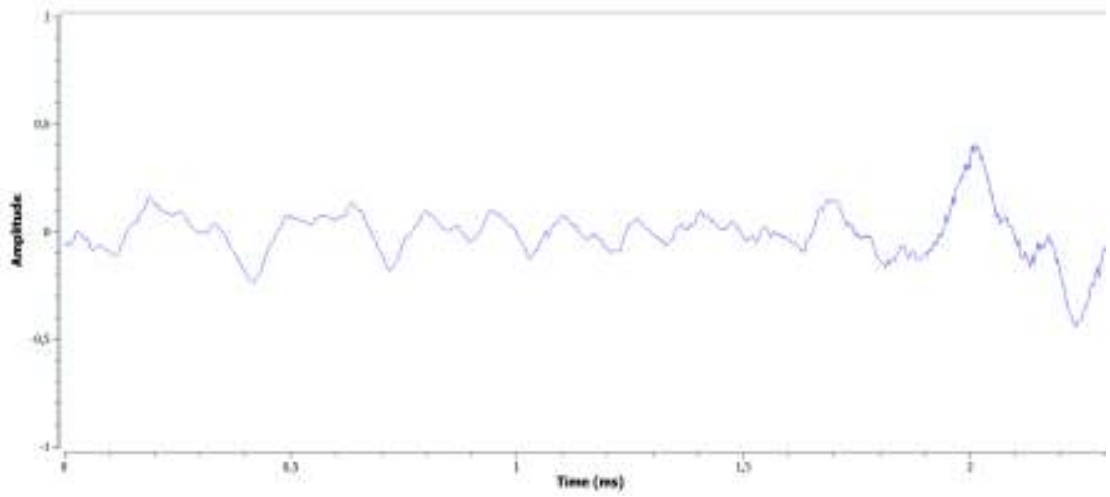

Figura 3. 46 Señal de audio que ingresa en el sistema

El procesamiento que se realiza en la señal es el mismo realizado por el transmisor implementado, estos resultados se muestran en la sección 3.1.2, por esta razón no se muestran nuevamente estas imágenes del procesamiento. Así mismo, el procesamiento realizado por el receptor es el procedimiento realizado por el receptor implementado en la

sección 3.1.3, después de este procesamiento la señal con la información recuperada se puede observar en la figura 3.47.

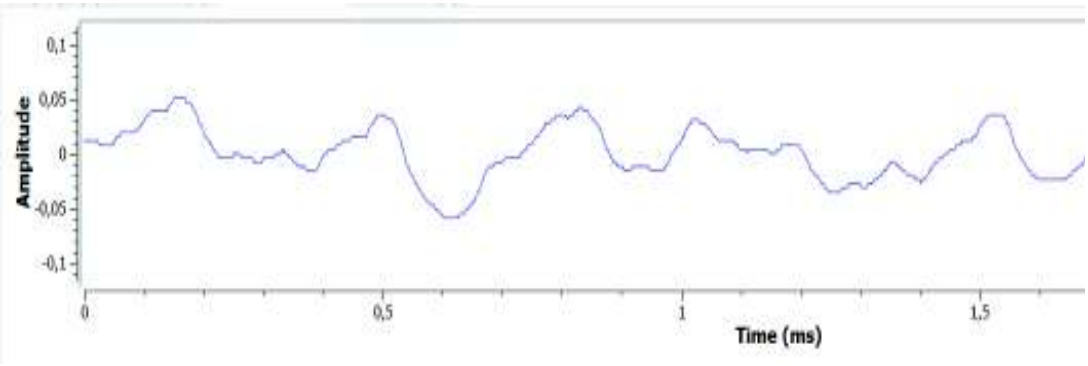

Figura 3. 47 Señal de información recuperada en recepción

La prueba permite observar una compatibilidad de extremo a extremo, además los parámetros configurados, como variables, frecuencias de muestreo, librerías, entre otras, se ejecuten sin inconvenientes de funcionamiento. Los instrumentos de visualización permiten observar la señal en las diferentes etapas o bloques, cabe recalcar que para cada una de estas etapas se utiliza instrumentos del mismo tipo, pero diferentes e independientes, es decir, cada instrumento consume recursos individualmente y al mismo tiempo.

En esta prueba el consumo de recursos de procesamiento es superior al utilizado en las pruebas realizadas en las secciones 3.1.2 y 3.1.3, incluso aunque se utilicen los bloques "*Throttle*" en GNU Radio, mismos que permiten reducir el procesamiento de los núcleos del computador, el consumo de memoria RAM es elevado. Además, esta prueba muestra que los prototipos funcionan sin problema, sin embargo, la memoria RAM del ordenador suele saturarse al momento de observar el procesamiento de las etapas, sin observar el procesamiento la transmisión – recepción de datos, el procesamiento se produce sin ningún problema de funcionamiento o compatibilidad.

La compatibilidad y funcionamiento verificados en las simulaciones permite excluir estos problemas para las siguientes pruebas. A continuación, se configura y conecta los equipos USRP para realizar las pruebas de transmisión en tiempo real.

3.2 Pruebas de transmisión en tiempo real con los prototipos transmisor – receptor

Como se indicó al inicio de este capítulo, éstas pruebas presentan el funcionamiento de los prototipos implementados, configurados con los equipos USRP y la compatibilidad entre estos. Como se describió se realizan dos pruebas para realizar esta comprobación, estas pruebas se realizan en un ambiente controlado.

El ambiente controlado en el cual se llevan a cabo estas pruebas es un ambiente “in-door”, es decir, se realizan las pruebas en un cuarto cerrado, específicamente en el laboratorio de comunicaciones inalámbricas 3, ubicado en el 7mo piso del edificio de química – eléctrica. Se implementan 2 estaciones, una transmisora y una receptora. Esto se realiza conectando dos computadoras y dos equipos USRP y configurándolos adecuadamente.

Parte del ambiente controlado es el medio de transmisión o canal para la comunicación, en la primera prueba este canal se implementa a través de un medio guiado, al usar este canal de transmisión la prueba busca comprobar la correcta configuración de los equipos USRP. A continuación, se reemplaza el medio guiado y se instalan las antenas en los equipos, esto se realiza para cambiar el canal de transmisión. Es así como la interfaz aire pasa a ser el canal de transmisión.

Al usar la interfaz aire como canal de transmisión, las pruebas buscan comprobar el funcionamiento final de los prototipos. Es por esta razón que las pruebas con este canal se realizan transmitiendo a 3m y 6m de distancia, así se comprueba el rango de cobertura de los equipos. También se añade obstáculos en la línea de vista en la transmisión de 6m, esto se realiza para verificar que el nivel de potencia en la señal disminuya mostrando que el prototipo recibe la señal que se transmite y que esta señal es propensa a las anomalías del canal, en este caso atenuación real.

