ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN PROTOTIPO DE COMUNICACIÓN INALÁMBRICA BASADO EN LA CAPA FÍSICA DEL ESTANDAR IEEE 802.15.4 ZIGBEE EN LAS BANDAS DE 868/915 MHZ MEDIANTE SDR

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

CARLOS EDUARDO CATUCUAGO ZURITA

carlos.catucuago@epn.edu.ec

DIRECTOR: Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO

felipe.grijalva@epn.edu.ec

DMQ, febrero 2022

CERTIFICACIONES

Yo,CARLOS EDUARDO CATUCUAGO ZURITA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

CARLOS EDUARDO CATUCUAGO ZURITA

Certifico que el presente trabajo de integración curricular fue desarrollado por CARLOS EDUARDO CATUCUAGO ZURITA, bajo mi supervisión.

Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO
DIRECTOR

DECLARACIÓN DE AUTORÍA

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

CARLOS EDUARDO CATUCUAGO ZURITA
FELIPE LEONEL GRIJALVA ARÈVALO

DEDICATORIA

Este trabajo esta dedicado a:

A mi amada esposa Paola y querido hijo Alonso Sebastián quienes han sido mi inspiración para no darme por vencido y luchar por mis objetivos.

A mis amados Padres Manuel y Celia quienes con su amor y sacrificio me brindaron la hermosa oportunidad de prepararme y ser un profesional.

A mis queridos hermanos y hermanas los cuales aportaron con su granito de arena para llegar a culminar con esta preciada meta.

Carlos Catucuago

AGRADECIMIENTO

Agradezco a mi Padre Celestial, quien a estado a mi lado de manera incondicional y me a dado fuerza para superar los desafíos que he hallado en el camino.

A mis padres y a mi familia por brindarme el consejo , la guía y la inspiración en cada etapa de mi vida.

Al Ph.D. Felipe Grijalva y al Ing. Jorge Carvajal por su tiempo y dirección en el desarrollo y culminación de este proyecto.

Finalmente, a la Escuela Politécnica Nacional por brindarme los conocimientos y habilidades necesarias para el desempeño en mi campo profesional.

Gracias.

Carlos Catucuago

ÍNDICE DE CONTENIDO

RESUMEN

En el presente proyecto de titulación se implementará un prototipo de comunicación inalámbrica basada en la capa física del estándar IEEE 802.15.4, conocido como ZigBee. Para la implementación del sistema se hará uso de las herramientas GNU radio y Radio Definida por Software.

Las bandas de frecuencia escogidas para llevar a cabo los prototipos serán 915/868 MHz. Además, se construirá el transmisor y receptor para la banda de 2450 MHz en entorno de simulación. El proyecto pretende ser una herramienta de apoyo en el ámbito académico para la materia de comunicaciones inalámbricas; esto debido a las bondades que presenta GNU Radio, al mostrar el procesamiento digital de las señales. Se espera que ayude de manera didáctica a la compresión de temáticas como la modulación y demodulación O-QPSK.

Inicialmente con la ayuda de bloques de procesamiento se construirán los transmisores y receptores para las tres bandas por separado. Posterior a eso se formarán bloques jerárquicos con las etapas anteriormente concebidas, para luego ser acopladas y configuradas con los dispositivos USRP.

Posteriormente, se realizarán las pruebas de funcionamiento en entorno de simulación y con los equipos USRP en un ambiente interior controlado. Para este último, la comunicación se llevará a cabo enviando un mensaje desde el transmisor hacia al receptor mediante el uso de una interfaz de aire. Finalmente, se analizarán los resultados de las pruebas y se presentarán las conclusiones y recomendaciones.

PALABRAS CLAVE: GNU Radio, USRP, Radio Definida por Software, O-QPSK, ZigBee.

ABSTRACT

A prototype of wireless communication based on the physical layer of the IEEE 802.15.4 standard, known as ZigBee, will be implemented in this degree project. For the implementation of the system, the GNU Radio and Software Defined Radio tools will be used.

The frequency bands chosen to carry out the prototypes will be 915/868 MHz. In addition, the transmitter and receiver for the 2450 MHz band will be built in a simulation environment. The project aims to be a support tool in the academic field for wireless communications; this because of the benefits presented by GNU Radio, showing the digital processing of signals. It is expected to help didactically the compression of topics such as modulation and demodulation O-QPSK.

Initially with the help of processing blocks will be built the transmitters and receivers for the three bands separately. After that, hierarchical blocks will be formed with the previously conceived stages, to then be coupled and configured with the USRP devices.

Subsequently, operational tests will be carried out in a simulation environment and with the USRP equipment in a controlled indoor environment. For the latter, communication shall be carried out by sending a message from the transmitter to the receiver using an air interface.Finally, test results will be analysed and conclusions and recommendations presented.

KEYWORDS: GNU Radio, USRP, Software Defined Radio, O-QPSK, ZigBee.

1. INTRODUCCIÓN

Desde sus inicios los sistemas de comunicación por radio se han construido en su totalidad en hardware, limitando de esta manera un solo tipo de operación. Una vez establecido el diseño del sistema de radio, mediante el dimensionamiento de parámetros como; el ancho de banda, frecuencia de transmisión, tipo de modulación, ganancia, entre otros, estos no se pueden modificar a menos que se proceda al cambio de módulos en hardware con métricas distintas de los parámetros mencionados.

Con los grandes avances en la tecnología informática hoy en día existen plataformas de desarrollo programables, abiertas y gratuitas, que hacen que la mayoría de los componentes de radio se implementen a través de software, obteniendo flexibilidad, disminución de costos y una gran variedad de ventajas operativas y de enseñanza; SDR (Radio Definida por Software) [1] es una de las tecnologías que permite este tipo de implementación, brindando un gran avance en el estudio de las comunicaciones inalámbricas, ya que se puede utilizar a los SDR como recursos académicos y pedagógicos para consolidar conocimientos adquiridos de manera teórica. A su vez permite experimentar, confirmar y contrastar las características de tecnologías que ya han sido implementadas en Hardware, por ejemplo una de estas tecnologías es ZigBee [2].

ZigBee es una tecnología inalámbrica basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (Wireless Personal Area Network, WPAN) [3]. El estándar opera en las bandas no licenciadas de 2.4 GHz, 915 MHz y 868 MHz; su campo de aplicación está en las redes de sensores de bajo consumo, baja tasa de envío de datos y bajo costo. Actualmente, no existe su implementación en SDR en la Facultad. Por lo que, nace la necesidad de implementar dicho protocolo que es objeto de estudio en las carreras de Telecomunicaciones y Tecnologías de la información; el mismo que llegará a ser un instrumento de enseñanza para el área de comunicaciones inalámbricas ya que permitirá observar el procesamiento de la señal al pasar por las etapas de modulación y codificación que se especifica en dicha tecnología. La modulación en la capa física del estándar IEEE 802.15.4 cuenta con los procesos de conversión de datos PPDU (The PHY Protocol Data Unit) de bit a símbolos, de símbolos a chips y posteriormente la modulación de cada una de las 16 secuencias de chips con modulación por desplazamiento de fase en cuadratura escalonada O-QPSK (Offset Quadrature Phase-Shift Keying); este proceso permitirá observar el envío de un archivo de texto desde un trasmisor hacia un receptor basándose en la capa física

del protocolo ZigBee. Por tanto, el presente trabajo busca implementar un prototipo de comunicación inalámbrica basado en la capa física del estándar IEEE 802.15.4 utilizando el software de programación GNU Radio y como Hardware el dispositivo USRP 2920, permitiendo con esto aprovechar los beneficios y ventajas del uso de la tecnología SDR para el estudio de la capa física del estándar en mención.

1.1. OBJETIVO GENERAL

El objetivo general de este proyecto técnico es implementar un prototipo de sistema de comunicación inalámbrica basado en la capa física del estándar IEEE 802.15.4 ZigBee en las bandas 868/915 MHz mediante GNU Radio y SDR.

1.2. OBJETIVOS ESPECÍFICOS

- Describir las características principales de la capa física del estándar IEEE 802.15.4, SDR y GNU Radio.
- Diseñar e implementar los bloques de procesamiento no disponibles en la librería de GNU Radio.
- Implementar el prototipo de comunicación y acoplarlo a los equipos USRP.
- Analizar resultados obtenidos mediante las pruebas de funcionamiento.

1.3. ALCANCE

Este trabajo implementará un sistema de radio comunicación basado en la capa física del protocolo ZigBee, utilizando GNU Radio y el dispositivo USRP 2920.

El sistema de comunicación constará de un transmisor y receptor implementados por medio de software en computadores personales. El sistema tendrá la posibilidad de ajustar parámetros de trasmisión como la ganancia y el canal de comunicación. En GNU Radio se implementará únicamente la capa física del estándar IEEE 802.15.4.

El transmisor así como el receptor utilizarán un hardware RF externo que ofrece rangos de frecuencias de hasta 2.2 GHz con hasta 20 MHz de ancho de banda instantáneo.

Debido a las limitaciones del dispositivo USRP, en cuanto a la banda de frecuencia, se implementará para las bandas de 2 GHz y 915/868 MHz. De las tres modulaciones posibles

que se especifican en el estándar (O-QPSK, BPSK y ASK), se utilizará la modulación O-QPSK.

El transmisor estará basado en los bloques que se observan en la Figura 1.1. La fuente será un mensaje de texto el cual será trasmitido en pequeños paquetes.

A cada paquete se le añadirá un preámbulo, un delimitador y un campo longitud tanto para el tamaño de la trama y el payload. Seguidamente se convertirá la trama de bits en símbolos; en este proceso los bits serán agrupados en bloques de 4 bits, cada bloque representará un símbolo. El primer símbolo constará de los 4 bits menos significativos del octeto y el siguiente símbolo constará de los 4 bits más significativos. Este proceso se repetirá secuencialmente con cada uno de los octetos del PPDU; iniciando con el preámbulo y terminando con el último octeto de la unidad de datos de servicio de la PHY (PSDU).Una vez obtenido los símbolos de datos, a cada uno de estos símbolos se asignará una secuencia pseudo-aleatoria de 16 bits denominada chip. Finalmente se utilizará la modulación O-QPSK con forma de onda medio pulso seno para modular cada chip.

La velocidad para la unidad de datos en la capa física PPDU será de 250 Kbps y 100 Kbps para las bandas de frecuencia de 2450 MHz ,915 MHz y 868 MHz respectivamente.

Para el transmisor se utilizarán bloques preexistentes planteados en el trabajo de Thomas Schmid [4] y bloques disponibles en la librería de GNU Radio; los bloques definidos por Tomas Schmid están diseñados para operar en la banda de 2450 MHz. Parte del aporte del presente proyecto de titulación será adaptar los bloques de procesamiento para que puedan operar en las bandas de 868/915 MHz. La adaptación de los bloques se realizará modificando parámetros en la interfaz de cada bloque; además se realizará la modificación a nivel de código de ser necesario.

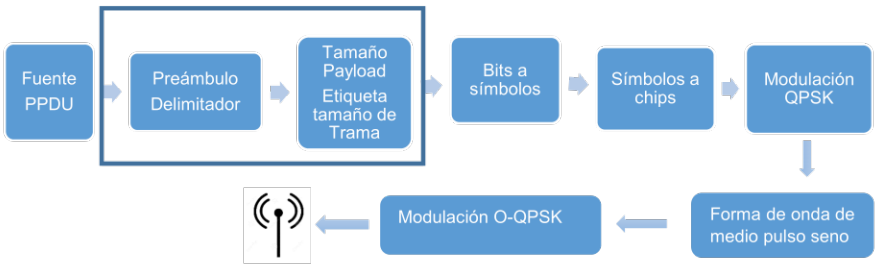

Figura 1.1: Diagrama de bloques sistema transmisor

El receptor realizará el proceso inverso al transmisor; la señal será receptada por el equipo RF USRP 2920, seguidamente pasará por el proceso de demodulación y conversión de

chips a bits para finalmente obtener la trama con el mensaje inicial. La Figura 1.2 muestra mediante un diagrama de bloques el proceso que se llevara a cabo en el receptor.

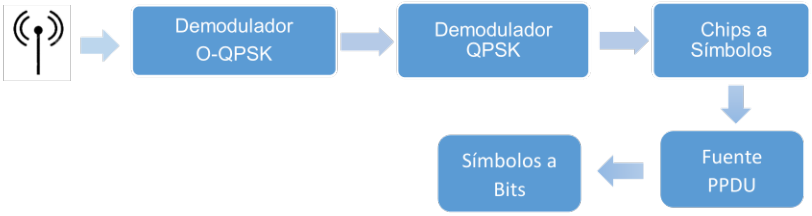

Figura 1.2: Diagrama de bloques sistema receptor

La mayor parte de los bloques de procesamiento de señal que conforman el receptor serán modificados, ya que se busca que los mismos estén sujetos a las especificaciones definidas en el estándar IEEE 802.15.4. En [5] se muestra algunos de los bloques disponibles para el procesamiento de señales.

Las pruebas se realizarán en tres etapas; la primera etapa consistirá en verificar el funcionamiento correcto del sistema transmisor – receptor en un entorno de simulación, esta prueba se realizará en un mismo ordenador y con la ayuda de un canal simulado. La segunda y tercera etapa permitirá verificar y contrastar el funcionamiento de la primera etapa al introducir en el escenario de prueba a los equipos USRP; para este caso se utilizarán dos computadores y dos equipos USRP. En la transmisión se tendrá un computador y un equipo USRP y en la recepción se tendrá un segundo computador con su respectivo equipo USRP. Para esta prueba se variarán parámetros como la distancia entre transmisor y receptor y ganancia.

Una vez implementado los escenarios propuestos, se comprobará la recepción de datos así como la validación cualitativa del enlace; para esto se revisará el parámetro de calidad de enlace LQI (Link Quality Indicator) [3], además de la tasa de errores de paquetes recibidos.

En el presente trabajo se tendrá un producto final demostrable que constará de un prototipo de comunicación transmisor receptor, donde mediante bloques de programación se observará el procesamiento de la señal en la capa física.

1.4. MARCO TEÓRICO

En esta sección se presenta la descripción de las características principales de la capa física del estándar IEEE 802.15.4, con un enfoque en las bandas 2450 MHz y 868/915 MHz. Además de manera general se describe los conceptos de SDR y GNU Radio.

1.4.1. ESTÁNDAR IEEE 802.15 .4

El estándar IEEE 802.15 .4 define el nivel físico y el control del acceso al medio para las tecnologías inalámbricas de bajo consumo de energía y bajas tasas de transmisión de datos. El estándar IEEE 802.15.4 está diseñado principalmente para redes inalámbricas de área personal como son las redes LR- WPAN (Low – Rate Wireless Personal Area Netwok) [6]; siendo, además, la base para la especificación ZigBee. Fue publicado inicialmente en el año 2003 y posteriormente revisado en el 2006; en [3] se presenta la versión 2006, la cual implementa mejoras en la velocidad de datos (20Kbps, 40Kbps, 100kbps y 250 Kbps) para las bandas de frecuencia de 868/915 MHz.

1.4.1.1. Características Generales

En [2],se detalla de manera extendida las características principales entre las cuales se resume las siguientes:

- Velocidad de datos de 250 Kbps, 40 Kbps, 100 Kbps, 40 Kbps y 20 Kbps. Velocidad de símbolo 62.5 Ksímbolo/segundo
- Modos de direccionamiento: dirección corta 16 bits y dirección larga 64bits
- Topología estrella, topología punto a punto y topología árbol de clúster
- Método de acceso al canal CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)
- Detección de energía (ED)
- Indicación de calidad del enlace (LQI)
- La banda de 2450 MHz tiene 16 canales, la banda de 915 MHz cuenta con 30 canales y la banda 868 MHz con 3 canales
- Potencia de transmisión , cerca de 1mW

1.4.1.2. Arquitectura IEEE 802.15.4

Con el afán de simplificar y representar de forma gráfica el estándar, se organiza en términos de bloques basados en el modelo OSI (Open System Interconnection); los cuales representan capas. La Figura 1.3 muestra la Capa física (PHY Physical Layer) y Control de acceso al medio (MAC Medium Access Control); por las que está formado el estándar IEEE 802.15.4.El enfoque de estudio para el presente trabajo estará basado en la capa física.

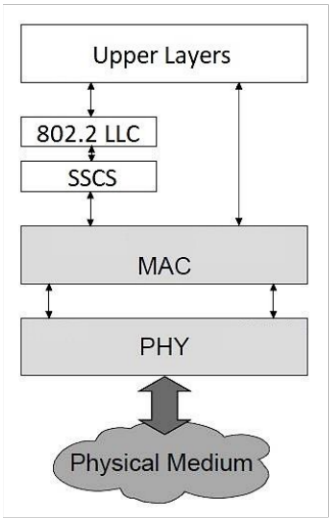

Figura 1.3: Arquitectura del dispositivo LR-WPAN [3]

1.4.2. CAPA FÍSICA PHY

La capa física proporciona principalmente dos servicios: el servicio de datos PHY y el servicio de gestión PHY. El servicio de datos PHY realiza la transmisión y recepción de unidades de datos de protocolo PHY (PPDU) a través de un canal físico de radio. Además la capa física IEEE 802.15.4 es responsable de:

- Activación y desactivación del transceptor de radio
- Transmisión y recepción de datos
- Indicador de calidad (LQI) para paquetes recibidos
- Selección de frecuencia de canal

1.4.2.1. Frecuencias de operación y parámetros de propagación

El estándar IEEE 802.15.4 opera en tres bandas de frecuencia no licenciadas, que son: 868-868.9 MHz, 902-928MHz y 2400-2483MHz. Cada banda de frecuencia cuenta con ve-

locidades de trasmisión de datos, modulación, tipo de dispersión y velocidad de símbolo, los cuales están resumidos en la Tabla 1.1.

