UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

"IMPLEMENTACIÓN DEL ALGORITMO WATER-FILLING PARA LA ASIGNACIÓN DE POTENCIA ÓPTIMA EN EL CANAL INALÁMBRICO DE UN SISTEMA MIMO 2X2 SOBRE SDR"

Presentado por:

Bach. Miguel Angel Champi Castro

Para optar al Título Profesional de **Ingeniero Electrónico**

Asesor:

Mgt. Ing. Jorge Luis Arizaca Cusicuna

Financiada por: Programa "YACHAYNINCHIS WIÑARINANPAQ" UNSAAC

Cusco - Perú

2024

INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo de investigación/tesis titulada: IMPLEMENTACIÓN DEL ALGORITMO WATER-FILLING PARA LA ASIGNACIÓN DE POTENCIA ÓPTIMA EN EL CANAL INALÁMBRICO DE UN SISTEMA MIMO 2X2 SOBRE SDR

presentado por: MIGUEL ANGEL CHAMPI CASTRO con DNI Nro.: 71709975 presentado por: ..... con DNI Nro.: ..... para optar el título profesional/grado académico de INGENIERO ELECTRÓNICO

Informo que el trabajo de investigación ha sido sometido a revisión por TRES veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 8 **%.**

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No se considera plagio.	X
Del 11 al 30 %	Devolver al usuario para las correcciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica para que tome las acciones correspondientes. Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de asesor, firmo el presente informe en señal de conformidad y **adjunto** la primera página del reporte del Sistema Antiplagio.

Cusco, 6 de Junio de 2024

Firma

Post firma JORGE LUIS ARIZACA CUSICUNA

Nro. de DNI 42348906

ORCID del Asesor 0000-0003-2658-5492

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: **oid:** 27259:359346037

NOMBRE DEL TRABAJO

Tesis CHAMPI CASTRO, MIGUEL ANGEL Final.pdf

AUTOR

Miguel Champi Castro

RECUENTO DE PALABRAS

23238 Words

RECUENTO DE CARACTERES

114446 Characters

RECUENTO DE PÁGINAS

94 Pages

TAMAÑO DEL ARCHIVO

16.6MB

FECHA DE ENTREGA

Jun 5, 2024 9:43 AM GMT-5

FECHA DEL INFORME

Jun 5, 2024 9:45 AM GMT-5

8% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base de datos.

- 6% Base de datos de Internet
- 1% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossref
- 6% Base de datos de trabajos entregados

Excluir del Reporte de Similitud

- Material bibliográfico
- Coincidencia baja (menos de 10 palabras)

DEDICATORIA

Esta tesis es dedicada a mis padres, quienes me guiaron con su sabiduría, por sus constantes sacrificios y apoyo que me han dado.

Índice

Índice de figuras

Índice de tablas

Lista de segmentos de código

Resumen

En el presente trabajo se implementa el algoritmo de *water-filling* para asignación de potencia óptima sobre un sistema MIMO-OFDM 2x2, utilizando algoritmos de estimación del estado del canal (CSI), basado en la tecnología de Radio Definida por Software (SDR).

Primero se realizó un estudio de las comunicaciones MIMO y los parámetros de los que depende como el conocimiento del estado del canal (CSI) y su aplicación para la asignación de potencia óptima conocida como *water-filling*. También se hizo una revisión de la modulación por Multiplexación por División de Frecuencias Ortogonales (OFDM), ya que es el tipo de modulación más utilizada en redes LTE y módems Wi-Fi MIMO.

Luego se hizo un diseño del sistema MIMO-OFDM 2x2 en la plataforma de GNU Radio, donde se utilizó el estándar 802.11n para el formato de las tramas y portadoras. Después, se implementó el algoritmo junto con las radios NI USRP 2920 y 2950R. Y finalmente, se hicieron pruebas de distancia a 1, 1.5, 2 y 3 metros, donde se midieron y compararon de las tasas de error de bit(BER), obteniendo una mejora del 25 % en promedio entre un sistema MIMO ordinario y una sistema MIMO con algoritmo de *water-filling*, donde la asignación de potencia es mostrada por medio de gráficas de barras a partir de la descomposición paralela del canal MIMO 2x2. Resultando en un aumento de la tasa de datos, uso de potencia de manera eficiente y un sistema MIMO con capacidad de adaptarse a los cambios de canal.