Los equipos físicos que se utilizan en estas pruebas se mencionan a continuación:

- 1 Computador Core i7, memoria RAM 4GB.
- 1 Computador Laptop Core i5, RAM de 8GB.
- Dos equipos USRP NI 2920.
- 2 antenas VERT900.
- 2 cables Gigabit Ethernet.
- Cable coaxial SMA

- Atenuador 30 dB, 50 ohm, SMA

Las computadoras tienen instalado GNU Radio, los drivers UHD y los archivos de configuración de los prototipos transmisor – receptor. Cabe recalcar que las computadoras deben tener activados los dispositivos de audio para escuchar la señal transmitida y la señal recibida.

Los equipos USRP y las antenas se los puede observar en la figura 3.48.

Figura 3. 48 Equipos USRP conectados a las antenas VERT900

Los equipos USRP conectados a los ordenadores se pueden observar en la figura 3.49.

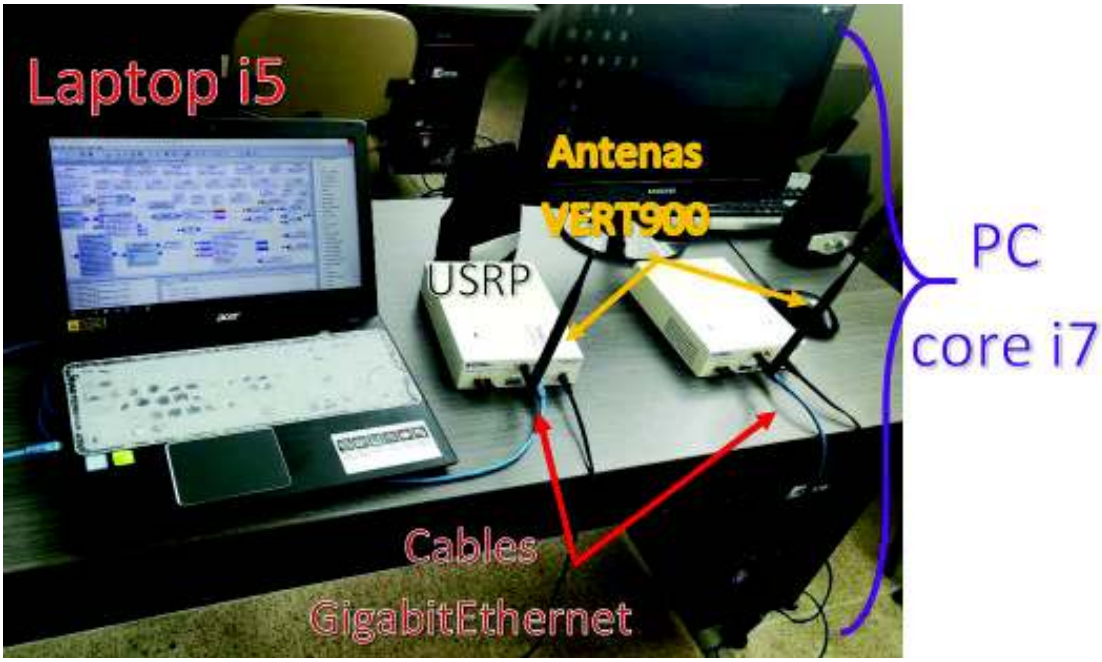

Figura 3. 49 Conexión equipos USRP con los ordenadores

3.2.1 Transmisión – recepción con canal guiado

Esta prueba se realiza para verificar la correcta configuración de los parámetros RF en los equipos USRP. Se reemplaza a las antenas VERT900 por el cable coaxial SMA, sin embargo, los fabricantes de *National Instruments* recomiendan que no se debe conectar dos equipos USRP sin usar un atenuador, esto se realiza para garantizar la integridad de los equipos frente a daños por exceso de potencia en recepción. Por esta razón se utiliza un cable coaxial SMA y un atenuador, que vienen incluidos en los equipos USRP y se los puede observar en la figura 3.50.

Figura 3. 50 Cable coaxial SMA y atenuador 30dB conector SMA

La conexión de este medio guiado como canal de transmisión con los equipos USRP se observa en la figura 3.51.

Figura 3. 51 Equipos USRP conectados a través de un medio guiado

Para la verificación de la configuración de los parámetros configurados en los equipos USRP se procede a ejecutar los programas del transmisor implementado y ambos receptores, el implementado y el modelo. Esto se realiza para verificar la configuración de los USRP en los programas (.grc) mencionados, también permite verificar nuevamente la compatibilidad entre estos, de esta manera se demuestra el funcionamiento independiente tanto del transmisor como del receptor.

Se realizan las pruebas de funcionamiento de transmisor OFDM implementado con el receptor modelo. Al ejecutar el prototipo transmisor OFDM en la laptop, la interfaz gráfica muestra el procesamiento por etapas, las mismas etapas mostradas en las pruebas por simulación, este proceso se observa en la figura 3.52.

Figura 3. 52 Estación 1, Laptop y equipo USRP como transmisor

A la par el ordenador de escritorio conectado al otro equipo USRP componen la estación 2, el receptor. El receptor modelo no genera problemas de compatibilidad en cuanto a la comunicación, esto comprueba nuevamente el funcionamiento del prototipo transmisor OFDM. Además, el audio recibido es similar al percibido en las simulaciones, esto se debe al medio guiado y la atenuación controlada (el atenuador). El receptor muestra en la interfaz gráfica el procesamiento de la señal recibida y como se recuperan los datos. Esto se observa en la figura 3.53.

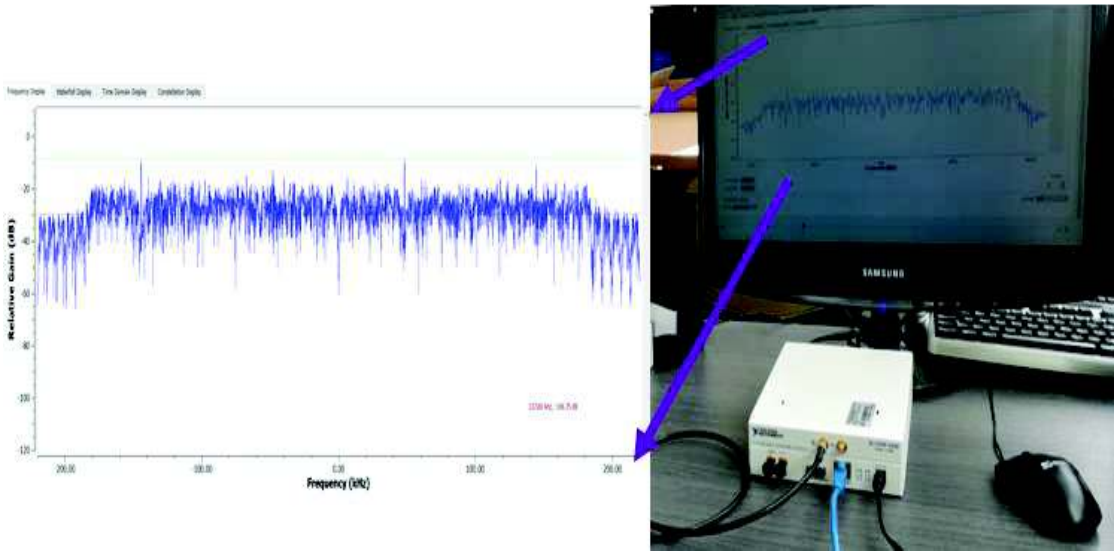

Figura 3. 53 Estación 2: receptor OFDM compuesto por el ordenador y el equipo USRP

Se debe tomar en cuenta que la configuración de la ganancia en los equipos para esta prueba es importante para el funcionamiento, aun con el atenuador si existe una mala configuración de este parámetro los resultados no serían adecuados, sino erróneos. De igual manera al ejecutar el receptor implementado es posible observar su funcionamiento y el procesamiento que realiza en sus etapas, la compatibilidad y correcta configuración de los equipos USRP con los programas generados en GNU Radio son verificadas nuevamente.