Tabla 1.1: Bandas de frecuencia y velocidad de datos [3]

PHY (MHz)	Frequency band (MHz)	Spreading Parameters		Data Parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868-868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902-928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400-2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

1.4.2.2. Asignación de canales

En [3] se definen un total de 27 canales que están disponibles para las tres bandas de frecuencia. Los canales están numerados del 0 al 26; y están designados de la siguiente manera: la banda de 2450 MHz con 16 canales disponibles, la banda de 915 MHz con 10 canales y 868 MHz con 1 canal disponible

La frecuencia central para las diferentes bandas está dada por:

Fc=863.3 MHz ,para K=0

Fc=906 + 2(K-1)MHZ para k=1,2,…,10

Fc=2405 + 5(K-11)MHZ para k=11,12,…,26

Donde: K es el número de canal

1.4.2.3. Formato de la trama PPDU

Tal y como se define en IEEE 802.15.4-2006, la Tabla 1.2 muestra el formato de la unidad de protocolo de datos físico (PPDU) la cual consta de:

Tabla 1.2: Formato de la trama PPDU [7]

		Octeto		
		1 Byte		Variable
Preámbulo	SFD	Longitud de trama (7 bits)	Reservado (1 bit)	PSDU
SHR		**PHR**		**PHY Carga Útil**

- **Encabezado de sincronización (SHR):** ayuda a que un dispositivo receptor pueda lograr la sincronización; contienen los campos de secuencia de preámbulo y delimitador de inicio de trama.
  - **Secuencia de preámbulo:** campo utilizado por el radio receptor para identificar el inicio de la trama y así obtener la sincronización de chip y símbolo.En [3] muestra que la longitud del campo preámbulo es 4 octetos para las tres bandas en estudio.

- **Delimitador de inicio de trama SFD:** este campo permite delimitar el encabezado de sincronización SHR y el paquete de datos. Los SFD para las diferentes capas físicas IEEE 802.15.4 se fija en un octeto.

- **Encabezado de PHY (PHR):** contiene información sobre la longitud de la trama
  - **Longitud de la trama:** campo de 7 bits, que define la longitud del PSDU.
- **PHY Carga útil :** campo de longitud variable

1.4.3. ESPECIFICACIONES PHY PARA LA BANDA 2450 MHz

Inicialmente se revisará las características de la capa física pertenecientes a la banda 2450 MHz para tener una visión mas amplia en la implementación de las bandas de 915/868 MHz. Para esto se partirá del proceso de modulación y dispersión, como lo indica la Figura 1.4.

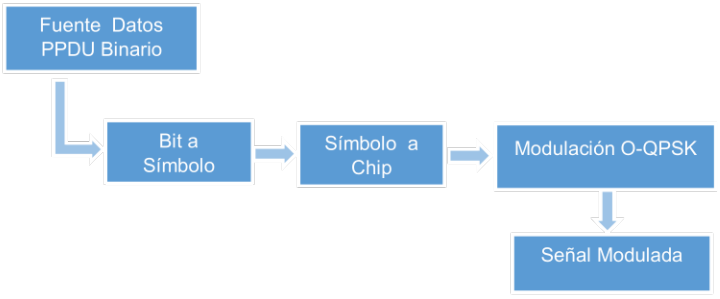

Figura 1.4: Proceso de modulación y dispersión [3]

1.4.3.1. De bits a símbolos

Antes de iniciar el proceso de modulación el dispositivo transmisor reúne los datos a transmitir en un formato apropiado (formato PPDU). Una vez obtenido el mensaje en formato PPDU, se inicia el proceso de conversión de bits a símbolos; para esto los bits del mensaje son agrupados en bloques de 4bits; cada bloque representa un símbolo. Los 4 bits menos significativos LSB (b0, b1, b2, b3) de cada octeto se mapearán en un símbolo de datos y los 4 bits más significativos del octeto MSB (b4, b5, b6, b7) se mapearán en el siguiente símbolo de datos [2]. Este proceso se repite secuencialmente con cada octeto del PPDU, iniciando con el preámbulo y finalizado con el ultimo byte del PSDU.

1.4.3.2. De símbolos a chips

Una vez obtenido los símbolos, cada uno de estos es traducido en un secuencia de ruido pseudo-aleatoria casi ortogonal de 32 bits denominados Chip. Mientras mayor sea el patrón

de chips, mayor será la resistencia de la señal de datos a la interferencia y ruido [7]. A la técnica que realiza este ensanchamiento de los datos antes de que se transmitan se denomina técnica de espectro expandido DSSS. Este tipo de técnicas crean señales que ocupan un ancho de banda más amplio. En el receptor se utiliza la misma secuencia pseudo-aleatoria para la dispersión. En la Tabla 1.3 se muestras los 16 símbolos con sus respectivas secuencias de chip.

Tabla 1.3: Secuencia de chips por cada símbolo de datos [3]

Símbolo de datos (decimal)	Símbolo de datos (binario) (*b0b1b2b3*)	Valores de chip (*c0c1…c30c31*)
0	0000	11011001110000110101001000101110
1	1000	11101101100111000011010100100010
2	0100	00101110110110011100001101010010
3	1100	00100010111011011001110000110101
4	0010	01010010000101110110110011100011
5	1010	00110101001000101110110110011100
6	0110	11000011010100100010111011011001
7	1110	10011100001101010010001011101101
8	0001	10001100100101100000011101111011
9	1001	10111000110010010110000001110111
10	0101	01111011100011001001011000000111
11	1101	01110111101110001100100101100000
12	0011	00000111011110111000110010010110
13	1011	01100000011101111011100011001001
14	0111	10010110000001110111101110001100
15	1111	11001001011000000111011110111000

1.4.3.3. Modulación O-QPSK

La modulación que se especifica en IEEE 802.15.4 para la banda 2450MHz es O-QPSK con forma de pulso de medio seno. O-QPSK es un esquema que modula los bits de cada chip en fase (I) y cuadratura (Q). Los chips pares (C0,C2,C4… C30) se transmiten modulando la portadora de fase (I) y los chips impares(C1,C3,C5… C31) son transmitidos modulando la portadora de fase de cuadratura (Q) con un ángulo de desfase de 90 grados con la señal anterior [7]. La velocidad de chip es 32 veces la velocidad de símbolo, la cual es 2 Mchip / s; y para modular cada chip se utiliza un pulso de medio seno. En la Figura 1.5 se muestra el desplazamiento entre la modulación de chip de fase (I) y la de cuadratura (Q); el cual se forma retrasando los chips de fase en cuadratura (Q) 0.5 us respecto a los chips de fase (I). Este retraso es conocido como Tc.

La modulación O-QPSK limita el salto de fase a no más de 90°, permitiendo que las fluctuaciones de amplitud sean mucho menores a las que presenta QPSK tradicional . El comportamiento de fase de QPSK puede llegar hasta 180° a la vez [8]. La forma de onda de medio pulso seno esta descrita por la siguiente Ecuación 1.1.

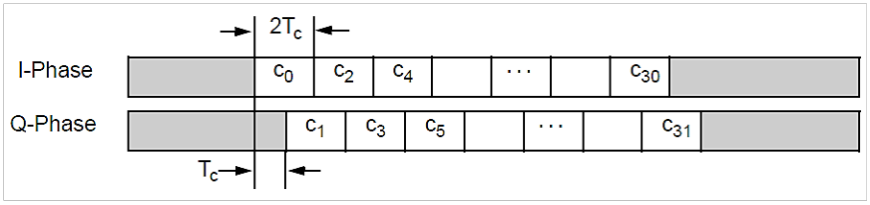

Figura 1.5: Desplazamiento entre chips O-QPSK [3]

$$P(t) = \begin{cases} \text{sen}\,(\pi \frac{t}{2T_c}) & , \quad 0 \leq t \leq 2T_c \\ \\ 0 & , \quad otro\ valor \end{cases} \tag{1.1}$$

La forma de onda medio pulso seno positiva representa “1L”, la forma de onda medio pulso negativa representa el “0L”, tal y como se presenta en la Figura 1.6.

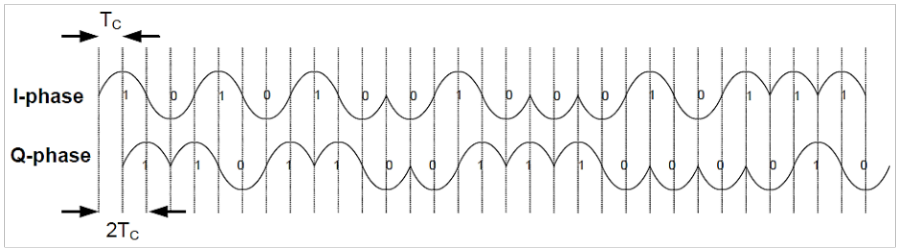

Figura 1.6: Forma de onda medio pulso sinusoidal [3]

La trasmisión de los chips durante cada periodo de símbolo se realiza primero con el chip menos significativo C0 y finaliza con el C31.

1.4.4. ESPECIFICACIONES PHY PARA LA BANDA 868/915 MHz

1.4.4.1. Modulación y dispersión.

Al igual que en la banda de frecuencia 2450 MHZ el tipo de modulación utilizado es O-QPSK. El proceso de conversión de bits a símbolos permanece inalterable; el cambio radica en la conversión de símbolos a chips ya que para las bandas de 868/915 MHz a cada símbolo de datos se le asignará una secuencia pseudo-aleatoria de 16 chips, tal y como se muestra en la Tabla 1.4.

Tabla 1.4: Secuencia de chips para las bandas 868 y 915 MHz [3]

Símbolo de datos (decimal)	Símbolo de datos (binario) (*b_0 \ b_1 \ b_2 \ b_3*)	Valores de chip (*c_0 \ c_1…c_14 \ c_15*)
0	0000	0011111000100101
1	1000	0100111110001001
2	0100	0101001111100010
3	1100	1001010011111000
4	0010	0010010100111110
5	1010	1000100101001111
6	0110	1110001001010011
7	1110	1111100010010100
8	0001	0110101101110000
9	1001	0001101011011100
10	0101	0000011010110111
11	1101	1100000110101101
12	0011	0111000001101011
13	1011	1101110000011010
14	0111	1011011100000110
15	1111	1010110111000001

Los chips pares se transmiten modulando la portadora en fase (I) y los chips impares se trasmiten modulando la portadora en fase de cuadratura (Q). La velocidad de chip nominal es 400 Kchip/s para la banda de 868 MHz y 1.0 Mchip/s para las bandas de 915 MHz. Los chips de fase Q se retrasan Tc con respecto a los chips de fase I, esto para formar un desplazamiento entre las modulaciones de fase de chip como se puede ver en la Figura 1.7.Tc representa el inverso de la velocidad de chip.

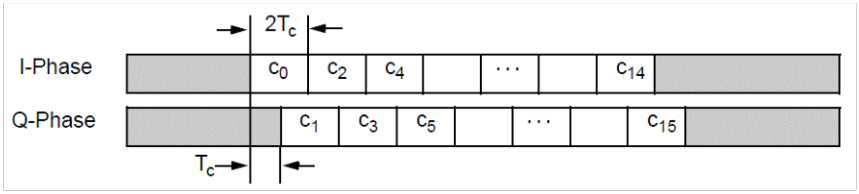

Figura 1.7: Desplazamiento entre chips O-QPSK para 868 y 915 MHz [3]

Al igual que en la banda 2450 MHz el orden de transmisión de los chips, comienza con el chip menos significativos C0 y termina la transmisión del chip más significativos C15 ; esto mientras trascurre periodo de duración de cada símbolo.

1.4.4.2. Filtro Coseno Levantado para la banda 868 MHz

Antes de realizar la transmisión en la banda 868MHz la señal debe ser filtrada, esto para regular la densidad espectral de potencia PSD. Para filtrar la señal se necesita que el mismo se aproxime a un filtro coseno levantado ideal, con un factor de corrección de r = 0.2 [3]. Para esto se especifica la Ecuación 1.2.

$$P(t) = \begin{cases} \frac{\text{sen}\,(\pi \frac{t}{T_c})}{\pi \frac{t}{T_c}} \frac{\cos\,(\pi \frac{t}{T_c})}{1-\frac{4r^2\ t^2}{T_c^2}} & , \quad t \neq 0 \\ \\ 1 & , \quad t = 0 \end{cases} \tag{1.2}$$

1.4.5. RADIO DEFINIDA POR SOFTWARE SDR

Un SDR se define como una clase de radio reconfigurable y programable, donde sus características de capa física pueden ser modificadas a través de software [9]. A nivel de software SDR, permite la implementación de varias funciones operativas de radio en banda base como: tipo de modulación, corrección de errores, control sobre la frecuencia de portadora, entre otros. Los parámetros operativos de estos bloques funcionales pueden ser modificados potencialmente en tiempo real y ajustado con la ayuda de un operador humano o un proceso automatizado. Un dispositivo SDR está compuesto por arreglo de compuertas programables (FPGA), procesadores de señal digital (DSP), procesadores de señal de propósito general (GPP), etc. En [10, 11] se explica que estas tecnologías permiten, modificar o agregar nuevas funciones y capacidades a los sistemas de radio existentes sin tener que hacer el uso de un nuevo hardware. A diferencia de los sistemas de radio tradicionales basados en hardware, limitados en funcionalidades y que únicamente pueden modificarse mediante el cambio físico de los mismos.Las características mas relevantes se mencionan a continuación.

- Funcionalidad múltiple: pueden soportar múltiples funciones de radio mediante el uso de una misma plataforma de comunicación de radio digital.
- Costo de desarrollo: gracias a la reutilización de software en diferentes tecnologías de radio el coste de desarrollo se reduce ampliamente.
- Actualizaciones Over-The-Air: la tecnología SDR permite hacer la corrección de fallas y actualizaciones de forma remota. Esto mientras el sistema de radio está en funcionamiento.

1.4.6. UNIVERSAL SOFTWARE RADIO PERIPHERAL (USRP)

Los dispositivos USRP son Radios Definidas por Software que ayudan al diseño y creación de prototipos de sistemas de comunicación inalámbrica complejos de manera rápida. Estos aparatos permiten la trasmisión y recepción de señales RF en diferentes bandas y son utilizadas a nivel de educación e investigación en comunicaciones.

Los USRP proporcionan plataformas de radio frecuencia reconfigurable gracias al procesamiento digital de señales personalizado. Estos dispositivos están constituidos por procesadores basados en hosts, FPGA y RF front ends. Los FPGA integrados poseen una alta capacidad de procesamiento, ideal para aplicaciones que cuentan con un gran ancho de banda y con la necesidad de procesamiento en tiempo real. Los dispositivos USRP además pueden ser utilizados en aplicaciones relacionadas con el monitoreo espectral. National Instruments (NI) en [12] ofrece una gran variedad de USRP, los cuales se comercializan como unidades cerradas, ensambladas en su totalidad y completamente probadas.

1.4.6.1. NI USRP 2920

El dispositivo USRP 2920 trabaja con un ancho de banda de 20 MHz y un rango de frecuencia desde 50MHz a 2.2 GHz.

El transceptor USRP 2920 presenta una arquitectura en hardware común a SDR. El dispositivo cuenta con convertidores analógico-digital ADC (Analog to digital converter) de alta velocidad y con convertidores digital –analógicos DAC (Digital to analog converter). Además cuenta con circuito integrado FPGA de personalidad fija que puede ser configurado luego de la fabricación del mismo. La Figura 1.8 muestra el diagrama del hardware USRP 2920. El proceso en el receptor USRP 2920 inicia con un front- end analógico con alta sensibilidad, capaz de captar señales de baja intensidad y digitalizarlas a señales de banda base I y cuadratura Q mediante la conversión analógica–digital de alta velocidad. Posteriormente el DDC reduce la frecuencia y empaqueta las señales I y Q para ser trasmitidas y procesadas para un host mediante Gigabit Ethernet. El proceso en el transmisor inicia con la generación de las señales I y Q en el computador. Posteriormente son trasladadas al dispositivo USRP mediante el cable Ethernet. La conversión digital –analógico se realiza por el DAC y se produce la mezcla de I y Q para convertir la señal a una frecuencia RF, finalmente la señal es amplificada y trasmitida [12].

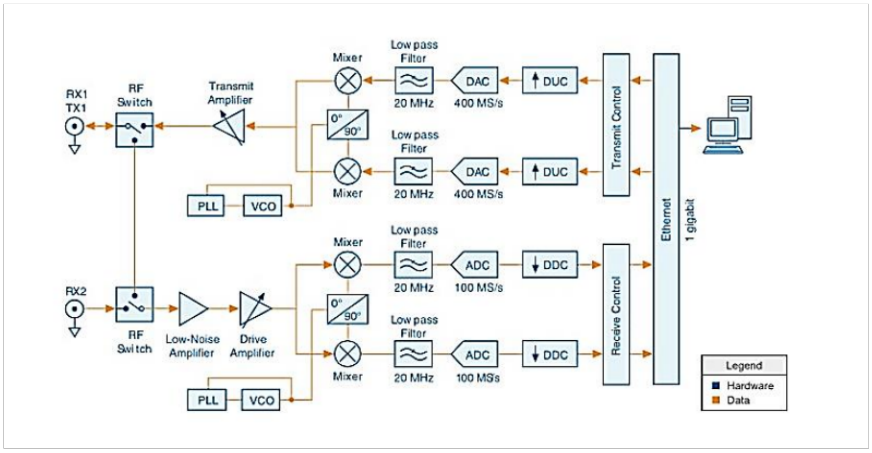

Figura 1.8: Arquitectura en Hardware del USRP 2920

1.4.7. GNU RADIO

GNU Radio es una plataforma de software libre y gratuito, que utiliza bloques de procesamiento de señales (DSP) para implementar la tecnología de Radio Definida por Software SDR . Además, GNU Radio permite la utilización de un hardware RF externo para implementar tecnologías inalámbricas complejas a un bajo costo [13]. La utilización más frecuente de esta plataforma se encuentra en los campos académicos, militares, aficionados y comerciales.

GNU Radio ofrece un entorno gráfico fácil de usar, el cual permite construir diagramas de flujo con bloques previamente definidos o a su vez bloques nuevos construidos por el programador, los cuales son diseñados para aplicaciones específicas mediante lenguajes de programación Python y C++.GNU Radio permite a los usuarios el acceso a proyectos y librerías que ya han sido implementados gracias a la comunidad de código abierto.

1.4.7.1. Programación en GNU Radio

En GNU Radio las aplicaciones son desarrolladas en Python y C++. Los bloques de procesamiento de señales generalmente son implementados en C++. Para escribir las aplicaciones, construcción de bloques jerárquicos e interconexión entre bloques de procesamiento se utiliza Python. SWIG (Simplified Wrapper and Interface Generator) es la herramienta utilizada como interfaz, para acceder desde Python a las funciones de procesamiento implementadas por C++. Además, para la creación de aplicaciones en GNU Radio se puede utilizar GNU Radio Companion (GRC), que es una herramienta gráfica basada en bloques que previamente han sido definidos en la librería [14]. Una vez creada la aplicación; GRC automáticamente

genera el código en Python de la misma, permitiendo su visualización y modificación. En la Figura 1.9 se muestra la arquitectura del software GNU Radio.