Palabras clave: MIMO, SDR, OFDM, Waterfilling

Introducción

El uso de múltiples antenas en el transmisor y receptor en sistemas inalámbricos, mejor conocido como MIMO, puede lograr incrementos de tasas de datos y fiabilidad en comunicaciones móviles, sin requerir de uso adicional de ancho de banda. No obstante, el uso de múltiples antenas de radio hace incurrir a un mayor consumo de potencia.

El rendimiento de los sistemas MIMO depende del conocimiento del canal, el cual es caracterizado como múltiples trayectos entre antenas transmisoras y receptoras, siendo que cada uno de estos trayectos presentan diferentes condiciones. Asumiendo un perfecto conocimiento del canal, un sistema MIMO convencional no efectúa un control de potencia que permita mejorar la señales de información sobre los canales que presenten mejores condiciones, representando un uso ineficiente de este recurso. Por lo que para optimizar el consumo de potencia del lado del transmisor, se requiere una estrategia de control de potencia que permita mejorar el rendimiento del sistema, esto en base a las condiciones del canal. Para lo cual, se utiliza el algoritmo de *water-filling*, que a partir de la descomposición paralela del canal MIMO, reasigna de manera óptima la potencia hacia las antenas transmisoras en base a la relación señal a ruido (SNR). Siendo que los canales descompuestos que presenten un SNR alto, la antena que transmite por dicho medio le será asignado una mayor parte de la potencia, pero si presentan un SNR bajo, le será asignado menor potencia.

En la presente tesis se implementa y evalúa el algoritmo de *water-filling* para la asignación de potencia óptima en sistema de comunicación inalámbrica MIMO 2x2 con modulación OFDM basado en el estándar de WiFi 802.11n, esto mediante tecnología de Radio Definida por Software (SDR) utilizando hardware USRP y software GNU Radio.

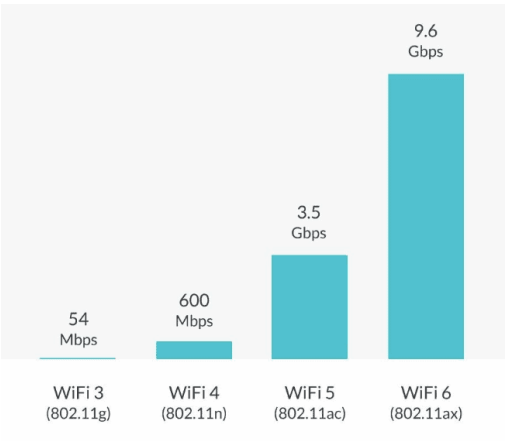
CAPITULO I

1. Generalidades

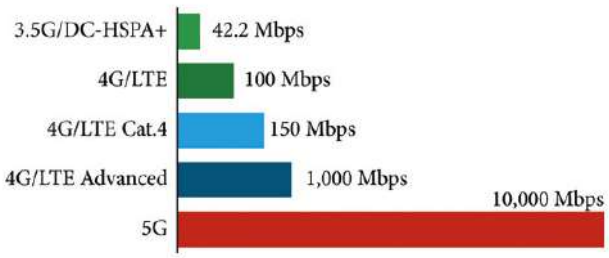
1.1. Planteamiento del problema

1.1.1. Problemática

El desarrollo de las comunicaciones inalámbricas ha estado en constante crecimiento y mejora conforme al surgimiento constante de nuevas e innovadoras tecnologías que demandan más información y calidad en las comunicaciones. Estas mejoras se basan en tres objetivos: Mayor cantidad de información, mayores distancias y mejor calidad [1]. La actual demanda de tecnología móvil y el acrecentamiento de las tasas de transmisión han impulsado el trabajo y la investigación de esta tendencia en las telecomunicaciones, planteando nuevos métodos de transmisión que mejoran la tasa de transferencia y la eficiencia espectral sin afectar la calidad del servicio o el ancho de banda limitado disponible [2]. Además, también se tiene el medio de transmisión inalámbrico que es el aire, considerado un medio hostil, ya que degrada la señal por constantes interferencias y ruido introducido. Para mantener la calidad se necesita aplicar técnicas de codificación y corrección de errores, cuyo efecto resulta en una reducción de la tasa de error [1].