A pesar que la compatibilidad, configuración y funcionamiento son verificados en los equipos, el problema de consumo de recursos de procesamiento en los ordenadores continua. Al observar el procesamiento de la señal en cada etapa la memoria suele saturarse y la computadora detiene la ejecución de la transmisión. La figura 3.54 muestra el sistema de comunicación con canal guiado utilizado en estas pruebas.

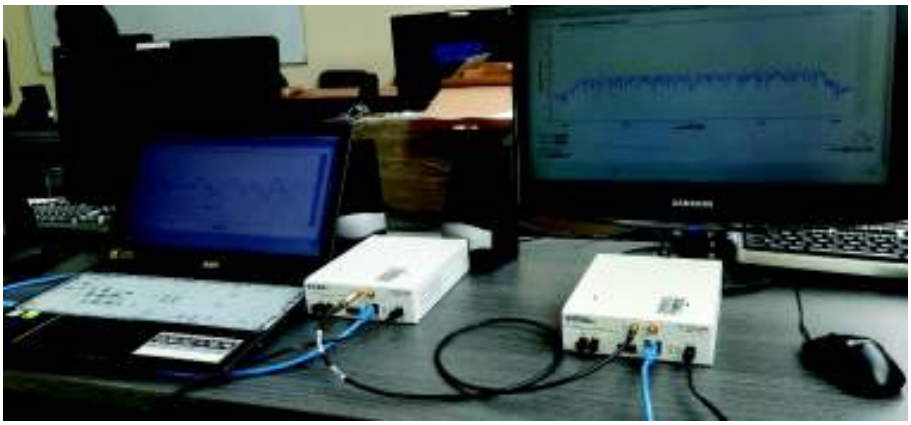

Figura 3. 54 Sistema transmisor implementado -receptor implementado con canal guiado

La atenuación del cable del canal guiado es despreciable, esto lo indica la información del fabricante, sin embargo, la atenuación de 30dB del atenuador actúa en la ganancia del transmisor, incluso si la ganancia de ambos equipos, transmisor y receptor se configura en 0 dB, el nivel de potencia de recepción está dentro del rango de funcionamiento especificado por el fabricante.

3.2.2 Transmisión – recepción con medio no guiado

Para estas pruebas se utiliza el sistema mostrado en la figura 3.49, se utiliza la misma potencia en estas pruebas (potencia definida por el fabricante entre 15 y 18 dBm, observar tabla 1.4), la ganancia de los equipos se fija en 30dB. En las pruebas se mantendrá la frecuencia y ganancia constantes, mientras que se variará la distancia entre transmisor y receptor de 3m a 6m. La figura 3.55 muestra al sistema transmisor – receptor a una distancia de 3m.

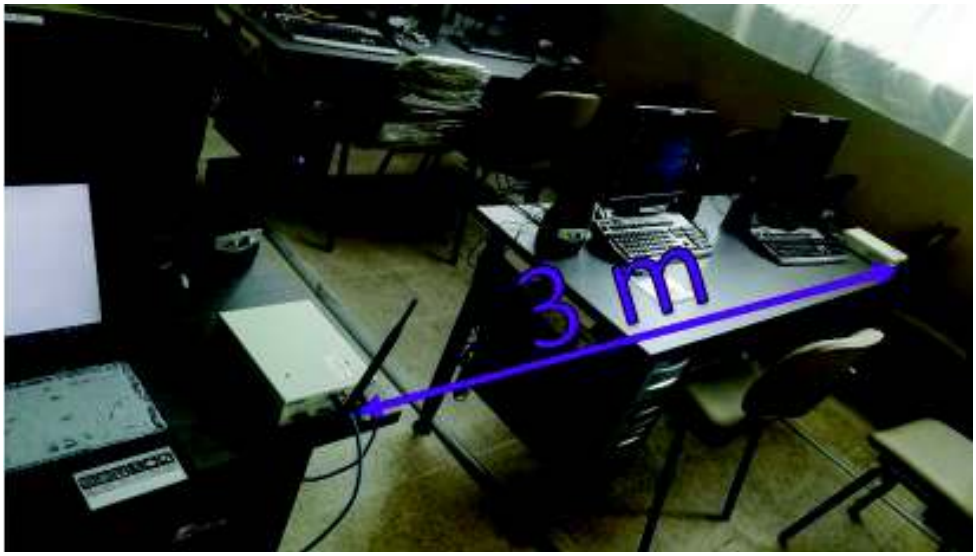

Figura 3. 55 Sistema transmisor - receptor OFDM a 3m de distancia

A 3 metros los equipos funcionan sin inconvenientes, el sistema de comunicaciones funciona correctamente y la señal en el receptor llega con un nivel de potencia al cual el equipo puede funcionar sin dificultades, incluso cuando se desea variar la frecuencia de transmisión y se sintonizan manualmente ambos equipos, no hay problemas de compatibilidad. Además, es posible observar el procesamiento en cada etapa, sin embargo, el consumo de recursos de procesamiento en los ordenadores es notable, la comunicación puede ser interrumpida al usar las herramientas de visualización.

Las anomalías del canal real, como atenuación, interferencia y principalmente el ruido afectan a la comunicación, esto se puede apreciar en el audio que la señal recupera con la

información recibida, este audio se escucha con interrupciones y se ha degradado con respecto al audio que se escuchó en las pruebas con el canal guiado.

El nivel de potencia que se puede observar en recepción cuando se transmite a 3m se puede observar en la figura 3.56.