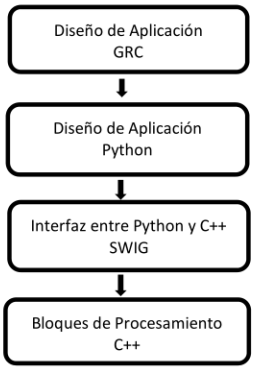

Figura 1.9: Arquitectura GNU Radio

1.4.7.2. GNU Radio Companion (GRC)

GNU Radio Companion se creó con la misión de facilitar el uso de GNU Radio, mediante el uso de bloques gráficos basados en Python y C++ en lugar de implementar el procesamiento únicamente mediante código.

1.4.7.3. Interfaz Gráfica

La interfaz gráfica cuenta con cinco partes: librería, terminal, variables, área de trabajo y barra de herramientas como se muestra en la Figura 1.10.

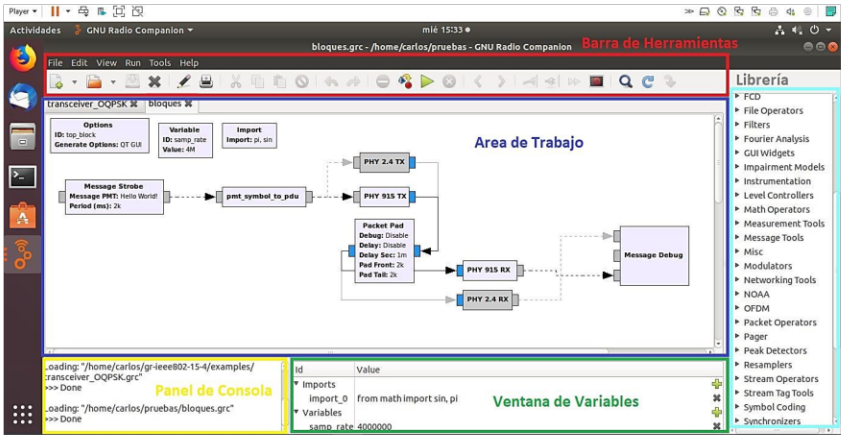

Figura 1.10: Interfaz gráfica GRC

1.4.7.4. Bloques

GNU Radio cuenta con una gran variedad de bloques que han sido creados para aplicaciones específicas. La sección librería organiza estos bloques por categoría, entre los cuales se encuentra:

- Waveform Generators
- Modulators
- Instrumentation
- Math Operators
- Channel Models
- Filters
- Fourier Analysis

Estos bloques permiten la realización de tareas estándar en la manipulación digital de señales. Mediante su combinación se pueden crear diagramas de flujo que simulen aplicaciones de sistemas de radio [15].

A partir de los bloques previamente definidos se puede crear bloques jerárquicos. Estos bloques brindan modularidad a los diagramas de flujo y a la vez proporciona bloques simples y flexibles al cambio. Al igual que en los bloques normales los bloques jerárquicos cuentan con entradas y salidas. Para la creación de un bloque jerárquico se utilizan principalmente cuatro bloques: Options block, Parameter blocks, Pad source y Pad sink [16].

2. METODOLOGÍA

En esta sección se describe el diseño e implementación del prototipo de comunicaciones basado en la capa física del estándar IEEE 802.15.4 mediante la utilización de GNU Radio y los dispositivos USRP. La Figura 2.1 muestra las 4 etapas en las que se divide el desarrollo del proyecto de titulación; de primera mano se realizará la generación del mensaje en bits. Posteriormente, se hará la implementación del sistema transmisor y receptor. Finalmente, se realizará el acoplamiento de las etapas anteriores y la configuración de equipos USRP.

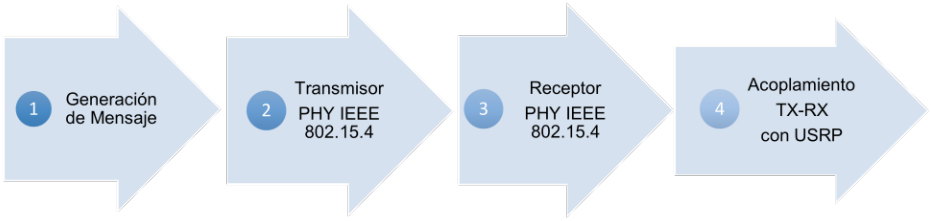

Figura 2.1: Esquema de implementación sistema transmisor - receptor

Para la implementación de las etapas 2 y 3 en las bandas de 915 MHz y 868 MHz se toma como referencia el trabajo de implementación de la capa física de la banda de 2450 MHz; por tanto, inicialmente se conocerá la función que desempeña cada bloque de procesamiento utilizado para la elaboración del prototipo de comunicación de la banda 2450 MHz. Posterior a eso; se realizarán los cambios necesarios para implementar el prototipo en las bandas de 915 MHz y 868 MHz, ya que la modulación y codificación utilizada en las 3 bandas de frecuencia es O-QPSK y DSSS.

2.1. GENERACIÓN DEL MENSAJE

El mensaje a transmitir será una archivo de texto plano en formato .txt con un tamaño de acuerdo a la capacidad de procesamiento de las etapas subsiguientes. La generación del mensaje inicia con el bloque *File Source* el cual permite la selección del archivo a enviar así como la característica de repetir la transmisión del mismo; posteriormente el flujo binario es empaquetado en streams de 64 bytes por el bloque *Stream to Tagged Estream* y etiquetado por el bloque *Tagged stream to PDU*, para finalmente ser transmitido. La Figura 2.27 muestra la etapa de generación del mensaje así como su acoplamiento a las etapas de transmisión y recepción que se presentan en las siguientes secciones.

2.2. IMPLEMENTACIÓN DEL SISTEMA TRANSMISOR

El transmisor en las tres bandas de estudio comparten de manera general el mismo procesamiento de señal; ya que todas las bandas utilizan la modulación O-QPSK especificada en la Sección 1.4.3. La Figura 2.2 permite observar el proceso de implementación del transmisor.

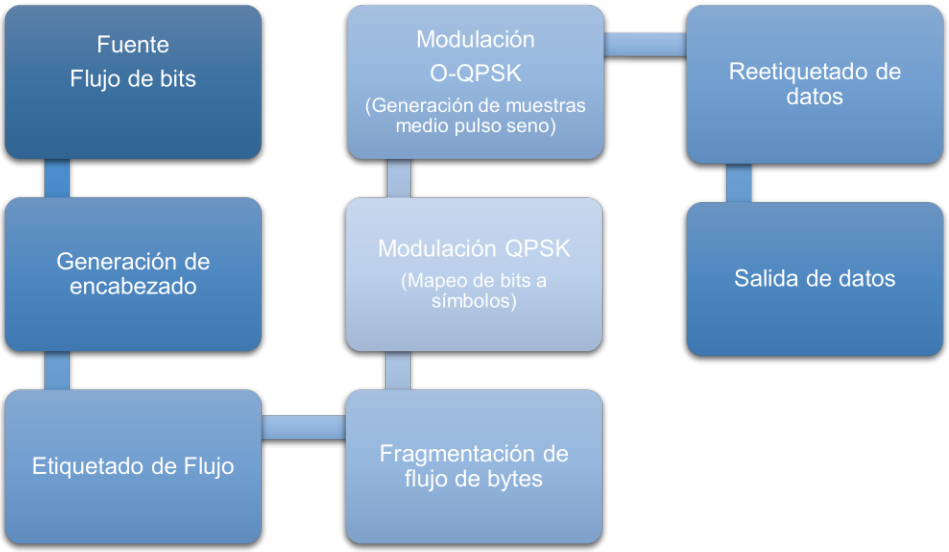

Figura 2.2: Proceso a seguir para implementar el sistema transmisor

2.2.1. TRANSMISOR BANDA 2450 MHz

Para el desarrollo del sistema transmisor se toma como referencia el trabajo de Tomas Schmid [4] y las especificaciones contenidas en el estándar IEEE 802.15.4-2006 [3]. El diagrama de procesamiento de señal fue construido en el entorno GRC generando un archivo ".grc" por cada diagrama de procesamiento, el cual será acoplado a las etapas subsecuentes. Por las limitaciones de los equipos USRP en cuanto a la frecuencia de operación, el prototipo de comunicación en la banda 2450 MHz será implementado únicamente en un entorno de simulación. La Figura 2.3 presenta los bloques de procesamiento implementados en el transmisor.

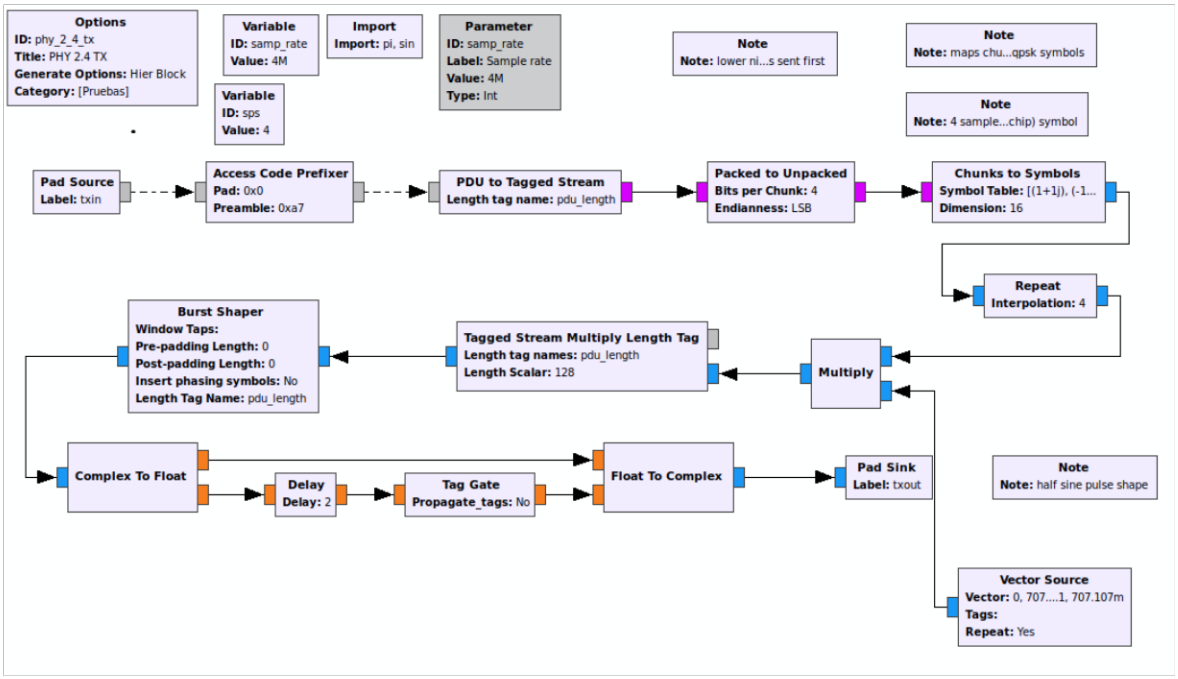

Figura 2.3: Implementación del transmisor en la banda de 2.4 MHz

2.2.1.1. Entrada y salida de bloques jerárquicos

Los bloques *Pad Source* y *Pad Sink* son usados para la construcción de los bloques jerárquicos. La salida del bloque *Pad Source* se convierte en la entrada del diagrama de flujo en un bloque jerárquico. La entrada del bloque *Pad Sink* a su vez, se convierte en la salida del bloque jerárquico. Cada bloque presenta un campo *etiqueta* para identificar la entrada y salida del bloque jerárquico.

2.2.1.2. Generación de encabezado de sincronización SHR

La inserción del campo preámbulo y delimitador de inicio de trama está dado por el bloque *Access Code Prefixer*. El campo preámbulo tiene una longitud de 4 bytes (cada byte de 0Ls) y el campo SDF de un byte.

Para ejemplificar el papel que desempeña el bloque generador, se muestra un ejemplo de trama PPDU donde se puede observar en especial el preámbulo y delimitador de inicio de trama. El ejemplo se muestra en la Tabla 2.1.

Tabla 2.1: Ejemplo de formato de trama PPDU.

4 bytes	1 byte	1 byte		Variable
Preámbulo	SDF	Longitud de la trama 7 bits	Reservado 1 bit	PSDU
00 00 00 00	A7	32	1	EC 48 18 00 00 D0 6F D9...

2.2.1.3. Etiquetado del flujo PPDU

El bloque *PDU to Tagged Stream* ayuda a que las PPDU receptadas se conviertan en flujos etiquetados. La etiqueta que se inserta en el PPDU especifica la longitud del paquete [15]. El parámetro presente en el bloque es el nombre de la etiqueta de longitud. Para el presente proyecto el nombre de la etiqueta es *pdu_length*, como se observa en la Figura 2.4.

Figura 2.4: Bloque de etiquetado de flujo

2.2.1.4. Empaquetado y fragmentación de bytes

Seguidamente el flujo etiquetado ingresa al bloque *Packed to Unpacked*. El bloque *Packed to Unpacked* fragmenta el flujo empaquetado de bytes a la entrada en grupos cortos de bits a la salida. Cada fragmento se coloca a la derecha del byte de salida reservado para cada pedazo (chunk). El proceso de fragmentación de los bytes de entrada se realiza comenzando por los MSB (Most Significant bit) o LSB (Least significant Bit). Para el presente trabajo el parámetro que nos indica el orden de extracción de los bits es *Endianness* y es igual a LSB tal y como se pude ver en la Figura 2.5. Además se tiene el parámetro *bits per chunk* que especifica el número de bits por cada byte de salida; para la banda en análisis el parámetro *bits per chunk* es de 4 [17].

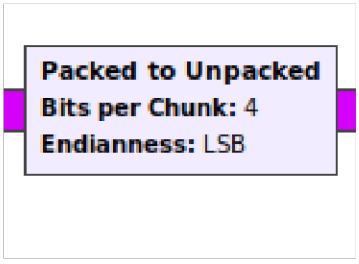

Figura 2.5: Bloque de fragmentación de bytes

2.2.1.5. Modulación QPSK

El bloque *Chunks to Symbols* recibe a la entrada los flujos de bytes fragmentados y a la salida se obtiene un stream de complejos [17]. En la Sección 1.4.3.3 se especifica que cada bloque de 4 bits representa un símbolo de datos. Según se detalla en el estándar IEEE 802.15.4-2006, para la capa física de 2450 MHz se definen 16 símbolos de datos. En la Tabla 2.2 a cada símbolo de datos se asigna una secuencia pseudo-aleatoria de 32 bits denominados chips. Posteriormente, cada par de bits es sometido a la modulación QPSK, donde a cada par de bits se le asigna una muestra (I+jQ); obteniendo así 16 muestras complejas por cada símbolo de datos. Las muestras pares corresponden a símbolos en fase (I) y las impares a símbolos en cuadratura (Q). Para realizar la modulación QPSK se utiliza codificación Gray.

Tabla 2.2: Chips como muestras complejas (I+JQ) para la banda 2450 MHz [3]

Símbolo	Chip (decimal)	Muestras (I + JQ)
0	3653456430	(1 + 1j), (-1 + 1j), (1-1j), (-1 + 1j), (1 + 1j), (-1-1j), (-1-1j), (1 + 1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1-1j), (-1-1j), (1-1j), (1 + 1j), (1 -1j),
1	3986437410	(1-1j), (-1-1j), (1 + 1j), (-1-1j), (1-1j), (-1 + 1j), (-1 + 1j), (1-1j), (-1-1j), (-1-1j), (-1 + 1j), (1 + 1j), (-1 + 1j), (1 + 1j), (1-1j), (1 + 1j),
2	786023250	(-1 + 1j), (-1 + 1j), (-1-1j), (1-1j), (-1-1j), (1-1j), (1 + 1j), (1-1j), (1 + 1j), (-1 + 1j), (1-1j), (-1 + 1j), (1 + 1j), (-1-1j), (-1-1j), (1 + 1j),
3	585997365	(-1-1j), (-1-1j), (-1 + 1j), (1 + 1j), (-1 + 1j), (1 + 1j), (1-1j), (1 + 1j), (1-1j), (-1-1j), (1 + 1j), (-1-1j), (1-1j), (-1 + 1j), (-1 + 1j), (1 -1j),
4	1378802115	(-1-1j), (1-1j), (1 + 1j), (1-1j), (1 + 1j), (-1 + 1j), (1-1j), (-1 + 1j) , (1 + 1j), (-1-1j), (-1-1j), (1 + 1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1 -1j),
5	891481500	(-1 + 1j), (1 + 1j), (1-1j), (1 + 1j), (1-1j), (-1-1j), (1 + 1j), (-1-1j) , (1-1j), (-1 + 1j), (-1 + 1j), (1-1j), (-1-1j), (-1-1j), (-1 + 1j), (1 + 1j),
6	3276943065	(1 + 1j), (-1-1j), (-1-1j), (1 + 1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1- 1j), (-1-1j), (1-1j), (1 + 1j), (1-1j), (1 + 1j), (-1 + 1j), (1-1j), (-1 + 1j),
7	2620728045	(1-1j), (-1 + 1j), (-1 + 1j), (1-1j), (-1-1j), (-1-1j), (-1 + 1j), (1+ 1j), (-1 + 1j), (1 + 1j), (1-1j), (1 + 1j), (1-1j), (-1-1j), (1 + 1j), (-1 -1j),
8	2358642555	(1 + 1j), (1-1j), (1 + 1j), (-1 + 1j), (1-1j), (-1 + 1j), (1 + 1j), (-1-1j) , (-1-1j), (1 + 1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1-1j), (-1-1j), (1 -1j),
9	3100205175	(1-1j), (1 + 1j), (1-1j), (-1-1j), (1 + 1j), (-1-1j), (1-1j), (-1 + 1j) , (-1 + 1j), (1-1j), (-1-1j), (-1-1j), (-1 + 1j), (1 + 1j), (-1 + 1j), (1 + 1j),
10	2072811015	(-1-1j), (1 + 1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1-1j), (-1-1j), (1- 1j), (1 + 1j), (1-1j), (1 + 1j), (-1 + 1j), (1-1j), (-1 + 1j), (1 + 1j), (-1 -1j),
11	2008598880	(-1 + 1j), (1-1j), (-1-1j), (-1-1j), (-1 + 1j), (1 + 1j), (-1 + 1j), (1+ 1j), (1-1j), (1 + 1j), (1-1j), (-1-1j), (1 + 1j), (-1-1j), (1-1j), (-1 + 1j),
12	125537430	(-1-1j), (1-1j), (-1-1j), (1-1j), (1 + 1j), (1-1j), (1 + 1j), (-1 + 1j) , (1-1j), (-1 + 1j), (1 + 1j), (-1-1j), (-1-1j), (1 + 1j), (-1 + 1j), (-1 + 1j),
13	1618458825	(-1 + 1j), (1 + 1j), (-1 + 1j), (1 + 1j), (1-1j), (1 + 1j), (1-1j), (-1-1j) , (1 + 1j), (-1-1j), (1-1j), (-1 + 1j), (-1 + 1j), (1-1j), (-1-1j), (-1 -1j),
14	2517072780	(1-1j), (-1 + 1j), (1 + 1j), (-1-1j), (-1-1j), (1 + 1j), (-1 + 1j), (-1+ 1j), (-1-1j), (1-1j), (-1-1j), (1-1j), (1 + 1j), (1-1j), (1 + 1j), (-1 + 1j),
15	3378542520	(1 + 1j), (-1-1j), (1-1j), (-1 + 1j), (-1 + 1j), (1-1j), (-1-1j), (-1- 1j), (-1 + 1j), (1 + 1j), (-1 + 1j), (1 + 1j), (1-1j), (1 + 1j), (1-1j), (-1 -1j)

Entre los parámetros más relevantes del bloque *Chunks to Symbols* se tiene:

- **Symbol Table:** permite la inserción de una tabla de muestras complejas (I+jQ) que se asignarán a cada fragmento (chunk).
- **Dimension:** especifica la dimensión de la tabla; para la banda de 2450 MHz la dimensión es de 16.
- **Number of Ports:**define el número de flujo a la entrada y salida que se va a procesar; es necesario que cada flujo esté separado uno del otro.