(a) Tasas de velocidad en WiFi.



(b) Tasas de velocidad en redes móviles.

Figura 1: Evolución de las tasas de velocidad.

MIMO (Multiple Input-Multiple Output) es una tecnología de comunicación inalámbrica que puede ser utilizada para incrementar la tasa de datos. El coste de las mejoras en la

comunicación obtenidas por técnicas MIMO es el despliegue de múltiples antenas en el lado del transmisor o en el receptor, lo que implica un aumento del espacio utilizado y mayor requerimiento de consumo del sistema [3].

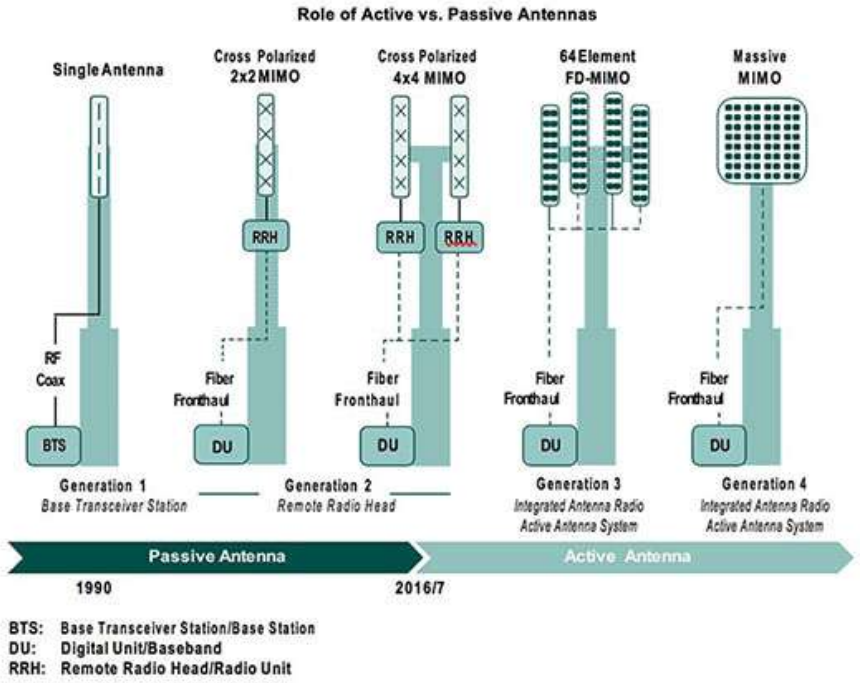


Figura 2: Evolución de comunicaciones MIMO.

La capacidad de un sistema MIMO puede incrementarse si se tiene conocimiento del estado de información del canal caracterizado como múltiples trayectos entre antenas transmisoras y receptoras, de esta manera se puede aplicar una estrategia de asignación óptima de potencia conocida como water filling [4], [5], que distribuye automáticamente un porcentaje de la potencia a los canales dependiendo de las condiciones que presenten, lo que mejora la eficiencia del consumo frente a un sistema ordinario. Para mantener un control adecuado de la potencia que hacen uso los sistemas multicanal MIMO, se hace necesario aplicar un algoritmo de asignación de potencia óptima, en base a la relación señal a ruido, utilizando una plataforma SDR que permita implementar y evaluar el prototipo de comunicación inalámbrica en un entorno real.

1.1.2. Formulación del problema general

¿Cómo se puede aplicar el algoritmo de *water-filling* en un sistema de comunicación MIMO 2x2 para la asignación de potencia óptima en el canal inalámbrico?

1.1.3. Problemas específicos

- Los sistemas de comunicación inalámbrica MIMO experimentan disminuciones en la tasa de transferencia debido a que carecen de una estrategia de asignación de potencia que haga frente a las condiciones adversas del canal inalámbrico.
- La plataforma de SDR no posee técnicas de procesamiento digital de señales para implementar tecnología MIMO.
- No se cuenta con un sistema MIMO 2x2 sobre SDR validado dentro de un ambiente controlado.
- No se tiene registro de desempeño del algoritmo de *water-filling* implementado en un sistema MIMO 2x2 sobre SDR.