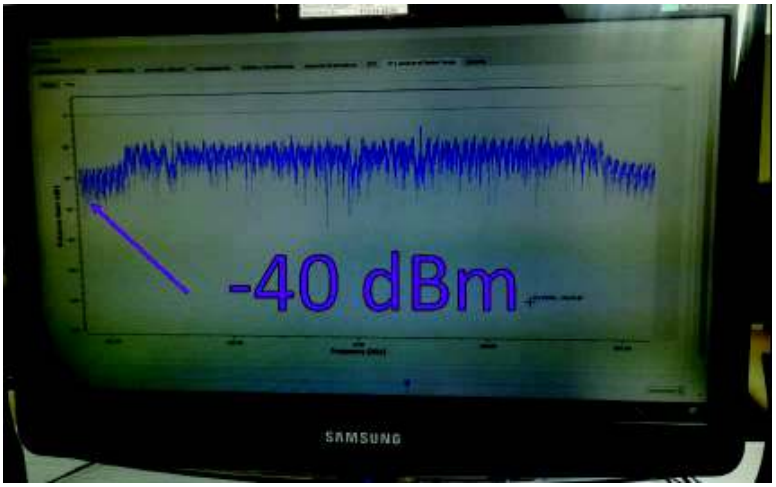

Figura 3. 56 Nivel de potencia de la señal recibida a 3m con línea de vista

Al aumentar la distancia entre los equipos como se observa en la figura 3.57 a 6m, el nivel de potencia en la señal recibida disminuye.

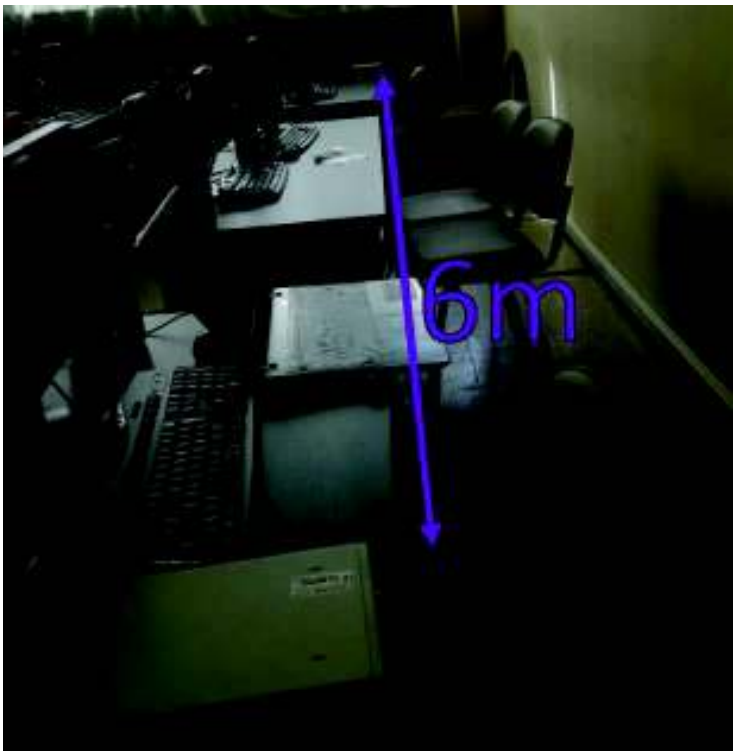

Figura 3. 57 Sistema Transmisor - receptor a 6m con línea de vista

Al aumentar la distancia entre los equipos, el nivel de potencia en recepción se reduce, eso se observa en la figura 3.58.

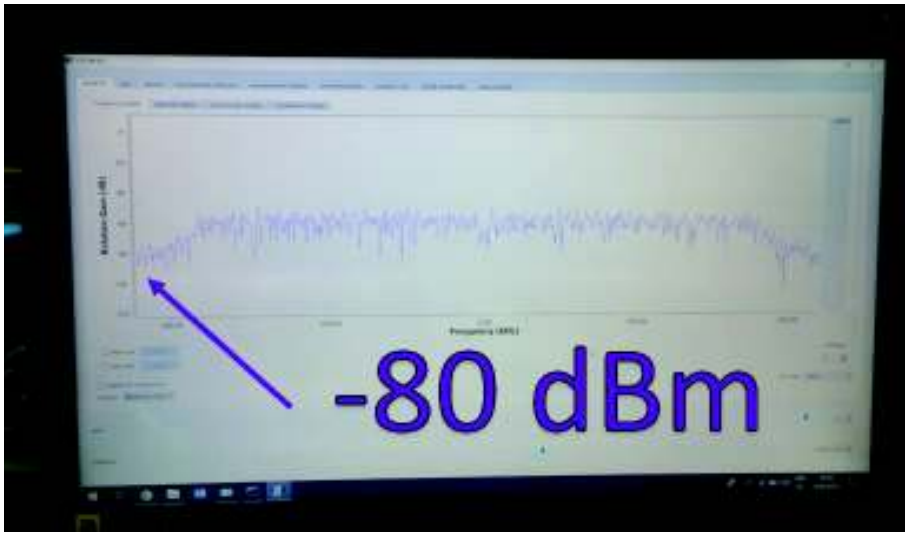

Figura 3. 58 Nivel de potencia de la señal recibida a 6m con línea de vista

En esta prueba el funcionamiento del sistema no presenta inconvenientes además del consumo de recursos de procesamiento, no obstante, al aumentar la distancia entre transmisor y receptor la información recuperada llega con menor nivel de potencia y con más ruido. El ruido se lo puede apreciar en el audio que el receptor recupera, el audio se escuchó más degradado con respecto a la transmisión a 3 metros.

Posteriormente, se agregan obstáculos en la línea de vista del sistema, el escenario a 6m, con transmisión sin línea de vista se observa en la figura 3.59.

Figura 3. 59 Escenario de prueba con obstáculos y distancia de 6m

El nivel de potencia de la señal recibida se observa en la figura 3.60.

Figura 3. 60 Nivel de potencia recibido en el escenario de pruebas con obstáculos.

La atenuación ha disminuido el nivel de potencia recibida, sin embargo, el nivel de potencia de esta señal recibida funciona sin inconvenientes, las anomalías del canal en la señal de audio que se reproduce no son tan notorias, sino que el audio es similar al recibido en la prueba realizada a 6 metros con línea de vista. Es posible que la densidad de los obstáculos no sea suficiente para degradar aún más el nivel de potencia que recibe el receptor.

3.3 Encuesta

Para evaluar la utilidad del prototipo se desarrolló una encuesta, esta encuesta además de ayudar a medir la utilidad permite medir la funcionalidad y el grado de satisfacción de los usuarios al momento de usar el prototipo.

Para la muestra se tomó una porción de 10 estudiantes de las carreras de Electrónica y Telecomunicaciones y Electrónica y Redes de Información, el requisito previo es que tengan conocimiento de OFDM, es decir, que estén cursando la materia de Comunicaciones Inalámbricas.

3.3.1 Resultados de las encuestas

- **PREGUNTA 1:** ¿Considera Ud. al prototipo didáctico y de fácil entendimiento?