La señal QPSK resultante se observa en la Figura 2.6 ; la cual se obtuvo a la salida del bloque Chunks to Symbols.

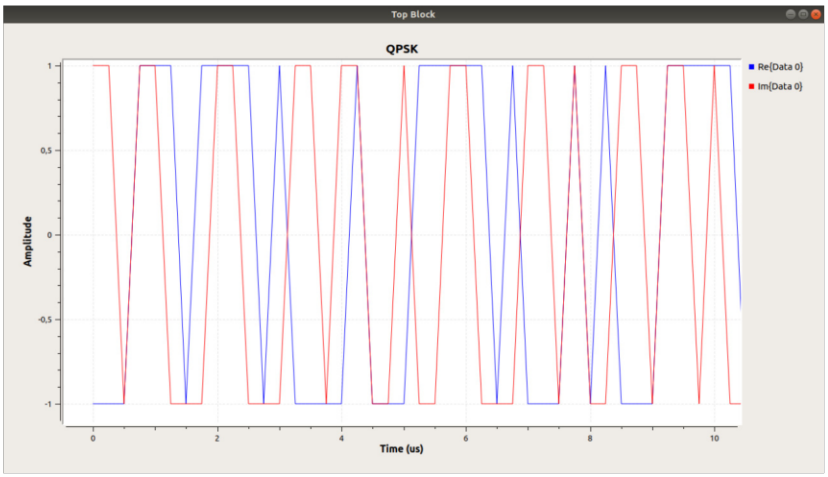

Figura 2.6: Señal QPSK

2.2.1.6. Repetición de número de muestras por símbolo.

El bloque *Repeat* repite la secuencia de muestras complejas pertenecientes a cada símbolo de acuerdo al número de veces que se define en el parámetro interpolación. Para la banda de 2450 MHz el parámetro es cuatro.

La Ecuación 2.1 muestra el cálculo del parámetro de repetición de muestras.

$$Muestra_{simb} = \frac{velocidad\ muestreo\left[M\frac{sim}{seg}\right]}{velocidad\ simbolo\left[M\frac{sim}{seg}\right]} \tag{2.1}$$

Datos:

-Velocidad del estándar: 250 Kbps

-Bits por chunk: 4

-Chips por cada símbolo: 32

Nota: por cada 2 chips se tiene un símbolo QPSK

Cálculo de velocidad de símbolo

Velocidad de chip $= \frac{250kbps}{4bits} \times 32\ chips$

Velocidad de chip $2\frac{Mchips}{seg}$

Por tanto;

Velocidad de símbolo $2\frac{Mchips}{seg} \times \frac{1\ simbolo\ QPSK}{2\ chips}$

Velocidad de símbolo $= 1\frac{Msimb}{seg}$

Finalmente

$Muestra_{simb} = \frac{4\frac{Msimb}{seg}}{1\frac{Msimb}{seg}}$

$Muestra_{simb} = 4$

2.2.1.7. Generación de muestras medio pulso seno

De forma paralela el bloque *Vector Source* genera vectores con muestras de medio pulso seno por cada señal (I) y (Q). Se generan 4 muestras:

$$sin(\frac{\pi}{4}), sin(\frac{3\pi}{4}), sin(\frac{5\pi}{4}), sin(\frac{7\pi}{4})$$

En la modulación O-QPSK las transiciones de fase de las señales I y Q son escalonadas. El cambio de fase máxima se realiza cada π/2 .El parámetro principal del bloque *Vector Source* es *Repeat*; el cual indica si se quiere que las 4 muestras se repitan continuamente. Una vez obtenidas las salidas de los bloques *Repeat* y *Vector Source*, estos flujos pasan al bloque *Multiply* para formar la señal con forma de onda medio pulso seno como se observa en la Figura 2.7.

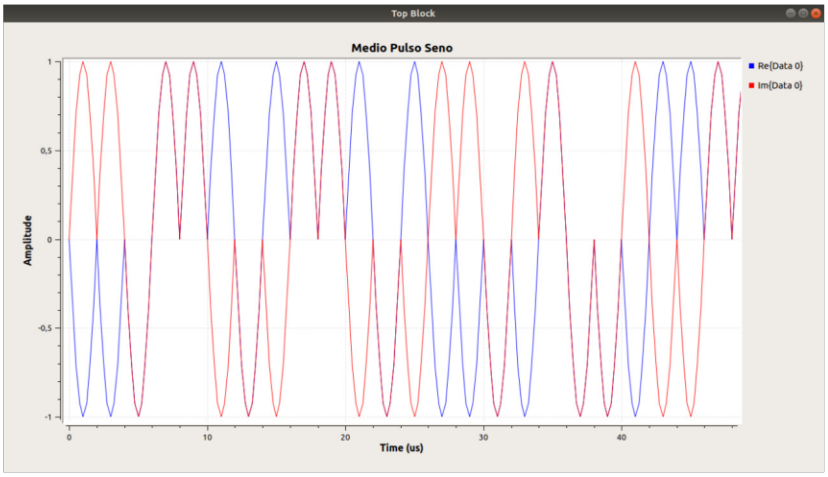

Figura 2.7: Señal con muestras medio pulso seno

2.2.1.8. Corrección de etiqueta

El bloque *Tagged Stream Multiply Length Tag* permite corregir la posición de las etiquetas inicialmente insertadas, ya que en los bloques anteriores se insertaron muestras adicionales. Los parámetros del bloque son: nombre de la etiqueta de longitud y longitud del escalar [17]. El nombre de la etiqueta de longitud fue asignado en el bloque *PDU to Tagged Stream* inicialmente. Para la corrección de la etiqueta se utiliza el escalar 128, el cual es el resultado de multiplicar, el número de muestras complejas por símbolo de datos (16) por el factor de repetición de muestras (4) y por dos (16x4x2 = 128) para la banda 2450 MHz.El bloque encargado de realizar el procedimiento antes mencionado, se muestra en la Figura 2.8.

Figura 2.8: Bloque corrector de etiqueta

La Tabla 2.3 presenta las variaciones de longitud de la etiqueta de paquete , así como el retardo de muestras Q respecto a las muestras I, esto debido al cambio en la tasa de muestreo.

Tabla 2.3: Variación de tasa de muestreo [18]

Tasa de Muestreo	1	2	4	5	6	8	10	12	16
Muestras por símbolo I y Q	1	2	4	5	6	8	10	12	16
Muestras de retardo	0,5	1	2	2,5	3	4	5	6	8
Muestras Reloj M&M	0,5	1	2	2,5	3	4	5	6	8
Longitud de la PDU Múltiplo	32	64	128	128	160	192	256	384	512

Seguidamente la secuencia ingresa al bloque *Complex To Float* para ser dividida en dos flujos de salida, una salida real y la otra imaginaria.

2.2.1.9. Inserción de retardo y conformación del símbolo O-QPSK

Posteriormente, la salida imaginaria es retardada para crear el desfase de la señal Q respecto a la señal I, especificado en la modulación O-QPSK. La señal Q está desfasada medio tiempo de símbolo respecto de la señal I. Un tiempo de símbolo es 4; por tanto medio tiempo de símbolo es 2. De manera consecutiva en la señal Q se elimina la etiqueta existente con el bloque *Tag Gate* para evitar que se duplique al momento de unir nuevamente con la parte real de la señal.

Finalmente, se vuelve a conformar el símbolo complejo que ahora ya es símbolo O-QPSK; para esto se utiliza el bloque *Float To Complex*. La señal O-QPSK a transmitirse se muestra en la Figura 2.9

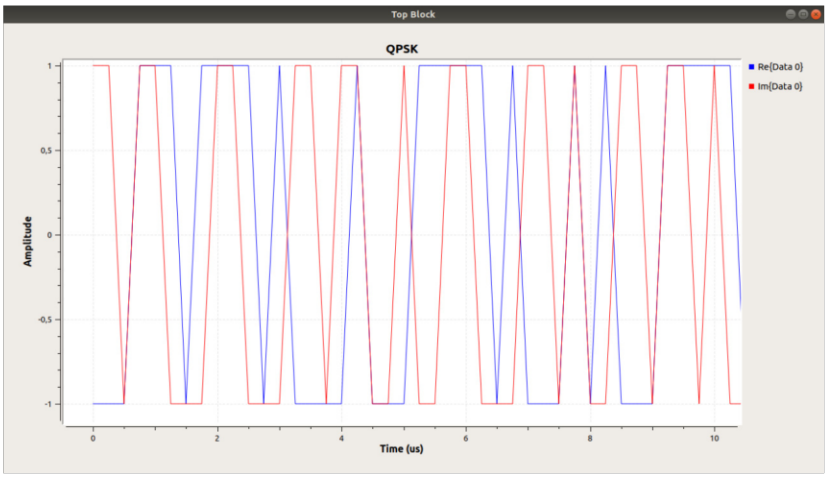

Figura 2.9: Señal O-QPSK

2.2.2. TRANSMISOR BANDA 915 MHz

En la banda de 915 MHz se presentan algunos cambios con respecto a la banda de 2450 MHz, a continuación se realizará la descripción de los bloques que poseen variaciones respecto a la banda anterior. La Figura 2.10 muestra los bloques de procesamiento para el sistema transmisor.

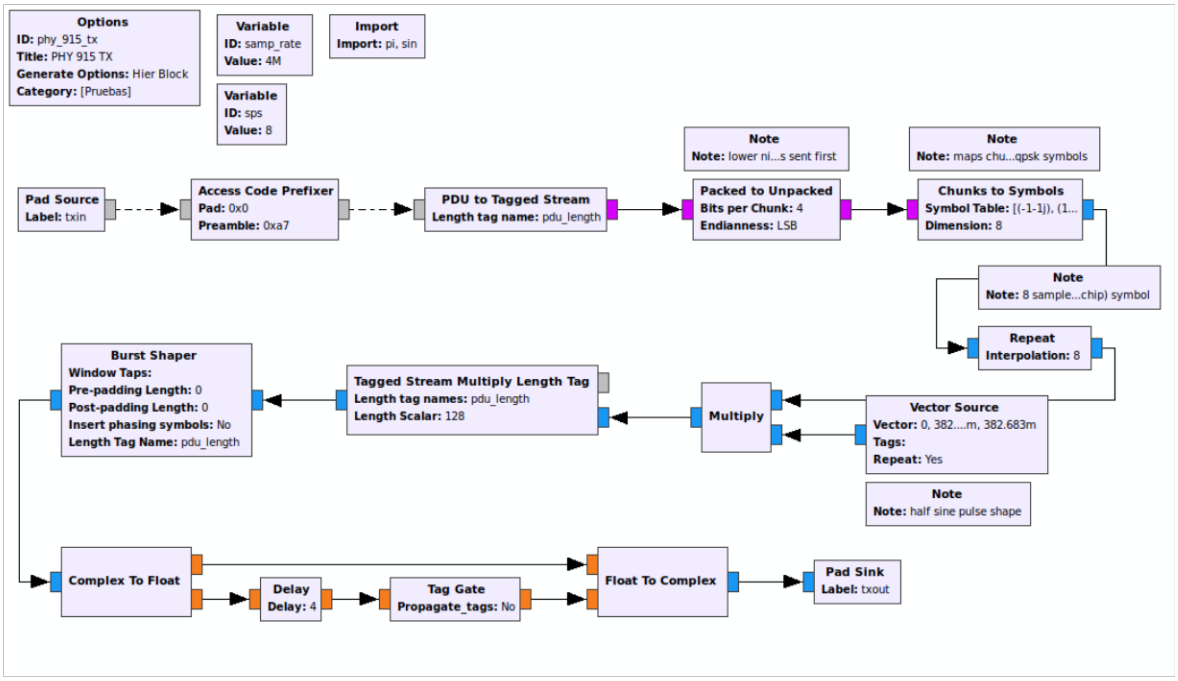

Figura 2.10: Implementación del transmisor en la banda de 915 MHz

2.2.2.1. Mapeo de bits y modulación QPSK

En la banda de 915 MHz el proceso de generación de encabezado y formación de chunks se conserva tal cual se especifica en la banda 2450 MHz . El bloque con el que inician los cambios para la banda 915 MHz es el bloque *Chunks to Symbols* donde comienza la modulación. Una vez obtenido los paquetes de cuatro bits (chunks) en el bloque anterior; estos son mapeados con dieciséis chips por cada símbolo de datos. Los chips están definidos en el estándar IEEE 802.15.4. El parámetro *Dimension* está definido en 8 ya que por cada símbolo de datos se van a mapear 8 muestras complejas. La Tabla 2.4 presenta las muestras complejas obtenidas por cada dos chips.

Tabla 2.4: Chips como muestras complejas (I+JQ) para la banda 915/868 MHz

Símbolo	Chips	Muestras (I + JQ)
0	0011111000100101	(-1-1j), (1 + 1j), (1 + 1j), (1-1j), (-1-1j), (1-1j), (-1 + 1j), (-1 + 1j)
1	0100111110001001	(-1 + 1j), (-1-1j), (1 + 1j), (1 + 1j), (1-1j), (-1-1j), (1-1j), (-1 + 1j)
2	0101001111100010	(-1 + 1j), (-1 + 1j), (-1-1j), (1 + 1j), (1 + 1j), (1-1j), (-1-1j), (1-1j)
3	1001010011111000	(1-1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1 + 1j), (1 + 1j), (1-1j), (-1-1j)
4	0010010100111110	(-1-1j), (1-1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1 + 1j), (1 + 1j), (1-1j)
5	1000100101001111	(1-1j), (-1-1j), (1-1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1 + 1j), (1 + 1j)
6	1110001001010011	(1 + 1j), (1-1j), (-1-1j), (1-1j), (-1 + 1j), (-1 + 1j), (-1-1j), (1 + 1j)
7	1111100010010100	(1 + 1j), (1 + 1j), (1-1j), (-1-1j), (1-1j), (-1 + 1j), (-1 + 1j), (-1-1j)
8	0110101101110000	(-1 + 1j), (1-1j), (1-1j), (1 + 1j), (-1 + 1j), (1 + 1j), (-1-1j), (-1-1j)
9	0001101011011100	(-1-1j), (-1 + 1j), (1-1j), (1-1j), (1 + 1j), (-1 + 1j), (1 + 1j), (-1-1j)
10	0000011010110111	(-1-1j), (-1-1j), (-1 + 1j), (1-1j), (1-1j), (1 + 1j), (-1 + 1j), (+1+ 1j)
11	1100000110101101	(1 + 1j), (-1-1j), (-1-1j), (-1 + 1j), (1-1j), (1-1j), (1 + 1j), (-1 + 1j)
12	0111000001101011	(-1 + 1j), (1 + 1j), (-1-1j), (-1-1j), (-1 + 1j), (1-1j), (1-1j), (1 + 1j)
13	1101110000011010	(1 + 1j), (-1 + 1j), (1 + 1j), (-1-1j), (-1-1j), (-1 + 1j), (1-1j), (1-1j)
14	1011011100000110	(1-1j), (1 + 1j), (-1 + 1j), (1 + 1j), (-1-1j), (-1-1j), (-1 + 1j), (1-1j)
15	1010110111000001	(1-1j), (1-1j), (1 + 1j), (-1 + 1j), (1 + 1j), (-1-1j), (-1-1j), (-1 + 1j)

2.2.2.2. Repetición de número de muestras por símbolo

En la banda 915 MHz el bloque *repeat* será modificado, ya que el número de chips por cada símbolo ahora es 16. La Ecuación 2.1 antes descrita se utilizará para este fin. A continuación se realizan los cálculos respectivos.

Datos:

-Velocidad del estándar: 250 Kbps

-Bits por chunk: 4

-Chips por cada símbolo: 16

Nota: por cada 2 chips se tiene un símbolo QPSK

Cálculo de velocidad de símbolo

Velocidad de chip $= \frac{250kbps}{4bits} \times 16\ chips$

Velocidad de chip $1\frac{Mchips}{seg}$

Por tanto;

Velocidad de símbolo $1\frac{Mchips}{seg} \times \frac{1\ simbolo\ QPSK}{2\ chips}$

Velocidad de símbolo $= \frac{1}{2}\frac{Msimb}{seg}$

Finalmente

$$Muestra_{simb} = \frac{4\frac{Msimb}{seg}}{\frac{1}{2}\frac{Msimb}{seg}}$$

$$Muestra_{simb} = 8$$

2.2.2.3. Generación de muestras de medio pulso seno

Otro de los cambios presentes en la banda en cuestión es el bloque *Vector Source*, el cual genera un flujo de muestras de acuerdo al vector de entrada. Para la banda 915 MHz se generan 8 muestras de medio pulso seno:

$$0, \sin(\frac{\pi}{8}), \sin(\frac{\pi}{4}), \sin(\frac{3\pi}{8}), \sin(\frac{5\pi}{8}), \sin(\frac{3\pi}{4}), \sin(\frac{7\pi}{8})$$

Estas muestras se repetirán de manera indefinida según se indica en el parámetro *repeat*. La Figura 2.11 muestra los parámetros principales del bloque a tomar en cuenta.