1.2. Objetivos

1.2.1. Objetivo general

Aplicar en un prototipo de comunicación MIMO 2x2 el algoritmo de *water-filling* para la asignación de potencia óptima en el canal inalámbrico.

1.2.2. Objetivos específicos

- Estudiar la tecnología MIMO y los parámetros que intervienen en el control de asignación de potencia a las antenas transmisoras.
- Diseñar e integrar técnicas de procesamiento digital de señales para tecnología MIMO sobre la plataforma de SDR.
- Desarrollar un sistema de comunicación inalámbrica MIMO 2x2 sobre la plataforma de SDR.
- Implementar y evaluar el desempeño del algoritmo de *water-filling* en el sistema de comunicación MIMO 2x2.

1.3. Justificación

El uso de tecnologías MIMO ha permitido el incremento de las tasas de transmisión en comunicaciones inalámbricas actuales, siendo utilizados en redes de telefonía móvil 4G LTE, redes LAN inalámbricas como WiFi, y siendo parte fundamental para la implementación del 5G junto a IoT (Internet de las cosas), el cual caracteriza una tasa masiva de transferencia de datos, razón por la que es importante realizar investigación de nuevas técnicas y en conjunto con las ya existentes poder mejorar los resultados. Si bien su funcionalidad es incrementar la capacidad de datos o presentar fiabilidad en la comunicación, la principal limitación es la potencia como recurso que se distribuye para los distintos canales generados, siendo que alguno de estos se presente como uno muy deficiente, el consumo de potencia para que este canal no es conveniente y se desperdicia, entonces se tiene que priorizar aquellos que presenten mejores condiciones para mantener la comunicación asignándoles una mayor parte de la potencia, por lo que requiere aplicar un sistema de control que optimice la potencia asignada a cada canal de transmisión en base a la relación señal a ruido SNR de cada canal. El uso de tecnología SDR ha abierto las posibilidades de cohesionar diversos sistemas para emular la capa física en una sola arquitectura de software reprogramable, para ello utilizando software de procesamiento digital de señales capaz de interactuar con plataformas SDR para diseño y prototipado de diversas tecnologías de comunicación inalámbricas. Con las herramientas ya mencionadas es posible evaluar y validar distintos trabajos de investigación en comunicaciones inalámbricas en un ambiente real utilizando hardware de bajo costo.

1.4. Alcances

Con el desarrollo de la presente tesis se permitirá:

- Establecer un modelo de comunicación inalámbrica MIMO 2x2 basado en SDR para futuras investigaciones.
- Comprender la implementación de estrategias de asignación de potencia para sistemas MIMO basadas en Radio Definida por Software.

1.5. Limitaciones

- El sistema implementado estará limitado por las capacidades de procesamiento del propio dispositivo SDR a ser utilizado.
- Los resultados a obtener estarán restringidos en un entorno de laboratorio - y no a pruebas de campo - debido a que el experimento se realizará en un ambiente interno controlado.

1.6. Metodología

Este trabajo de tesis se encuentra clasificado dentro de una investigación experimental aplicada de enfoque cuantitativo, ya que se vale de datos cuantificables, accesibles por medio de observaciones y mediciones.