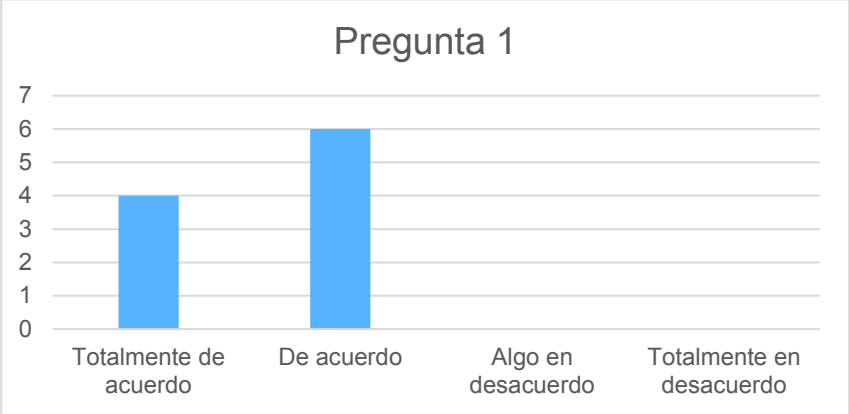

Figura 3. 61 Resultados de la pregunta 1

- **PREGUNTA 2:** Considera que el prototipo de Sistema Tx-Rx puede reforzar los conocimientos de OFDM.

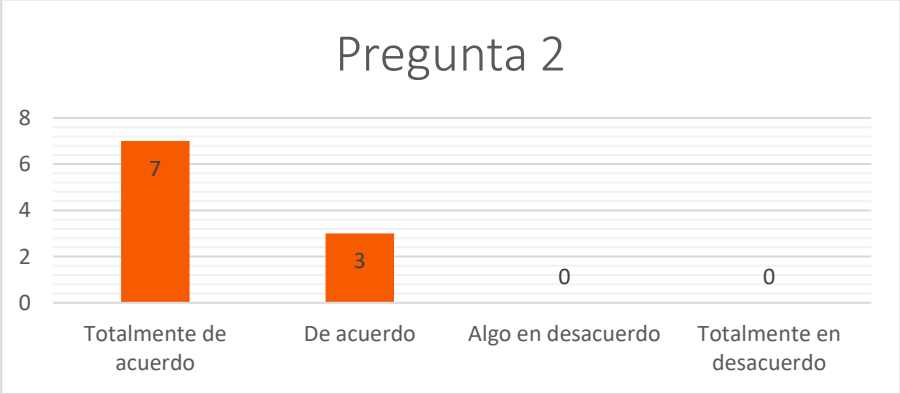

Figura 3. 62 Resultados de la pregunta 2

- **PREGUNTA 3:** ¿El prototipo de sistema Tx-Rx muestra cada etapa del procesamiento en OFDM?

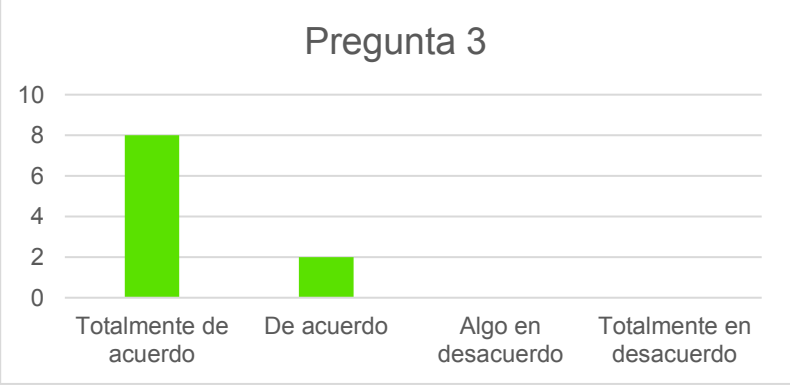

Figura 3. 63 Resultados de la pregunta 3

- **PREGUNTA 4:** Mediante el uso del prototipo, ¿pudo observar y comprender el procesamiento en OFDM?

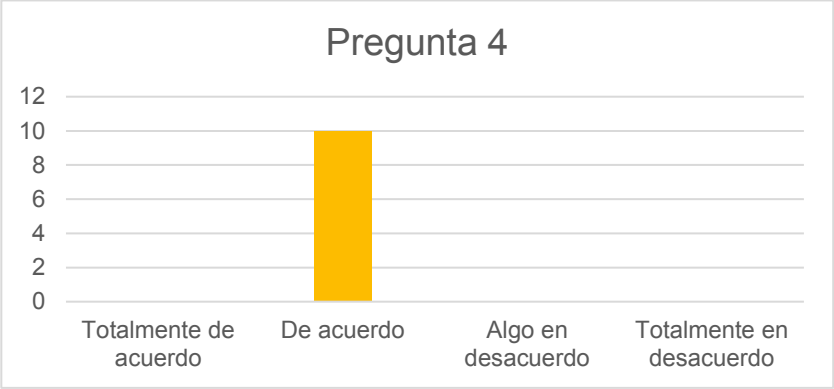

Figura 3. 64 Resultados de la pregunta 4

- **PREGUNTA 5:** ¿el uso de la interfaz es interactiva con el usuario?

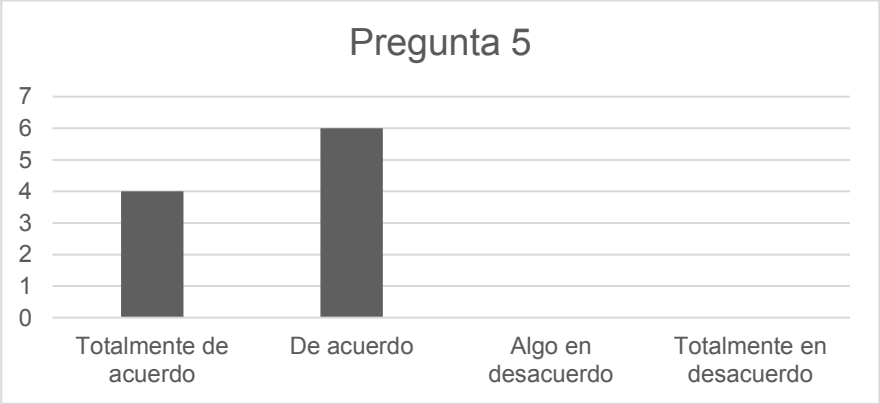

Figura 3. 65 Resultados de la pregunta 5

- **PREGUNTA 6:** ¿Considera que el prototipo de sistema Tx-Rx debería usarse en la materia de comunicaciones inalámbricas?

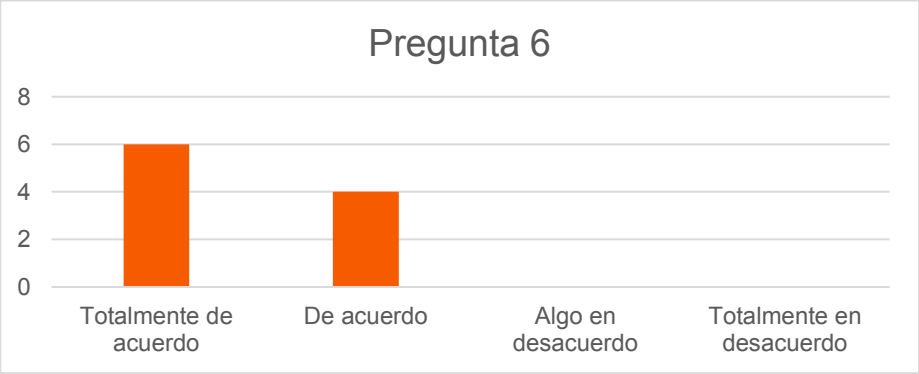

Figura 3. 66 Resultados de la pregunta 6

- **PREGUNTA 7:** ¿Cree Ud. que hacen falta más herramientas, prototipos o aplicaciones didácticas y educativas para la materia de comunicaciones inalámbricas?