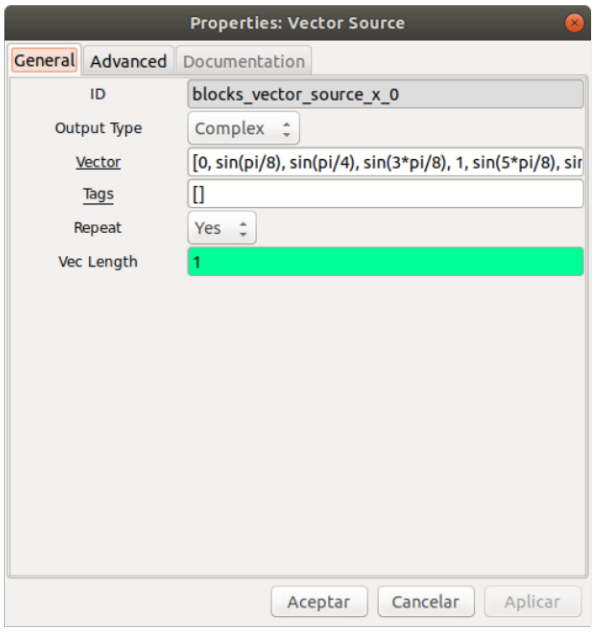

Figura 2.11: Propiedades del bloque ”Vector Source”

2.2.2.4. Corrección de etiqueta y formación de símbolo O-QPSK

El bloque *Tagged Stream Multiply Length Tag* conserva el escalar 128, esto a pesar del aumento del número de muestras de medio pulso seno. La banda 915 MHz disminuye a la

mitad el número de chips, compensando así el valor de la longitud de la etiqueta; el cual es el resultado de multiplicar, el número de muestras complejas por símbolo de datos (8) por el factor de repetición de muestras (8) y por dos (8x8x2 = 128).

Para finalizar la modulación O-QPSK, el bloque *Delay* genera un retardo de medio tiempo de símbolo de la señal Q respecto a la señal I; para la banda en cuestión el tiempo de símbolo es 8, por tanto medio tiempo de símbolo es 4.

2.2.3. TRANSMISOR BANDA 868 MHz

En la banda 868 MHz se presentan cambios en el número de muestras y la implementación de un filtro coseno levantado. El diagrama de procesamiento de señal del sistema transmisor se presenta en la Figura 2.12.

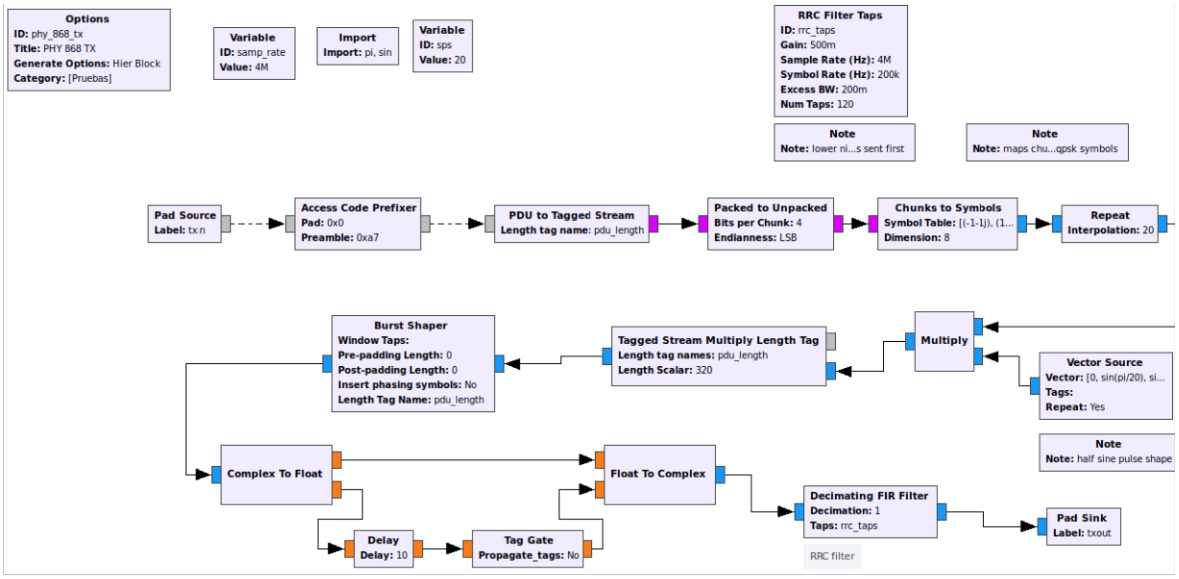

Figura 2.12: Implementación del transmisor para la banda 868 MHz

2.2.3.1. Repetición de muestras por cada símbolo de datos

En la banda 868 MHz se conservan varios parámetros presentes en los bloques de procesamiento de las bandas ya analizadas. Por ejemplo; el bloque *Chunks to Symbols* no presenta cambios con respecto a la banda de 915 MHz ya que utilizan los mismos chips por cada símbolo de datos descritos en la Tabla 2.4 y definidos en el estándar oficial.

Para el cálculo del número de muestras a generarse se debe tomar en cuenta el cambio velocidad de transmisión de datos respecto a la banda anterior. Al igual que en la banda 915 MHz la Ecuación 2.1 facilita el cálculo de muestras a repetir.

Datos:

-Velocidad del estándar: 100 Kbps

-Bits por chunk: 4

-Chips por cada símbolo: 16

Nota: por cada 2 chips se tiene un símbolo QPSK

Cálculo de velocidad de símbolo

Velocidad de chip $= \frac{100kbps}{4bits} \times 16\ chips$

Velocidad de chip $\frac{2}{5}\frac{Mchips}{seg}$

Por tanto;

Velocidad de símbolo $\frac{2}{5}\frac{Mchips}{seg} \times \frac{1\ simbolo\ QPSK}{2\ chips}$

Velocidad de símbolo $= \frac{1}{5}\frac{Msimb}{seg}$

Finalmente

$Muestra_{simb} = \frac{4\frac{Msimb}{seg}}{\frac{1}{5}\frac{Msimb}{seg}}$

$Muestra_{simb} = 20$

2.2.3.2. Generación de muestras de medio pulso seno

De manera paralela a las muestras generadas por símbolo de datos, el bloque *Vector Source* comienza a generar las muestras de medio pulso seno. El bloque *Vector Source* presenta cambios respecto al número de muestras. Para esta banda se generan 20 muestras de medio pulso seno:

$$0, \sin(\frac{\pi}{20}), \sin(\frac{\pi}{10}), \sin(\frac{3\pi}{20}), \sin(\frac{\pi}{5}), \sin(\frac{\pi}{4}), \sin(\frac{3\pi}{10}),$$
$$\sin(\frac{7\pi}{20}), \sin(\frac{2\pi}{5}), \sin(\frac{9\pi}{20}), 1, \sin(\frac{11\pi}{20}), \sin(\frac{3\pi}{5}), \sin(\frac{13\pi}{20}),$$
$$\sin(\frac{7\pi}{10}), \sin(\frac{3\pi}{4}), \sin(\frac{4\pi}{5}), \sin(\frac{17\pi}{20}), \sin(\frac{9\pi}{10}), \sin(\frac{19\pi}{20})$$

Debido al aumento de muestras introducidas por los bloques anteriores, el bloque *Tagged Stream Multiply Length Tag* presenta modificaciones. El parámetro *longitud escalar* para esta banda es 320, ya que se multiplica el número de muestras complejas por símbolo de datos por el factor de repetición y por dos (8x20x2). Seguidamente, se entra al proceso de retardo de la señal I respecto a la señal Q, mediante la introducción del bloque *Delay*. Para la banda 868MHz el valor del retardo es igual a 10; ya que esto representa un retardo de medio tiempo de símbolo de una señal respecto a la otra. El tiempo de símbolo es 20.

2.2.3.3. Filtro coseno levantado

Finalmente, la Sección 1.4.4.2 menciona que para la banda 868 MHz se hace necesaria la inserción de un filtro coseno levantado, antes de la transmisión para reducir la interferencia entre símbolos. GNU Radio no cuenta con este filtro, por lo que se remplaza con dos filtros raíz de coseno levantado; la combinación de estos dos filtros; uno en transmisión y el otro en recepción desempeñan la función de un filtro coseno levantado. Este tipo de proceder en los sistemas de comunicación se conoce como filtrado adaptado [19]. La Figura 2.13 muestra los parámetros de filtrado RRC (Root- Raised-Cosine) y el bloque que realiza el filtrado.

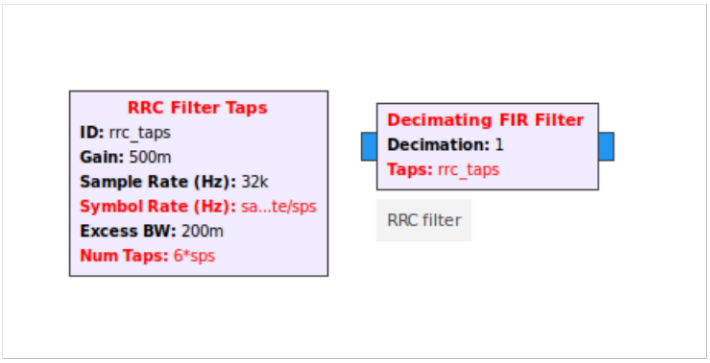

Figura 2.13: Parámetros de filtrado RRC y bloque *Decimating FIR Filter*

2.3. IMPLEMENTACIÓN DEL SISTEMA RECEPTOR

Las tres bandas en estudio comparten procesos similares en la recepción de los símbolos de datos. En el receptor se implementarán bloques que han sido creados específicamente para los procesos de: selección de muestras, demodulación de símbolos QPSK y desmapeo de chips; en este último, se utilizaron los mismos chips que en transmisión. En la banda 868 MHz el receptor introduce un proceso adicional el cual es el filtrado de la señal con un filtro raíz de coseno levantado. El esquema de procesamiento de señal en el receptor se muestra en la Figura 2.14.

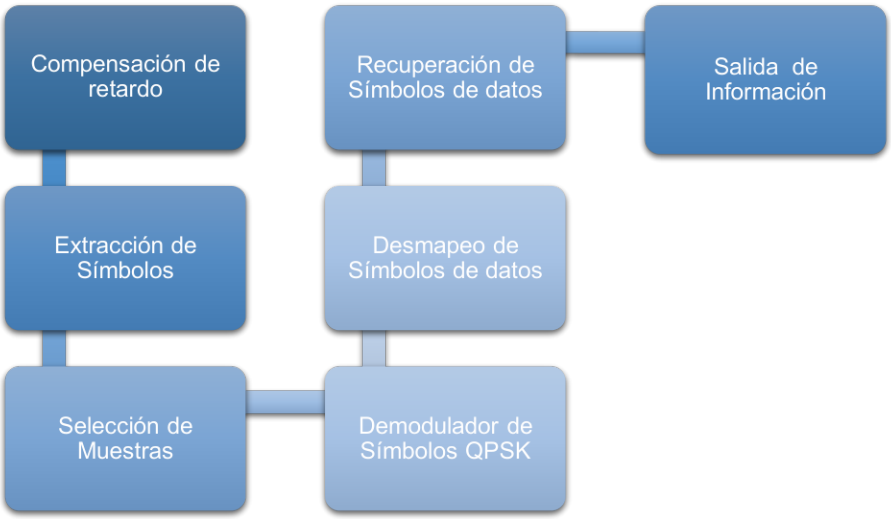

Figura 2.14: Proceso a seguir para implementar el sistema transmisor

2.3.1. RECEPTOR BANDA 2450 MHz

El receptor se implementa tomando en cuenta los mismos chips que se implementaron en transmisión. Esto con el fin de recuperar los símbolos de datos. En la Figura 2.15 se observan los bloques de procesamiento implementados.

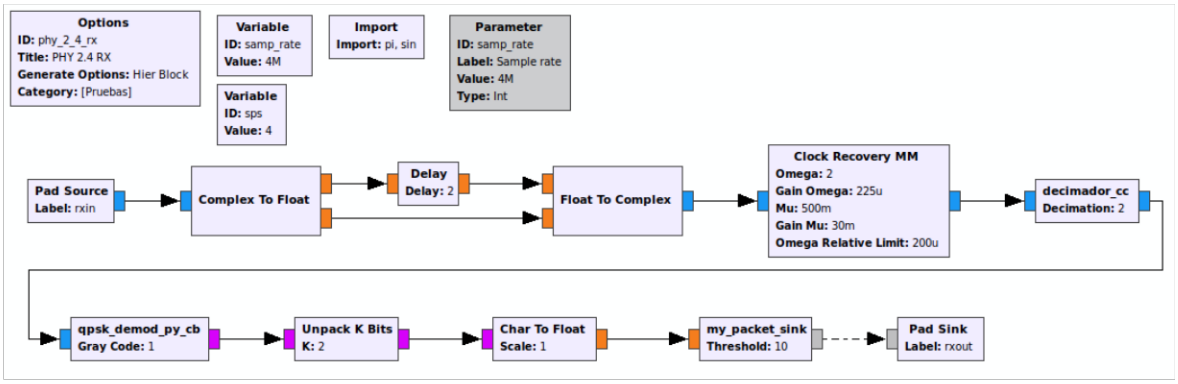

Figura 2.15: Implementación del receptor para la banda de 2450 MHz

2.3.1.1. Compensar el retardo

Antes de iniciar la compensación del retardo, el receptor recibe la señal O-QPSK afectada por el ruido del canal. Esto se muestra en la Figura 2.16.

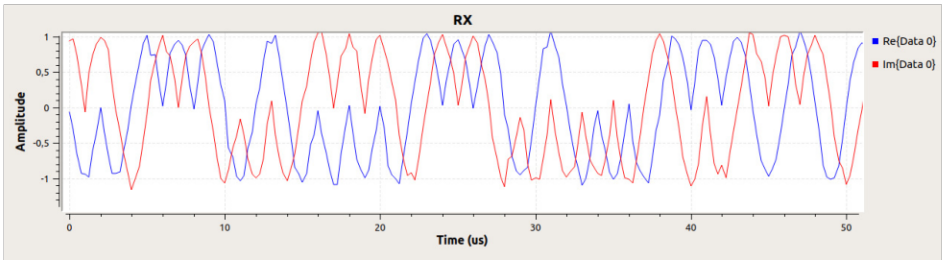

Figura 2.16: Señal O-QPSK con ruido

Una vez separada la parte imaginaria y la real por el bloque *Complex to Float*, el retardo en el transmisor es compensado al introducir un retardo en la parte real del símbolo O-QPSK de medio tiempo de símbolo. Una vez compensado el retardo se obtiene la señal de la Figura 2.17.

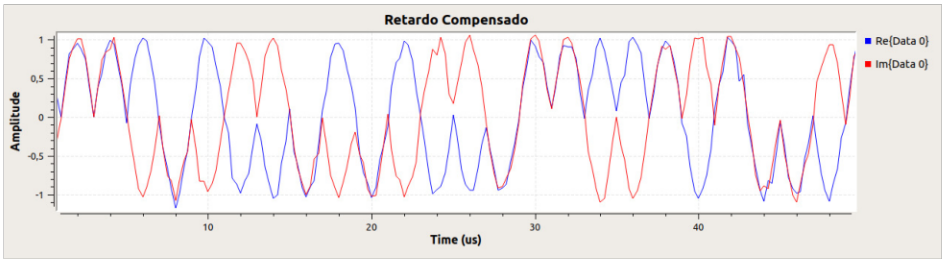

Figura 2.17: Señal O-QPSK compensada el retardo

2.3.1.2. Extracción de símbolos de datos

A la salida del bloque *Float To Complex* se forma nuevamente el símbolo complejo para ser ingresado en el bloque *Clock Recovery MM*; este bloque permite la extracción de los símbolos que se transmitieron, mediante su función como recuperador de reloj [28]. Para este caso, debido al retardo que se introduce se procede a la estimación del símbolo utilizando solo dos muestras. El parámetro principal a modificar en este bloque es el valor de *Omega* . Para la banda de 2450MHz *Omega* es 2. Esto de cuerdo al calculo realizado en la Sección 2.2.1.6 y se puede ver en la Figura 2.18.

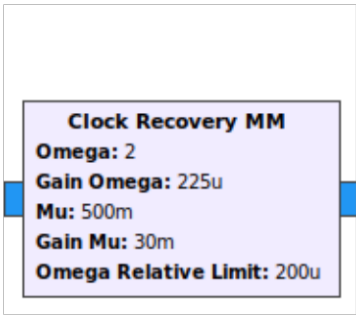

Figura 2.18: Bloque para la extracción de símbolos

2.3.1.3. Selección de muestras

El bloque *decimador_cc* recibe a la entrada un número estimado de muestras por el bloque *Clock Recovery MM* (2 muestras por símbolo).El bloque *decimador_cc* devuelve a la salida únicamente una muestra por símbolo; las muestras seleccionadas son las pares que se recibieron a la entrada, debido a que la primera muestra (impar) es afectada por el retardo. Esto se ve en la Figura 2.19.

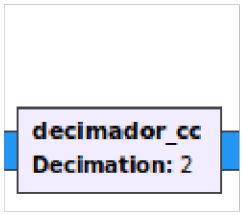

Figura 2.19: Bloque para seleccionar muestras

El bloque *decimador_cc* selecciona las muestras [1, 3, 5, 7, 9,…], de las muestras [0, 1, 2 3, 4, 5, 6, 7,8,…], por tanto, tomando en cuenta a la primera muestra con 0, la muestra par viene a ser la muestra 1, y así sucesivamente. El Segmento de código 2.1 muestra el lazo utilizado para seleccionar las muestras pares y almacenarse en el vector “Out [i]” de salida.

Segmento de código 2.1: Selección de muestras

```
1       // Do <+signal processing+>
2       for(int i = 0; i < noutput_items; i++)
3       {
4         out[i] = in[2*i+1];
5       }
6       // Tell runtime system how many output items we produced.
7       return noutput_items;
```

Una vez obtenida una única muestra por símbolo de datos; se obtiene la señal QPSK en la Figura 2.20 y la constelación en la Figura 2.21.

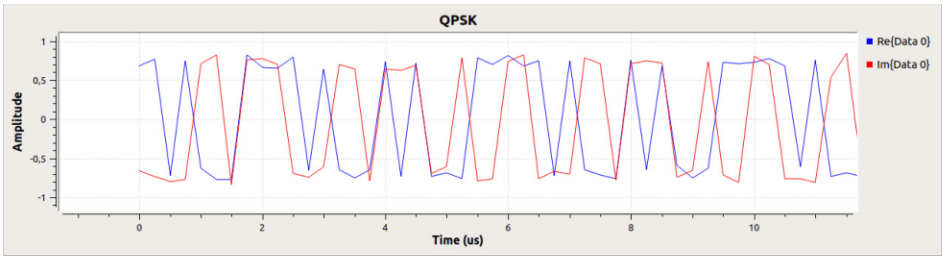

Figura 2.20: Señal QPSK en Recepción

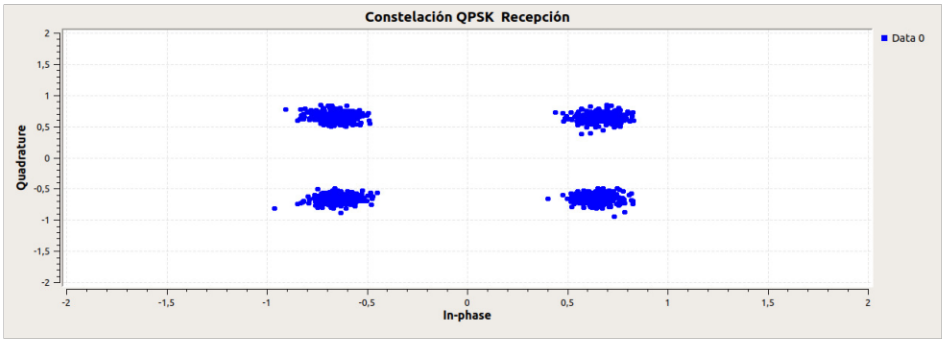

Figura 2.21: Constelación QPSK en recepción

2.3.1.4. Demodulación de símbolos QPSK

El bloque *qpsk_dem* recibe a la entrada las muestras pares complejas, para internamente demodularlas y obtener a la salida bytes. Los dos bits menos significativos de cada byte corresponden a los dos chips transmitidos por cada muestra compleja. El bloque se presenta en la Figura 2.22.

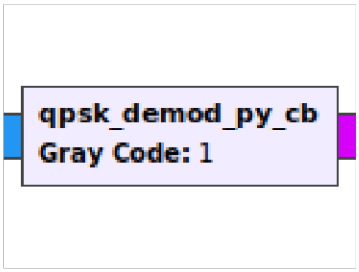

Figura 2.22: Bloque demodulador QPSK

El Segmento de código 2.2 muestra el lazo principal junto con la función (get minimum distances) la cual realiza la trasformación de cada símbolo complejo a chips (2 bits por chips).

Segmento de código 2.2: Conversión de símbolo QPSK a chip

```
1        // Do <+signal processing+>
2        for(int i = 0; i < noutput_items; i++)
3        {
4          out[i] = get_minimum_distances(in[i]);
5        }
6        consume_each (noutput_items);
7
8        // Tell runtime system how many output items we produced.
9        return noutput_items;
```

Los estados de símbolo en fase I y cuadratura Q en cada cuadrante están representados por dos bits. Según el segmento de código, el demodulador recupera los bits que representan cada símbolo, al comparar cada estado de símbolo; si el estado es mayor a cero el bit es 1L, si el estado es menor a cero el bit es 0L. El Segmento de código 2.3 muestra este proceso alineado a la codificación Gray.

Segmento de código 2.3: Conversion de simbolo QPSK a chip

```
1       qpsk_demod_cpp_cb_impl::get_minimum_distances(const gr_complex &sample)
2     {
3       if (d_gray_code) {
4         unsigned char bit0 = 0;
5         unsigned char bit1 = 0;
6         // The two left quadrants (quadrature component > 0) have this bit ...
              set to 1
7         if (sample.real() ≥ 0) {
8           bit0 = 0x01 << 1;
9         }
10        // The two lower quadrants (in-phase component > 0) have this bit ...
              set to 1
11        if (sample.imag() ≥ 0) {
12          bit1 = 0x01;
13        }
14        return bit0 | bit1;
15      }
```

El bloque *Unpack k bits* permite la extracción de los bits menos significativos de cada byte a la entrada. El parámetro *K=2* representa el número de bits a extraerse. Una vez extraídos los bits de cada byte estos son colocados de manera consecutiva hasta completar un byte. Este proceso se realiza de manera consecutiva hasta tener a la salida bytes completos (con 8 bits por byte en su interior).

Seguidamente el Bloque *Chart to float* permite convertir el byte de entrada a formato decimal. El factor de escala aplicado es el predeterminado Scale: 1.

2.3.1.5. Recuperación de símbolos de datos

El bloque *my_packet_sink* toma los decimales a la entrada y va formando una secuencia de 32 chips necesaria para formar un símbolo de datos. Los 32 chips obtenidos son comparados con los 32 chips establecidos por el estándar; dicha comparación permite obtener los 4 bits

que se empaquetaron por cada chunk en el transmisor. Entre los parámetros principales del bloque *my_packet_sink* está Threshold = 10 [5]; el cual representa el número de bits errados que se aceptarán. Esto se puede ver en la Figura 2.23.

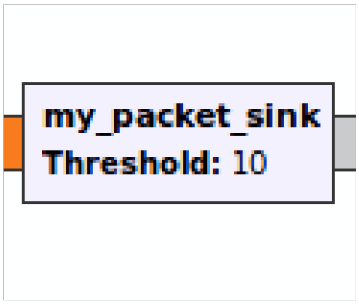

Figura 2.23: Bloque recuperador de símbolos de datos

El Segmento de código 2.4 presentado a continuación, muestra los chips en formato decimal para los dieciséis símbolos de datos definidos en el estándar IEE 802.15.4. Cada uno de los 16 símbolos tiene asignada una secuencia pseudo aleatoria de 32 chips. En el presente proyecto el receptor se implementó con los mismos chips que se utilizaron en el transmisor.

Segmento de código 2.4: Chips para la banda 2450 MHz

```
1       static const unsigned int CHIP_MAPPING[] = {
2                       3653456430,
3                       3986437410,
4                       786023250,
5                       585997365,
6                       1378802115,
7                       891481500,
8                       3276943065,
9                       2620728045,
10                      2358642555,
11                      3100205175,
12                      2072811015,
13                      2008598880,
14                      125537430,
15                      1618458825,
16                      2517072780,
17                      3378542520};
```

El bloque *my_packet_sink* de manera inicial busca el delimitador de inicio de trama, una vez que lo tiene busca la cabecera del paquete donde se encuentra el tamaño del payload. Una

vez obtenido el tamaño del payload empieza a construir la carga del paquete. El Segmento de código 2.5 muestra la comparación entre los chips obtenidos en el transmisor y los definidos en el estándar. La función *count_bits32* permite dicha comparación [17].

Segmento de código 2.5: Utilización de la función count_bits32

```
unsigned char
    my_packet_sink_impl::decode_chips(unsigned int chips){
    int i;
    int best_match = 0xFF;
    int min_threshold = 33; // Matching to 32 chips, could never have a ...
        error of 33 chips

    for(i=0; i<16; i++) {
        // FIXME: we can store the last chip
        // ignore the first and last chip since it depends on the last chip.
        threshold = count_bits32(chips ^ CHIP_MAPPING[i]);

        if (threshold < min_threshold) {
            best_match = i;
            min_threshold = threshold;
        }
    }
```

Finalmente,los chips recuperados por cada símbolo de datos en entorno de simulación se muestra en la Figura 2.24.

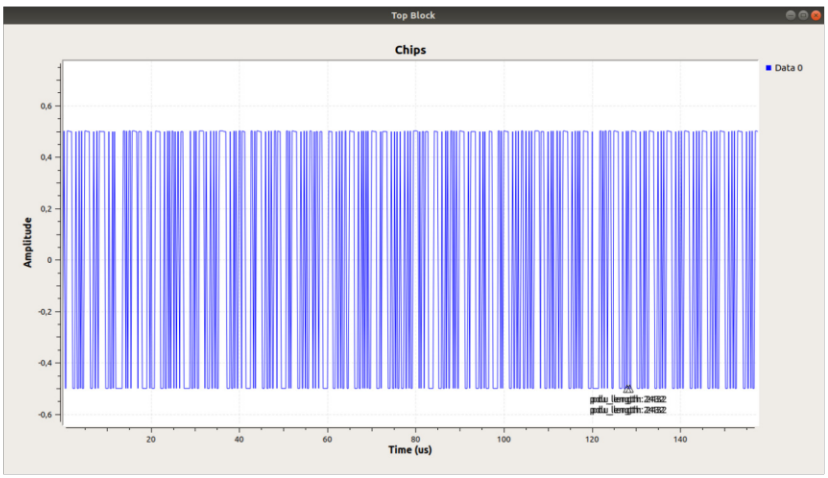

Figura 2.24: Chips Recuperados

2.3.2. RECEPTOR BANDA 915 MHz

A continuación, se muestra el diagrama de procesamiento de la señal en el receptor, donde se pueden observar modificaciones específicas en ciertos bloques de programación que se analizarán a continuación. Esto se ve en la Figura 2.25.

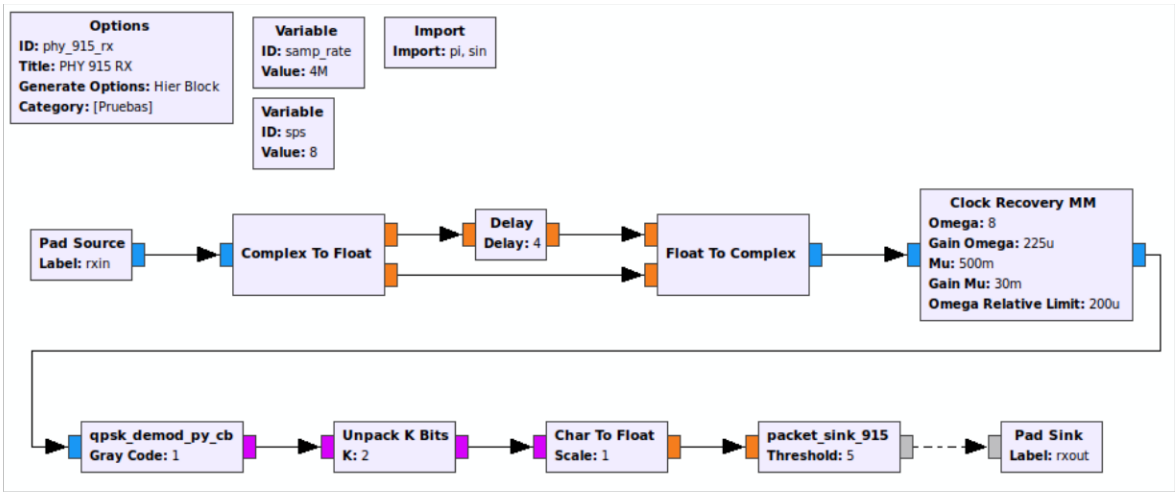

Figura 2.25: Implementación del receptor para la banda 915 MHz

2.3.2.1. Compensar el retardo

Una vez receptados los símbolos de datos, estos son separados en una parte real y otra imaginaria por el bloque *Complex To Float*. Seguidamente, para compensar el retardo introducido en el transmisor; el receptor retarda a 4 la parte real del símbolo de datos.

2.3.2.2. Extracción de muestras y demodulación

Otro de los cambios presentes en la banda 915 MHz está en el bloque *Clock Recovery MM*. Debido a la escasa información disponible sobre el funcionamiento del bloque se procedió a la modificación de los parámetros mediante prueba y error; en especial con la modificación del parámetro *Omega*; que para este caso, el número de muestras a recuperar por señal será ocho.

Una vez recuperadas las muestras por señal; estas son demoduladas utilizando un demodulador QPSK; posteriormente se realiza la extracción de los dos bits menos significativos y se procede al cambio de formato de bits a formato decimal. En comparación con las bandas 2450 MHz no se presentan cambios en el código fuente y parámetros de los bloques de programación.

2.3.2.3. Recuperación de símbolos de datos

Finalmente, en la banda 915 MHz se presenta el bloque *packe_sink_915*, que para este caso se realizaron modificaciones respecto a la banda de 2450 MHz, empezando por el número de chips asignados para esta banda en el código fuente. En el Segmento de código 2.6 se introducen los 16 chips en formato decimal por cada símbolo de datos.

Segmento de código 2.6: Chips para la banda 915 MHz en formato decimal

```
1  static const unsigned int CHIP_MAPPING[] = {
2                      15909,
3                      20361,
4                      21474,
5                      38136,
6                      9534,
7                      35151,
8                      57939,
9                      63636,
10                     27504,
11                     6876,
12                     1719,
13                     49581,
14                     28779,
15                     56346,
16                     46854,
17                     44481};
```

2.3.2.4. Demodulación de símbolos QPSK

Además, se introduce la función *count_bits16* la cual fue modificada para poder trabajar con los 16 chips por cada símbolo de datos. El Segmento de código 2.7 muestra el proceso.

Segmento de código 2.7: Utilización de la función count_bits16

```
1  unsigned char
2      packet_sink_915_impl::decode_chips(unsigned int chips){
3      int i;
4      int best_match = 0xFF;
5      int min_threshold = 17; // Matching to 32 chips, could never have a ...
           error of 33 chips
```

```
6
7       for(i=0; i<16; i++) {
8           // FIXME: we can store the last chip
9           // ignore the first and last chip since it depends on the last chip.
10          threshold = count_bits16(chips ^ CHIP_MAPPING[i]);
11
12          if (threshold < min_threshold) {
13              best_match = i;
14              min_threshold = threshold;
15          }
16      }
17      }
```

2.3.3. RECEPTOR BANDA 868 MHz

En esta sección se presentan los bloques de procesamiento para el receptor de la banda 868 MHz. Esto se observa en la Figura 2.27

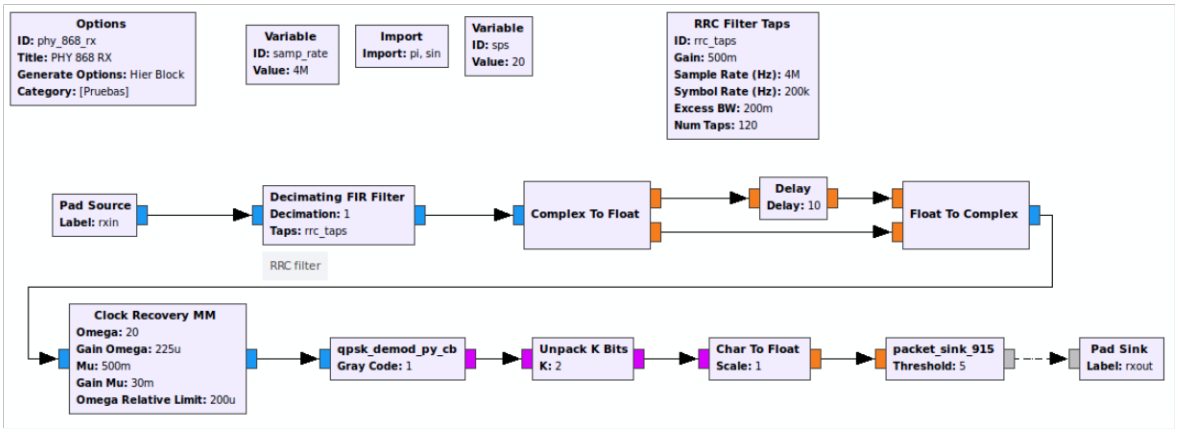

Figura 2.26: Implementación del receptor para la banda 868 MHz

2.3.3.1. Filtro coseno levantado.

Una vez receptada la señal, esta es sometida al filtro raíz de coseno levantado para finalizar el proceso de filtrado. Los bloques *Decimating FIR Filter* tanto en el receptor así como en el transmisor actúan como un filtrado adaptado.

2.3.3.2. Compensación del retardo y recuperación de muestras.

De manera seguida los símbolos complejos de datos receptados son separados para compensar el retardo introducido en el transmisor. El retardo es de la mitad de tiempo de símbolo; es decir 10. El bloque *Clock Recovery MM* también presenta cambios en el número

de muestras extraídas por símbolo; para el presente caso el parámetro *Omega* será de 20 muestras por símbolo.

2.3.3.3. Demodulación

El proceso de demodulación permanece inalterable con el bloque *qpsk_demod_py_cb*. El bloque *packet sink_915* no presenta cambios en el código fuente ya que la banda de 868 MHz utiliza los mismos chips por símbolo. El parámetro modificado en el bloque de procesamiento mencionado es el *Threshold* el cual es el número de bits errados que se aceptarán, que para este caso es 5.

2.4. ACOPLAMIENTO DE ETAPAS Y CONFIGURACIÓN DE USRP 2920

Una vez implementados los transmisores y receptores en las secciones anteriores, se procede a crear los bloques jerárquicos y a configurar los bloques *UHD:USRP Source* para facilitar el acoplamiento con otras etapas.

2.4.1. ACOPLAMIENTO DEL TRANSMISOR Y RECEPTOR

Con las etapas previamente concebidas como: la generación del mensaje, el transmisor , el canal de propagación y el receptor;estas son acopladas para realizar las pruebas de funcionamiento en entorno de simulación. La Figura 2.27 muestra el sistema de comunicación para las tres bandas en estudio.