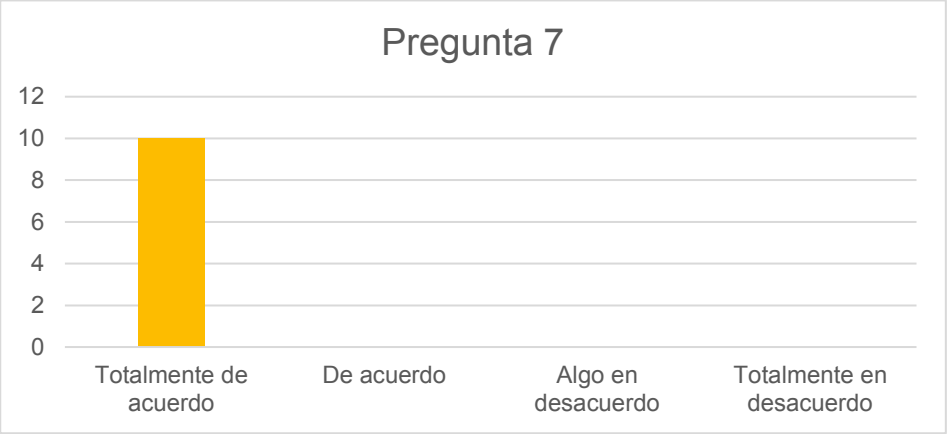

Figura 3. 67 Resultados de la pregunta 7

- **PREGUNTA 8:** ¿Cómo calificaría el nivel de aprendizaje que ha proporcionado el prototipo OFDM?

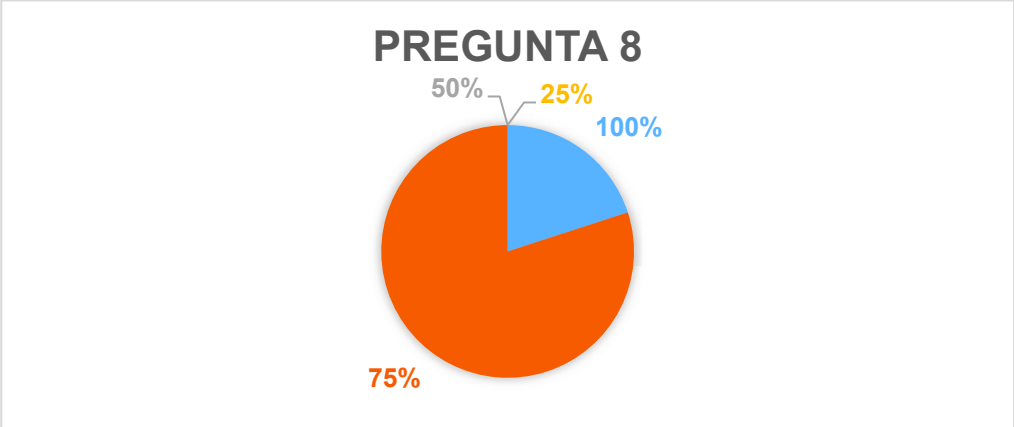

Figura 3. 68 Resultados de la pregunta 8

Al hacer una comparación entre las preguntas realizadas, excluyendo la pregunta 8 cuyas respuestas son diferentes, es posible apreciar de mejor manera los resultados, esto se observa en la figura 3.69.

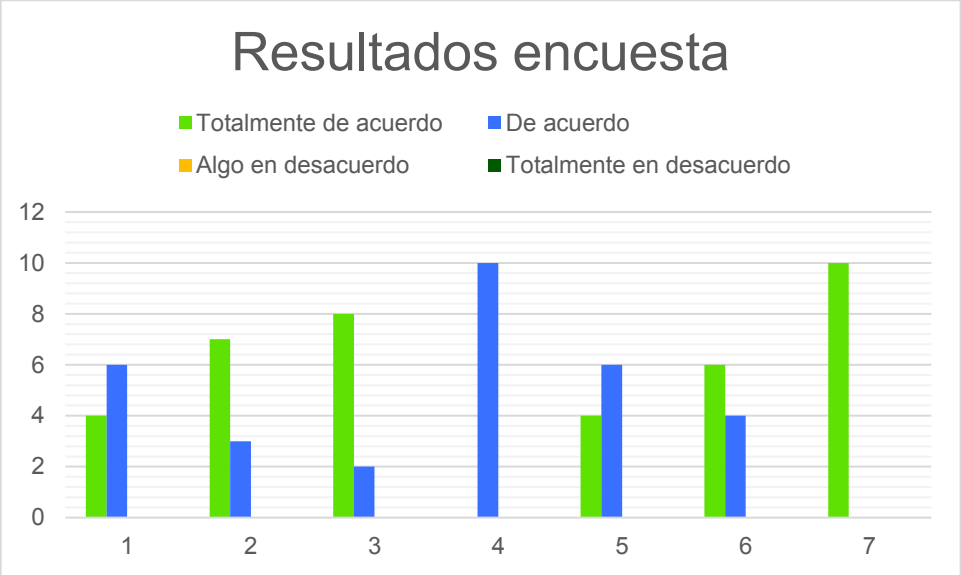

Figura 3. 69 Resultados de la encuesta

- **PREGUNTA 9:** Explique las dificultades del uso que haya encontrado en el prototipo

  RESPUESTAS GENERALIZADAS: El prototipo consume muchos recursos de procesamiento y RAM, razón por lo cual suele detenerse. Además, hay muchas opciones que no se explican cómo usar puesto que son de GNU Radio y existe desconocimiento del uso de este software.

- **Pregunta 10:** Indique las recomendaciones que se debería hacer para mejorar el prototipo

  RESPUESTA GENERALIZADA: se debe dar una explicación breve o un manual de usuario para el uso del prototipo en GNU Radio, a su vez mejorar las prestaciones de los equipos en los que se ejecute el prototipo.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

La interacción entre equipos USRP, Ordenadores, transmisor y receptor OFDM implementados en GNU Radio constituyen el prototipo de sistema de comunicación inalámbrico, este sistema presenta un conjunto de herramientas para la visualización del procesamiento etapa por etapa, permitiendo al usuario observar en tiempo real el procesamiento de la señal.

Al implementarse los prototipos de transmisor y receptor OFDM con los equipos USRP, fue necesario ejecutarlos en ordenadores por separado, esto se debe a los recursos de procesamiento que los programas requieren. Un computador I5 de 6ta Gen y 8Gb de RAM no pudo soportar la carga que produce ejecutar dos programas conectados a dos equipos USRP, además, un equipo USRP no puede soportar el procesamiento para transmisión y recepción a la vez.