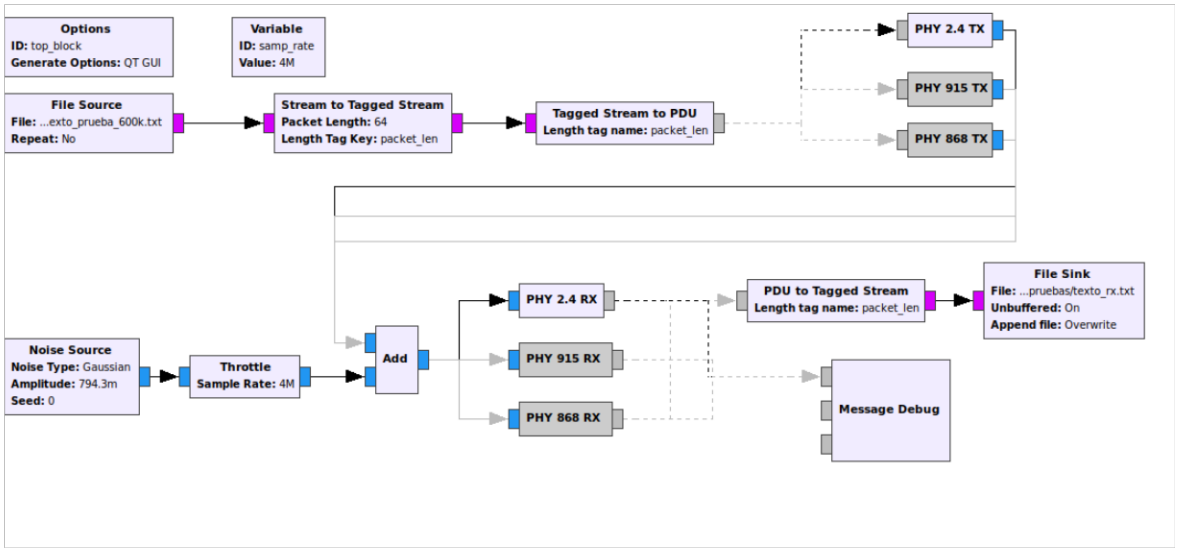

Figura 2.27: Implementación del transceptor para las bandas de 2450, 915 y 868 MHz

2.4.2. CONFIGURACIÓN USRP 2920

La conexión entre los equipos USRP y los bloques de procesamiento se realiza mediante los bloques *UHD:USRP Source* y *UHD:USRP Sink*. Previo a la configuración del mismo, es necesaria descargar e instalar las imágenes FPGA para controlar el dispositivo USRP 2920. Estas imágenes están disponibles en el github de Ettus Research [20].

La Figura 2.28 muestra las propiedades del bloque *UHD:USRP Source* , de donde se extrae los parámetros mas relevantes los cuales son [14].

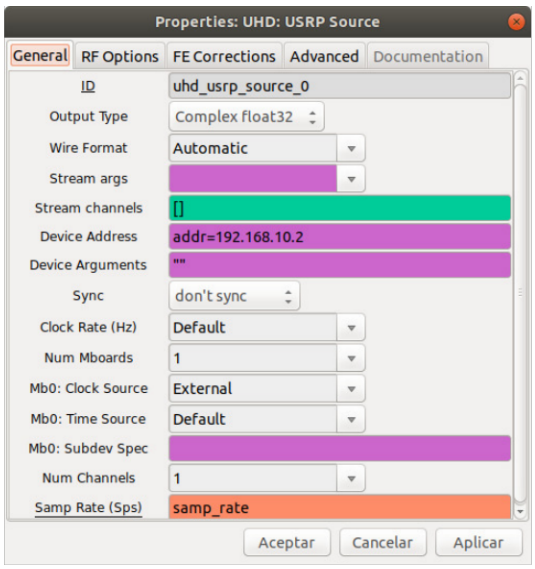

Figura 2.28: Propiedades del bloque *UHD:USRP Source*

- **Device Address:** permite identificar la dirección IP del dispositivo USRP. La dirección por defecto para los USRP 2920 es 198.192.10.2; la cual debe estar en red con la dirección IP del ordenador.
- **Samp Rate (Sps):** especifica la frecuencia de muestreo a la que va a trabajar el dispositivo.Para la banda de 915 MHz se estableció una frecuencia de 2 MHz.
- **Center Freq:**define la frecuencia central de operación. Para la banda de 915 MHz será 906 MHz.
- **Gain value:** permite asignar el valor de ganancia a la señal en transmisión y recepción. El USRP 2920 permite variar la ganancia desde 0 dB a 31.5 dB. El formato puede ser absoluto(dB) o normalizado.
- **Antena:** ayuda a identificar si la antena es transmisora Tx/Rx o receptora Rx2.

3. RESULTADOS,CONCLUSIONES Y RECOMENDACIONES

3.1. RESULTADOS

En esta sección se presentan los resultados obtenidos en las pruebas de funcionamiento tanto en simulación así como con los equipos USRP 2920. Cabe recalcar que en el trascurso de comprobar el funcionamiento de los sistemas transmisores y receptores se realizaron cambios en los parámetros que controlan el procesamiento de las señales; los cuales serán tratados en esta sección. De primera mano se realizarán pruebas de funcionamiento en un entorno de simulación, mediante la utilización de un canal con una fuente de ruido tipo Gaussian. Posterior a eso se realizará la configuración de los equipos USRP para realizar las pruebas en un entorno indoor. Con las pruebas de funcionamiento se espera obtener parámetros como la tasa de bit errados BER , nivel de calidad de enlace LQI, además de observar la llegada de paquetes en recepción.

3.1.1. CALCULO DEL BER (BIT ERROR RATE)

Para el calculo del BER se realiza la comparación en bits de dos archivos de texto plano .txt (archivo de texto original y archivo de texto receptado). La comparación se da sin tomar en cuenta los espacios entre palabras. El archivo a trasmitir tiene un tamaño de 600 kbytes. La herramienta utilizada para el cálculo en cuestión será Matlab; la cual provee la función *tex2bin* para convertir texto en bits y la función *biterr* par calcular el BER [21]. El Segmento de código 3.8 muestra a detalle el proceso a seguir para obtener el BER.

Segmento de código 3.8: Cálculo BER en Matlab

```
1       clearvars
2      x=fopen('texto.txt');%abrir archivo a enviar
3      x1=fscanf(x,'%c');%lee archivo sin espacios
4      [bit_tx,binTX]=txt2bin(x1);%texto a binario
5      y=fopen('texto_rx.txt');%abrir archivo receptado
6      y1=fscanf(y,'%c');%leer archivo sin espacios
7      [bit_rx,binRX]=txt2bin(y1);%texto a binario
8      ltx=length(bit_tx); % igualar y comparar longitudes de texto
9      lrx=length(bit_rx);
10     if ltx<lrx
11       bit_rx=bit_rx(1,1:ltx);
12     else
```

```
13          bit_rx(1,lrx+1:ltx)=0;
14        end
15        [nerr,rber]=biterr(bit_tx,bit_rx);%calcula el numero de bits errados y ...
              el BER
16        fclose(x);
17        fclose(y);
```

3.1.2. CÁLCULO DEL LQI (NIVEL DE CALIDAD DE ENLACE)

El parámetro LQI ayuda a tener una noción de la calidad de un paquete recibido, en la sección menciona que los valores mínimos y máximos del parámetro en cuestión, están relacionados con la calidad de las señales detectables por el receptor. En la transmisión el valor de LQI se genera como metadato por cada paquete de longitud, por lo que se implementó el bloque *display_lqi_value* que permitirá extraer los valores en bytes; tal y como se muestra en el Segmento de código 3.9.

Segmento de código 3.9: Extracción de valores LQI

```
1     def print_LQI(self, msg_pmt):
2     # Extraer el valor de LQI
3     key0 = pmt.intern("lqi")
4         msg_lqi = pmt.car(msg_pmt)
5     v_lqi = pmt.dict_ref(msg_lqi, key0, pmt.PMT_NIL)
6     if(pmt.eq(v_lqi, pmt.PMT_NIL)):
7         vLQI = 0
8     else:
9         vLQI = pmt.to_long(v_lqi)
10    msg = [ord(c) for c in str(vLQI)]
11    msg.append(10) # Insertar salto de linea
12    # Crea un PMT vacio
13    send_pmt = pmt.make_u8vector(len(msg), ord(' '))
14    # Copia caracteres a u8vector
15    for i in range(len(msg)):
16        pmt.u8vector_set(send_pmt, i, msg[i])
17    # Envia mensaje
18    self.message_port_pub(pmt.intern('out'), pmt.cons(pmt.PMT_NIL, send_pmt))
```

Posteriormente estos flujos de datos serán etiquetados por el bloque *PDU to Tagged Stream*, para finalmente ser receptados por el bloque *File Sink*, que permitirá escribir estos datos en un archivo binario. Para este caso en particular el archivo es *lqi_rx.txt*.

La Figura 3.1 muestra los bloques que realizan todo el proceso antes mencionado.

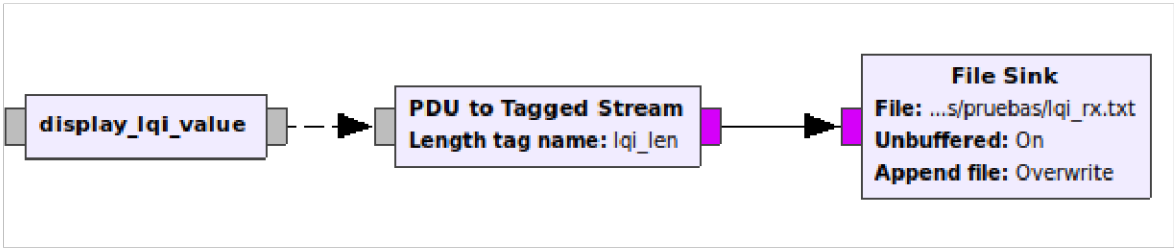

Figura 3.1: Proceso de extracción del parámetro LQI

3.1.3. PRUEBAS DE FUNCIONAMIENTO EN ENTORNO DE SIMULACIÓN

Para las pruebas de funcionamiento se toma en cuenta el uso de un canal tipo gaussiano, como se observa en la Sección 2.4.1 . En este escenario se realiza el cálculo de ciertos valores de amplitud de ruido para valores determinados de SNR. Esto se logra con la utilización de la Ecuación 3.1.

$$SNR = 10\log(\frac{A_s}{A_r})^2 \tag{3.1}$$

Con los valores determinados de ruido; se podrá observar el comportamiento de la transmisión reflejada en el archivo de texto obtenido en el receptor.

Se realizaron 20 pruebas por cada valor de ruido, obteniendo con cada una de ellas un archivo *texto_rx.tx* y posteriormente un valor de BER . Además se obtiene el archivo *lqi_rx.txt* que posteriormente será procesado para obtener un valor promedio de LQI por cada prueba. Los valores obtenidos se presentan en la Tabla 3.1.

Tabla 3.1: Resultados de las pruebas en entorno de simulación

PRUEBAS EN SIMULACIÓN			
SNR	**Ar**	**LQI (Promedio)**	**BER (Promedio)**
0	1.0000	218.9879	0,4455
1	0.8913	222.4310	0.4500
2	0.7943	226.7352	0.4656
3	0.7079	232.8171	0.4714
4	0.6310	237.8280	0.4714
5	0.5623	242.8473	0.4712
6	0.5012	246.5525	0.2582
7	0.4467	248,2402	0,1060
8	0.3981	249,57	6.4978E-05
9	0.3548	251.65	6.4978E-05
10	0.3162	253.6442	6.4978E-05
11	0.2818	254,6174	6.4978E-05
12	0.2512	254.92	6.4978E-05
20	0.1000	255	6.4978E-05

Posteriormente, los valores presentados fueron trasladados a gráficos, para observar el comportamiento del enlace. Se realizó el gráfico BER vs SNR y la variación del parámetro LQI respecto a la relación señal a ruido; los cuales se observan en la Figura 3.2 y la Figura 3.3 respectivamente.

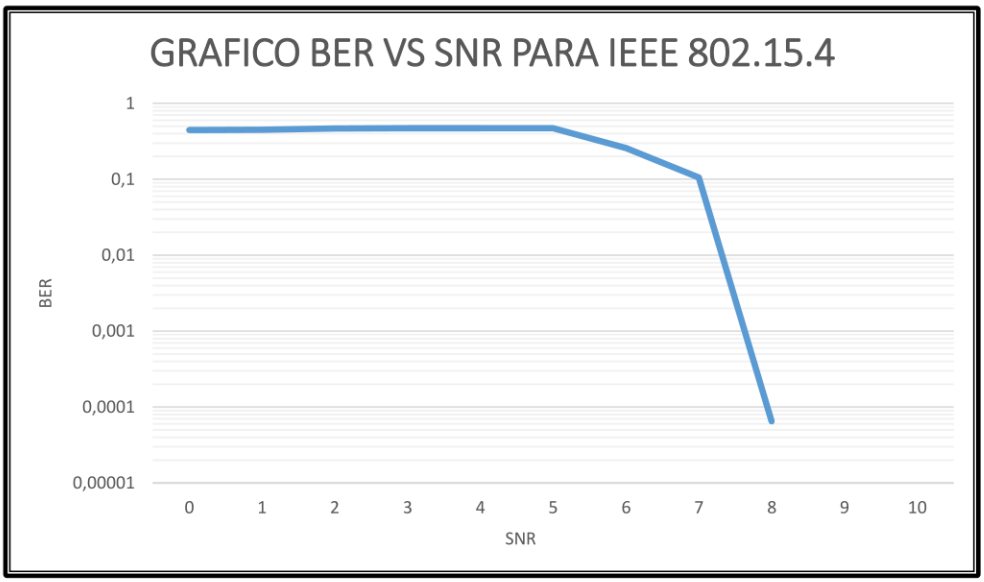

Figura 3.2: Gráfico BER vs SNR en entorno de simulación

En la gráfica se puede observar un punto de inflexión a un SNR de 8 y con Ar de 0.3961 a partir del cual el comportamiento del canal muestra su mayor eficiencia de transmisión, con un BER igual 6.4978E-05 el cual se mantiene a pesar de aumentar el valor de SNR. A partir de estos valores el texto llega al receptor de manera intacta. En este punto cabe recalcar que el tamaño de archivo máximo que se envió, sin antes producir el desbordamiento del búfer en la plataforma, fue de 600.08 Kbytes; evitando así el descarte de paquetes.

Comparando la gráfica con los datos obtenidos, podemos inferir que existe un intervalo en el valor del LQI para el cual los paquetes recibidos mantienen un correspondencia total con los enviados. Estos valores oscilan de 249 a 255; donde éste último es el valor prescrito en el estándar;con valores inferiores a 249 y Ar= 0.3961 los paquetes llegan incompletos y algunos se pierden.

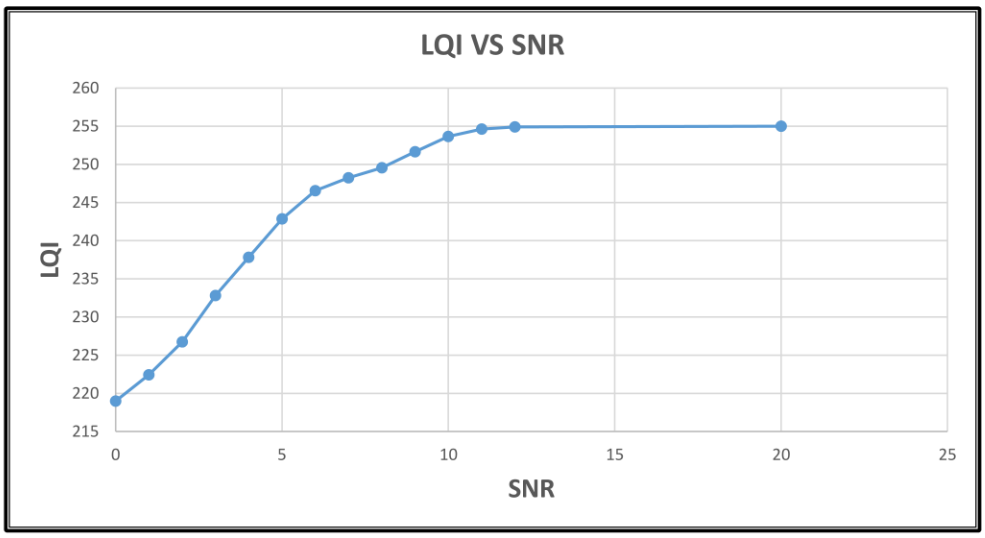

Figura 3.3: Gráfico LQI vs SNR

3.1.4. PROBLEMAS GENERADOS POR OVERFLOW (D/O) UNDERFLOW (U) AL UTILIZAR LOS USRP NI 2920

En las pruebas de funcionamiento con los equipos USRP se encontraron problemas relacionados con el desbordamiento del búfer;observando en el panel de consola la impresión de caracteres U y D. En [22] se menciona que las notas de overflow se producen cuando el ordenador no procesa los datos lo suficientemente rápido en recepción; si el búfer del socket se llena empezará a descartar los paquetes subsiguientes. Posteriormente el software UHD detecta ese desbordamiento como una falta continuidad en la secuencia de paquetes; imprimiendo finalmente el caracter D en la consola. Las notas underflow aparecen cuando el ordenador no produce datos lo suficientemente rápido para ser consumidos por el equipo USRP,imprimiendo el caracter U. Tanto en recepción así como en transmisión los USRP producen y consumen datos a una velocidad constante respectivamente. Para aliviar el procesamiento en los ordenadores se decidió modificar la frecuencia de muestro de 4 a 2 MHz, en la banda de 915/868 MHz.

Los bloques y cálculos modificados en repercusión a lo anterior son :

- Cálculo de número de repetición de muestras por símbolo (Sección 2.2.2.2)
- Generación de nuevas muestras de medio pulso seno (Sección 2.2.2.3)
- Corrección de etiqueta y retardo de muestras (Sección 2.2.2.4)

- Añadir el bloque *decimador_cc* al receptor (Sección 2.3.1.3)

El transmisor y receptor modificados se los puede ver en la Figura3.4.

A pesar de realizar el cambio en la frecuencia de muestreo aún persistía el problema del búfer, en menor grado. Se optó por adquirir computadores de mejores características en cuanto a procesador y memoria RAM. Adicionalmente se modificaron los parámetros de ganancia en transmisión y recepción y se cambió el método de generación del mensaje a transmitir. El mensaje a transmitir se realizará con la ayuda del bloque *Message Strobe* (Figura 3.4). El cual permite la generación de un mensaje de texto en formato PMT (Polymorphic Type) cada cierto tiempo en ms.

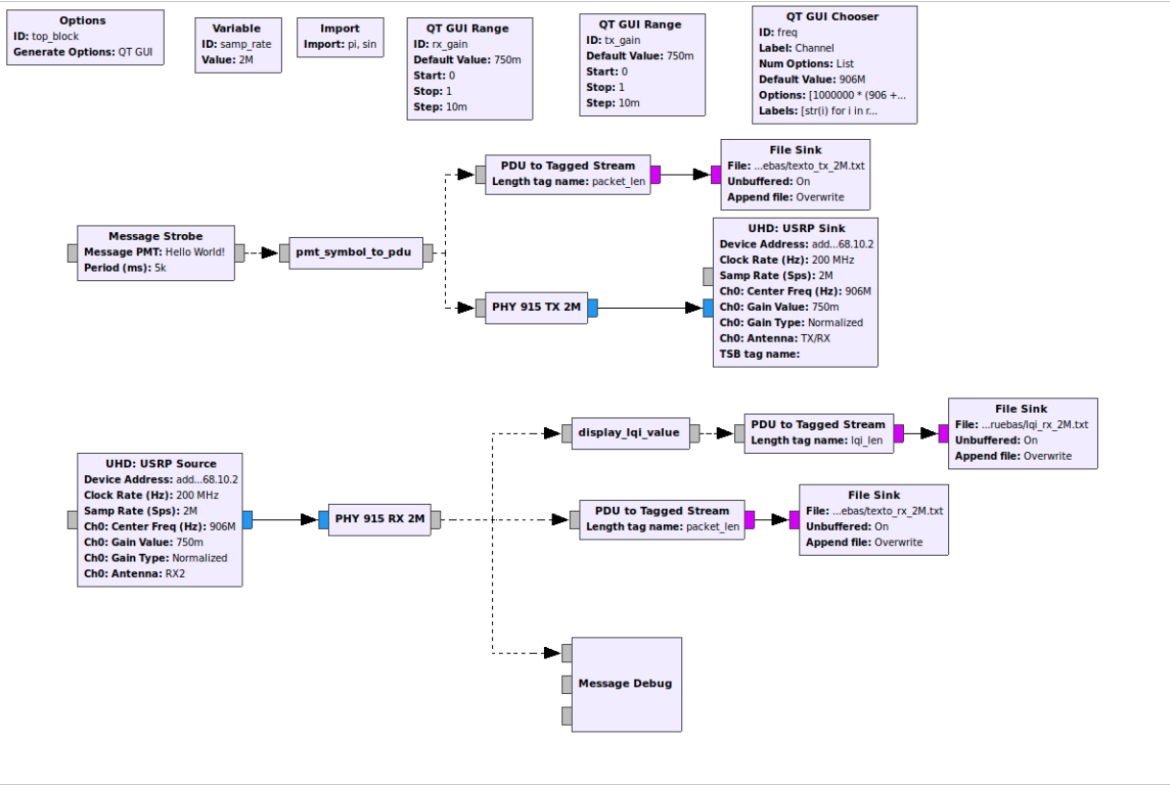

Figura 3.4: Transceptor en la banda de 915 MHz

El mensaje debe ser generado en bits, por tanto se creó el bloque *pmt_symbol_to_pdu* para obtener los mismos. La programación del bloque se muestra en el Segmento de código 3.10.

Segmento de código 3.10: Formato PMT a bytes

```
1    class pmt_symbol_to_pdu(gr.sync_block):
2    """
3    docstring for block pmt_symbol_to_pdu
4    """
```

```
5       def __init__(self):
6           gr.sync_block.__init__(self,
7               name="pmt_symbol_to_pdu",
8               in_sig=[],
9               out_sig=[])
10      self.message_port_register_in(pmt.intern("in"))
11      self.message_port_register_out(pmt.intern("out"))
12      self.set_msg_handler(pmt.intern("in"), self.make_PDU) #Funcion make_PDU
13      def make_PDU(self, msg_pmt):
14      # Toma Pmt Symbol a String y luego a lista
15          msg_pmt = pmt.symbol_to_string(msg_pmt)
16      msg = [ord(c) for c in msg_pmt]
17      # Crea un PMT vacio
18      send_pmt = pmt.make_u8vector(len(msg), ord(' '))
19      # Copia caracteres a u8vector
20      for i in range(len(msg)):
21          pmt.u8vector_set(send_pmt, i, msg[i])
22      # Envia mensaje
23      self.message_port_pub(pmt.intern('out'), pmt.cons(pmt.PMT_NIL, send_pmt))
```

Finalmente, se procede a la conformación del transmisor y receptor para realizar las pruebas con los equipos USRP el cual se observa en la Figura 3.4.

3.1.5. PRUEBAS DE FUNCIONAMIENTO UTILIZANDO LOS EQUIPOS USRP 2920 EN UN AMBIENTE INDOOR

Para comprobar el funcionamiento del sistema, se realizaron pruebas variando la distancia entre el transmisor y receptor. Para esto se utilizaron:

- Dos computadores
- Dos equipos USRP NI 2920
- Dos antenas VERT900
- Dos cables Gigabit Ethernet

El primer escenario se realizará con una separación entre los equipos USRP de 3 y 6 metros con línea de vista y el segundo escenario con una separación de 6 metros sin línea de vista. La Figura 3.5 y la Figura 3.6 muestra los escenarios respectivamente.

La ganancia en transmisión y recepción se estableció inicialmente en 10dB y el intervalo de generación del mensaje cada 2 segundos.

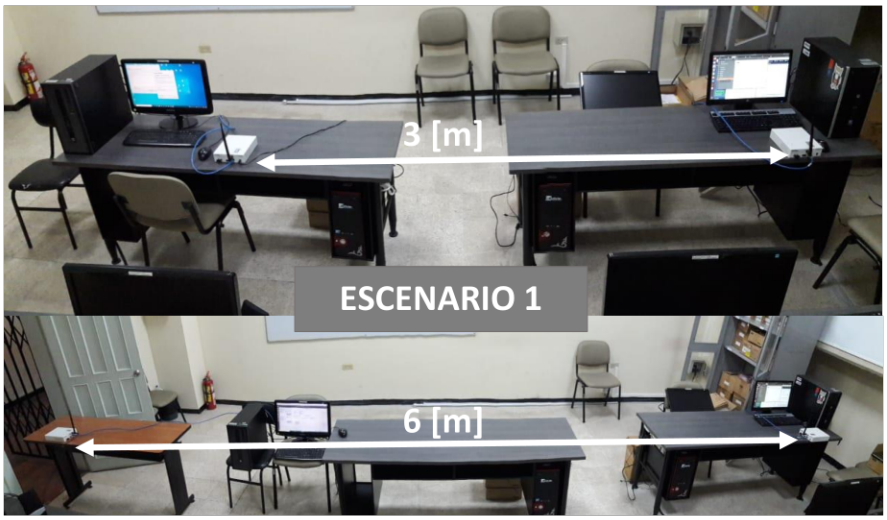

Figura 3.5: Escenarios con línea de vista

Figura 3.6: Escenario sin línea de vista

Una vez configurado y puesto en marcha el sistema, se observa el mensaje a la salida de la capa física en recepción. La Figura 3.7 muestra la longitud de la trama y la carga útil.
En la Tabla 3.2 se observa que para cada escenario fue necesario aumentar la ganancia, esto con el fin de lograr la recepción del mayor número de paquetes posible y así obtener un valor de LQI promedio válido. Sino llega una paquete , no se genera un LQI en recepción, por tanto no se puede hacer una valoración total en ese sentido del sistema. Para el presente proyecto se tuvo un LQI máximo de 254,5324.

En el escenario de 3 y 6 metros con línea de vista , se observa claramente que en primer

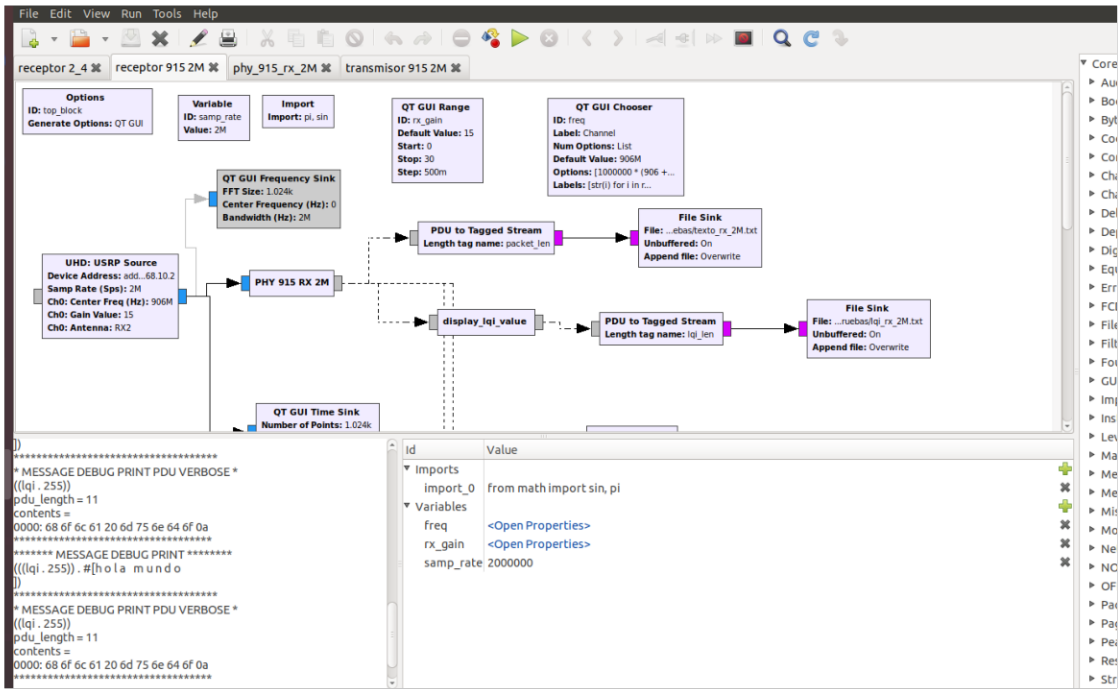

Figura 3.7: Recepción de paquetes

Tabla 3.2: Resultados de las pruebas de funcionamiento con dispositivos USRP

Distancia [metros]	Gtx, Grx [dB]	Paquetes	Paquetes Rx	BER	LQI
3	1	700	-	-	-
	10	700	OK	6.9796e-04	254.4484
6	10	700	-	-	-
	15	700	OK	6.9535e-04	254.3356
6 (sin línea de vista)	15	700	-	-	-
	20	700	OK	6.623e-04	254.5324

caso llegan los paquetes y en el segundo no llega ninguno a la potencia de 10 dB. Por tanto, a medida que aumenta la distancia se produce disipación de la potencia radiada.

Entre las causas que producen esta disipación también esta la reflexión de las señales en techo, paredes y obstáculos físicos.

3.2. CONCLUSIONES

- Al revisar las especificaciones de la capa física para las bandas de 2450, 915 y 868 MHz, se pudo determinar las características particulares que se debe tener en cuenta para diferenciar la implementación de cada banda en SDR. Entre las cuales esta: número de chips por cada símbolo de datos, número de muestras de medio pulso seno, frecuencia de muestreo, número de canales, velocidad de símbolo de datos, ancho de banda, entre otros.

- Gracias a las facilidades que presenta GNU Radio al ser un entorno de desarrollo libre y de código abierto, se logró diseñar e implementar el receptor con la ayuda de bloques disponibles en su librería y con bloques creados en base a las necesidades de procesamiento de la señal en lenguajes de programación Python y C++. Además del beneficio en el uso de funciones disponibles en el github de GNU Radio.

- Al implementar la etapa de demodulación QPSK en el receptor se puede concluir que los chips utilizados en transmisión pueden ser empleados en la recuperación de símbolos de datos en el receptor. Sin verse obligados a calcular otros chips para la etapa mencionada.

- La frecuencia de muestreo juega un papel muy importante en la implementación de prototipos de comunicación en SDR ya que el trabajar con frecuencias elevadas por sobre los 4 MHz, perjudica el procesamiento de datos por parte del ordenador;ocasionando pérdida de paquetes debido al desbordamiento del búfer.

- Debido a la imposibilidad de transmitir un archivo de texto de 600 Kbytes con los USRP. Se puede concluir que la tecnología ZigBee esta diseñada para transmitir pequeños paquetes de información, a bajas velocidades de transferencia de datos.

- En los distintos escenarios de pruebas se evidenció que la potencia con la que llega las señales en el receptor, disminuye al aumentar la distancia y a la presencia de obstáculos físicos.

- El parámetro LQI no permite caracterizar una transmisión en conjunto, ya que los paquete perdidos no se genera ningún valor de LQI en el sistema. Mas bien es un indicador singular de la calidad de un paquete recibido.

- De acuerdo a la baja tasa de bis errados y a una tasa de paquetes erróneos PER

menor al 1 % obtenidos en las pruebas de funcionamiento. Se concluye que el sistema transmisor receptor implementado cuenta con un funcionamiento adecuado en la transmisión de datos.

3.3. RECOMENDACIONES

- Para instalar las dependencias UHD para los dispositivos USRP, se recomienda realizar la instalación desde código fuente en el github de Ettus Research. La construcción desde el código fuente permitirá realizar modificaciones en las versiones de las imágenes FPGA para que sean compatibles, con las vesiones de GNU Radio y Ubuntu.
- Para evitar las notas Overflow/Underflow en la transmisión de datos, se recomienda: utilizar la herramienta GNU Radio en forma nativa y utilizar ordenadores con procesadores y memoria RAM recientes.
- Al diseñar e implementar sistemas de comunicaciones en SDR, es necesario delimitar la frecuencia de muestreo a utilizar. Ya que frecuencias por sobre los 4MHz generan un alto costo de procesamiento en los ordenadores , propiciando la pérdida de paquetes de datos.
- Para calcular el valor promedio de LQI en una trasmisión en entorno de simulación, se recomienda verificar que todos los paquetes de datos perteneciente al archivo de texto lleguen al receptor, para garantizar que por cada paquete de datos se obtenga un valor de LQI y así obtener un valor representativo de toda la transmisión.
- El estándar IEEE 802.15.4 ofrece diferentes tipos de modulaciones para la capa física, los cuales se podrían implementar en trabajos futuros y así poder realizar la comparación del desempeño en conjunto.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] W. I. Forum, “Introduction to SDR - Wireless Innovation Forum,” 2020. [Online]. Disponible en: https://www.wirelessinnovation.org/Introduction_to_SDR

[2] C. Wang, T. Jiang, and Q. Zhang, *ZigBee Network Protocols and Applications*, primera ed. Boca Raton: CRC Press Taylor and Francis Group, Jan. 2014, vol. 1.

[3] C. S. IEEE, “Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs),” p. 323, Sep. 2006. [Online]. Disponible en: http://www.ismlab.usf.edu/dcom/Ch8_802.15.4-2006.pdf

[4] T. S. Nesl, “GNU Radio 802 . 15 . 4 En-and Decoding.” Los Angeles California: University of California, 2006.

[5] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer, “A GNURadio-based IEEE 802.15.4 Testbed,” *University of Innsbruck*, p. 4, 2013. [Online]. Disponible en: https://www.ccs-labs.org/bib/bloessl2013gnu/bloessl2013gnu.pdf

[6] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, “Study on zigbee technology,” in *2011 3rd International Conference on Electronics Computer Technology*, vol. 6, 2011, pp. 297–301.

[7] O. Hersent, D. Boswarthick, and O. Elloumi, *The Internet of Things Key Applications and Protocols*, primera ed. United Kingdom: Jhon Wiley and Sons Ltd, 2012.

[8] J. G. Proakis and M. Salehi, *Digital communications*, 5th ed. Boston: McGraw-Hill, 2008.

[9] E. Grayver, *Implementing Software Defined Radio*. Springer Science & Business Media, Jul. 2012, google-Books-ID: FwVpvXuqWQsC.

[10] D. G. Gómez, J. M. R. Salís, and P. García-del Pino, “Implementación y configuración de un receptor de radio definido por software (SDR) para estudios de propagación,” p. 4.

[11] A. M. Wyglinski and D. Pu, *Digital Communication Systems Engineering with Software-Defined Radio*. Artech House, 2013, google-Books-ID: 3z2uklY5O3oC.

[12] N. Instruments, “What Is NI USRP Hardware?” Mar. 2019. [Online]. Disponible en: https://www.ni.com/es-cr/innovations/white-papers/11/what-is-ni-usrp-hardware-.html

[13] E. R. Brand, a National Instruments, “SDR Software - GNU Radio,” 2019. [Online]. Disponible en: https://www.ettus.com/sdr-software/gnu-radio/

[14] R. E. Erreyes Cedeño and M. N. López Núñez, “Implementación de un prototipo de sistema mimo-ofdm 2x2 basado en sdr mediante gnu radio,” B.S. thesis, Quito, 2019., 2019.

[15] G. Radio, “GNU Radio - The Free & Open Source Radio Ecosystem · GNU Radio,” 2022. [Online]. Disponible en: https://www.gnuradio.org/

[16] E. Gonzalez, C. Zerbini, G. Riva, D. Rosso, F. Suarez, and L. Guanuco, “Sdr con gnu radio: de la teorıa a la aplicacion.”

[17] G. Radio, “GNU Radio Manual and C++ API Reference: Components,” 2019. [Online]. Disponible en: https://www.gnuradio.org/doc/doxygen/page_components.html#components_blocks

[18] M. A. S. Caro and C. E. S. Cano, “Diseño de un concentrador para aplicaciones de redes de sensores con capacidades de reconfiguración de parámetros por software,” Ph.D. dissertation, UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS, Bogotá, 2016. [Online]. Disponible en: http://repository.udistrital.edu.co/bitstream/11349/2731/1/SastoqueCaroMiguelAngel2016.pdf

[19] G. Cornetta, “Lección 6: Demodulación y Detección en Banda Base. Parte II,” p. 27, 2012.

[20] E. Research, “GitHub - EttusResearch/uhd at v3.14.0.0,” 2014. [Online]. Disponible en: https://github.com/EttusResearch/uhd

[21] MathWorks, “Number of bit errors and bit error rate (BER) - MATLAB biterr,” 2006. [Online]. Disponible en: https://www.mathworks.com/help/comm/ref/biterr.html

[22] E. Research, “USRP Hardware Driver and USRP Manual: General Application Notes.” [Online]. Disponible en: https://files.ettus.com/manual/page_general.html

5. ANEXOS

Los siguientes anexos representan los códigos de los bloques implementados en GNU Radio, los cuales se encuentran en formato digital:

ANEXO I. *pmt_symbol_to_pdu.py*

ANEXO II. *decimador_cc_impl.cc*

ANEXO III. *qpsk_dem_py_cb.py*

ANEXO IV. *my_packet_sink_impl.cc*

ANEXO V. *packe_sink_915_impl.cc*

ANEXO VI. *display_lqi_value.py*