Siendo GNU Radio un software de libre y fácil acceso presenta algunos inconvenientes, en especial al momento de implementar los programas desarrollados y presentar los resultados de una manera interactiva, detallada y de fácil utilización. Esto ocasiona que el prototipo presente un nivel moderado de complejidad en su empleo, la complejidad se debe a las limitaciones de las herramientas para mostrar resultados y a la dificultad de utilización de estas herramientas.

GNU Radio presenta bloques utilitarios que permiten reducir el uso del procesamiento en los núcleos de las computadoras, esto ayuda a los prototipos a disminuir en parte el consumo de recursos en las computadoras, sin embargo, a pesar de usar estos bloques el nivel procesamiento es muy alto. Esto se debe a la cantidad de bloques necesarios para la visualización del procesamiento en cada etapa. Finalmente, aun cuando se utiliza estos bloques utilitarios, el consumo de RAM no disminuye.

Debido al uso de los equipos de SDR USRP NI 2910 existe problemas de conectividad con Ubuntu Linux, esto se debe a que el firmware de estos equipos no tiene soporte con GNU Radio en la versión UHD 3.10.7.0 y versiones superiores, sino, solamente en versiones anteriores a esta. Sin embargo, al cambiar la versión del firmware del equipo se estaría perdiendo la garantía de los mismos. Razón por lo cual para el acoplamiento de los equipos USRP a los programas desarrollados en GNU Radio se utilizó el entorno de GNU Radio con soporte para Windows.

A pesar que cada equipo USRP NI 2920 posee dos puertos para conectar antenas, uno de transmisión – recepción y un segundo puerto de exclusivamente de recepción, no es posible transmitir y recibir a la vez desde un mismo equipo. Esto se debe al elevado procesamiento que la tarjeta FPGA de estos equipos debería soportar, también ocasiona que los programas no se ejecuten. Por ello la solución es configurar tanto un equipo USRP para transmisor y otro para recepción, así mismo cada uno conectado a un ordenador diferente de esta manera el procesamiento que cada USRP y ordenador deban soportar se reduce y los programas puedan ejecutarse.

Al momento de ejecutar las pruebas de funcionamiento se puede apreciar en el receptor que el audio de la señal recuperada posee errores, estos errores principalmente se presentan como ruido y sonido entrecortado, se puede claramente apreciar que al aumentar la distancia de transmisión no solo el nivel de potencia se reduce, sino también se reduce la calidad del audio. El prototipo no cuenta con mecanismos de corrección detección de errores, por esta razón la degradación de la señal en recepción se vuelve muy perceptible para el usuario.

Aunque el prototipo tenga especificadas y configuradas características de la capa física del estándar IEEE 802.11a (OFDM), es posible adecuarlo, cambiarlo y reconfigurarlo con el fin de implementar alguna otra tecnología o estándar que emplee OFDM en su capa física sin requerir la visualización de los procesamientos en cada etapa, permitiendo el estudio de algún otro fenómeno, evento u objeto de estudio.

4.2. Recomendaciones

Para el uso y ejecución del prototipo se recomienda utilizar ordenadores que sean compatibles con el UHD de los equipos USRP y que puedan soportar los requerimientos para su utilización, una memoria RAM superior a 4GB y procesadores superiores a los Core i5 de 5ta generación o equivalentes. De esta manera se cuida que los recursos de procesamiento no se saturen y los programas se ejecuten sin inconvenientes.

Es necesario tomar en cuenta la versión del firmware que los equipos USRP tengan, según esto se debe instalar los drivers UHD. Por el cambio en el fabricante de los equipos USRP existen problemas de compatibilidad entre varias versiones del firmware y ciertas versiones de los drivers UHD. El firmware de *National Intrument*s genera problemas de compatibilidad con Ubuntu Linux, se recomienda usar el SO Windows si los equipos USRP tienen cargado el firmware desarrollado por *National Intruments*.

Al ejecutar un archivo en GNU Radio en la plataforma Ubuntu Linux se debe tomar en cuenta que se genera un cambio en el código fuente del archivo, si los drivers UHD no son

compatibles con Ubuntu el código fuente se verá afectado, a pesar que el programa se ejecuta sin problemas en Ubuntu, al momento de ejecutar el mismo archivo en Windows, si se usaron bloques que tienen relación con los drivers UHD, es posible que estos bloques dejen de funcionar, por ello el archivo no se ejecutará. Se recomienda reemplazar dicho bloque con uno similar y de igual configuración para poder ejecutar el archivo en Windows.

Al usar micrófonos para una transmisión en tiempo real de la voz se recomienda usar un periférico externo, de este modo el ruido ambiental se reduce en comparación al ruido ambiental del micrófono incorporado del ordenador.

Cuando se utilicen los equipos USRP conectados a través de un medio guiado, es necesario conectar un atenuador en uno de los terminales del cable, se recomienda hacer caso de esta normativa. Esto se realiza para cuidar la integridad de los equipos, además, *National Instruments* establece el uso del atenuador para el correcto funcionamiento. Sin el uso del atenuador los equipos USRP tienen el riesgo de sufrir una avería en los puertos.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] A. Marwanto, M. A. Sarijari, N. Fisal, S. Kamilah, S. Yusof, and R. A. Rashid, “Experimental Study of OFDM Implementation Utilizing GNU Radio and USRP - SDR,” Kuala Lumpur, Malaysia, 2009.

[2] K. Bansal and T. Vishrant, “OFDM Tranmission and Reception of Packets using GNU-Radio and USRP - Communications Lab Project,” p. 7, 2012.

[3] T. S. Rappaport, *Wireless Communications - Principles and Practice - Book (Theodore S Rappaport).pdf*, 2nd ed. MIchigan: Prentic Hall PTR, 2002.

[4] P. Muñoz Mora, “Introducción a Ofdm,” *Comp. Sist. CP-OFDM con ZP-OFDM*, pp. 1–84, 2006.

[5] L. Jiménez, J. J. Parrado, C. A. Quiza, and C. A. Suárez, “Modulación multiportadora OFDM,” *Ing. ,Ciencia, Investig. Acad. y Desarro. ,Universidad Dist. Fr. José Caldas*, vol. 6, no. 2, pp. 30–34, 2000.

[6] M. Majo Boter, “Design and implementation of an OFDM-based communication system for the GNU Radio platform,” Universität Stuttgart, 2011.

[7] L. Zhang, “Implementation of Wireless Communication based on Software Defined Radio,” UNIVERSIDAD POLITECNICA DE MADRID, E.U.I.T. Telecomunicación UPM, 2013.

[8] K. E. Manolopoulos, K. G. Nakos, D. I. Reisis, and N. G. Vlassopoulos, “Reconfigurable Fast Fourier Transform Architecture for Orthogonal Frequency Division Multiplexing Systems,” 2013.

[9] Emmanuel Candes, “Math 262 / CME 372: Applied Fourier Analysis and Elements of Modern Signal Processing,” *Stanford student webpage*, 2016. [Online]. Available: https://statweb.stanford.edu/~candes/math262/. [Accessed: 11-Jun-2018].

[10] B. Sidney, F. Matteo, J. Steven, P. Markus, and S. Ivan, “Fast fourier transforms,” C. S. Burrus, Ed. Houston, Texas: Rice University, Houston, Texas, 1996.

[11] A. V. Anand, “A brief study of discrete and fast fourier transforms,” p. 11, 1965.

[12] S. Roberts, “Lecture 7 - The Discrete Fourier Transform,” *Oxford Robot. Lect.*, pp. 82–96, 2003.

[13] C. A. Chancay Rojas, Eduardo Luis; Chonillo Ramirez, “DISEÑO, SIMULACIÓN, E IMPLEMENTACIÓN DE LA SINCRONIZACIÓN DE LA PORTADORA Y DE LA

TRAMA EN SISTEMAS OFDM," ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL, Guayaquil Ecuador, 2013.

[14] J. N. Patel and U. D. Dalal, "A comparative performance analysis of OFDM using MATLAB simulation with M-PSK and M-QAM mapping," *Proc. - Int. Conf. Comput. Intell. Multimed. Appl. ICCIMA 2007*, vol. 4, pp. 406–410, 2008.

[15] R. Prasad and F. J. Velez, *WiMAX Networks*. Aalborg Øst, Denmark: Springer ScienceþBusiness Media B.V. 2010, 2010.

[16] E. M. M. Meza, "DISEÑO DE UN SISTEMA DE TRANSMISIÓN/RECEPCIÓN BASADO EN OFDM PARA COMUNICACIONES PLC DE BANDA ANCHA," Pontificia Universidad Católica Del Perú, 2017.

[17] H. Sari, G. Karam, and I. Jeanclaude, "An analysis of orthogonal frequency-division multiplexing for mobile radio applications," *Proc. IEEE Veh. Technol. Conf.*, no. m, pp. 1635–1639, 1994.

[18] P. Calderon Valarezo, "ANALISIS DEL DESEMPEÑO DE LA TECNICA OFDM SOBRE CANALES DISPERSIVOS," ESCUELA POLITECNICA DEL EJERCITO, 2007.

[19] J. Fang, N. Zheng, H. Tan, X. P. Zhu, and B. Huang, "A hybrid CP detection algorithm for DVB-T," *Proc. - 2010 3rd IEEE Int. Conf. Broadband Netw. Multimed. Technol. IC-BNMT2010*, pp. 639–643, 2010.

[20] S. Rathinakumar, B. Radunovic, and M. K. Marina, "CPRecycle : Recycling Cyclic Prefix for Versatile Interference Mitigation in OFDM based Wireless Systems," *CoNEXT*, pp. 67–81, 2016.

[21] Y. Chiu, D. Markovic, H. Tang, and N. Zhang, "OFDM Receiver Design," *Final Rep.*, 2000.

[22] Y. Liu, Z. Tan, H. Hu, L. J. Cimini, and G. Y. Li, "Channel Estimation for OFDM," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 1891–1908, 2014.

[23] A. Polbach i Vinadé, "BLIND CHANNEL ESTIMATION IN OFDM SYSTEMS," UNIVERSITAT POLITECNICA DE CATALUNYA, 2011.

[24] Á. D. Juan Toral, "Prácticas de Laboratorio de Televisión Digital Terrestre basadas en Radio Definido por Software," UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD, 2013.

[25] M. Garcia-pineda, J. Segura-garcia, A. Soriano-asensi, and C. Botella-mascarell, “Uso de Software-Defined Radio en la enseñanza de sistemas de comunicaciones,” *ResearchGate*, no. January 2018, 2017.

[26] V. Rodríguez Abdalá and J. Sanchez Garcia, “Base-band Processing Model for Software Defined Radio , compatible with both IEEE802.11g OFDM and WCDMA standards in the Uplink,” *Ing. Investig. y Tecnol.*, vol. XIII, pp. 9–20, 2012.

[27] E&Q Engineering, “Radio Definida por Software | E&Q Engineering Solutions and Innovation,” *2003*. [Online]. Available: http://www.eqeng.com/site/en/node/38. [Accessed: 05-Jun-2018].

[28] A. F. B. Selva, A. L. G. Reis, K. G. Lenzi, L. G. P. Meloni, and S. E. Barbin, “Introduction to the Software-defined Radio Approach,” vol. 10, no. 1, pp. 1156–1161, 2012.

[29] M. Franco, “Procesamiento digital de señales y radios definidas en software,” pp. 1–6.

[30] A. J. Ángel and T. N. Alonso, “RDS (Radio Definido Por Software). Consideraciones Para Su implementacion en hardware SDR (Software Defined Radio). Considerations for Its Hardware Implementation,” *Telem@tica*, vol. 12, no. 2, pp. 56–68, 2013.

[31] A. Hussain, B. P??r Einarsson, and P. S. Kildal, “MIMO OTA testing of communication system using SDRs in reverberation chamber,” *IEEE Antennas Propag. Mag.*, vol. 57, no. 2, pp. 44–53, 2015.

[32] J. Tomljanović, T. Turina, and E. K. Kurelović, “Motherboard and user experience,” *MIPRO 2013*, pp. 689–694, 2013.

[33] NATIONAL INTRUMENTS, “USRP-2920 SPECIFICATIONS,” 375839C–01, 2017.

[34] The GNU Radio Foundation, “GNU Radio The Free & Open Software Ecosystem.” [Online]. Available: https://www.gnuradio.org/about/. [Accessed: 04-Jun-2018].

[35] J. M. A. Reyes and C. E. S. Forgach, “Un complemento al teorema de nyquist,” *Rev. Mex. Fis. E*, vol. 56, no. 2, pp. 165–171, 2010.

[36] H. O. Boada, *Comunicaciones Digitales basadas en Radio Definida por Software*, 1st editio. Bucaramanga, Colombia, Carrera 27, Calle 9. Ciudad Universitaria: Publicaciones Universidad Industrial de Santander, 2016.

[37] Python Software Foundation., “The Python Standard Library — Python 2.7.15 documentation.” [Online]. Available: https://docs.python.org/2/library/. [Accessed: 04-Jul-2018].

[38] T. M. Schmidl and D. C. Cox, “Robust frequency and timing synchronization for OFDM,” *IEEE Trans. Commun.*, vol. 45, no. 12, pp. 1613–1621, 1997.

6 ANEXOS

ANEXO I. Datasheet VERT900 (Formato Digital)

ANEXO II. Datasheet VERT2450 (Formato Digital)

ANEXO III. Datasheet USRP 2920-Specifications (Formato Digital)

ANEXO IV. Encuestas estudiantiles (Formato Digital)

***NOTA**: los anexos se encuentran adjuntos en el CD*

ORDEN DE EMPASTADO