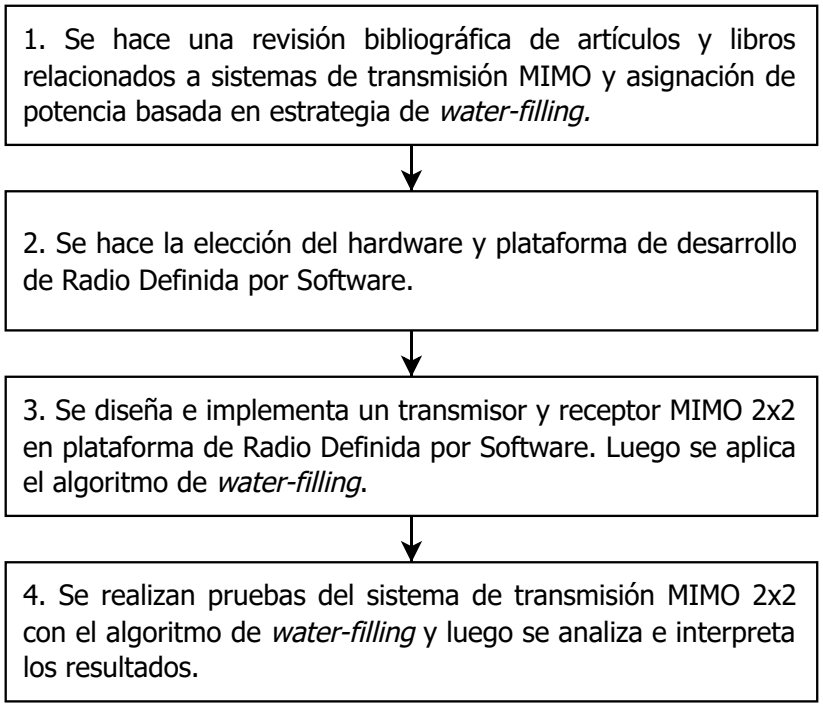


Figura 3: Diagrama de bloques de la metodología. Elaboración propia.

1.7. Variables e Indicadores

$$\frac{P_i}{P} = \begin{cases} 1/\gamma_0 - 1/\gamma_i & \gamma_i \geq \gamma_0 \\ 0 & \gamma_i < \gamma_0 \end{cases} \tag{1}$$

Variables independientes

- P: Potencia total de transmisión del sistema MIMO que se reparte entre las antenas.

Variables dependientes

- γ_i: Relación señal a ruido (SNR) de los i canales paralelos.
- γ_0: Umbral de relación señal a ruido (SNR) mínimo para distribuir potencia a cada antena.
- P_i: Potencia óptima a asignar a cada antena transmisora.

Indicadores

- BER: Tasa de error de bit del sistema.

CAPITULO II

2. Marco Teórico

2.1. Antecedentes

Dalveer Kaur y Neeraj Kumar en el artículo "Enhance the Capacity of MIMO Wireless Communication Channel using SVD and Optimal Power Allocation Algorithm", que tiene como objetivo la asignación de potencia óptima a cada flujo de datos de una comunicación MIMO, utilizando la descomposición de valores singulares (SVD) para descomponer paralelamente el canal MIMO en presencia del estado de información del canal (CSI). Se comparó mediante simulaciones la capacidad de transmisión con diferentes números de antenas transmisoras y receptoras en base a la relación señal a ruido (SNR). Como resultado se obtuvo una mejora de la capacidad del canal pero un elevado SNR que eclipsa la presencia del CSI. [6]

Lamia Grira y Ridha Bouallegue en el artículo "Using Water Filling technique and SVD decomposition for cooperative node selection in WSN", que tiene como objetivo maximizar la capacidad de transmisión de una red de nodos cooperativos utilizando descomposición de valores singulares y algoritmo de water-filling en 2 escenarios con desvanecimiento. Se comparó mediante simulaciones la selectividad de nodos en base a la máxima capacidad de canal con y sin los algoritmos de water-filling para configuraciones MIMO. Como resultado se obtuvo una mejora en capacidad de un sistema de nodos cooperativos. [7]

Harsha Gurdasani, A. G. Ananth y Thangadurai en el artículo "Channel Capacity Enhancement of MIMO System using Water-Filling Algorithm", que tiene como objetivo es analizar la tasa de error de bit (BER) de bloques de código espacio-temporales (STBC) y capacidad de canal MIMO mejorado utilizando el algoritmo de water-filling con distintos ecualizadores en el receptor. Se comparó mediante simulaciones la BER y la capacidad de canal en base a la relación señal a ruido (SNR) para distintas configuraciones. Como resultado se obtuvo una mejora de la capacidad en base a la correlación del transmisor para distintos parámetros como distancia y SNR. [8]

2.2. Definición de la descomposición de valores singulares

Sea $A \in \mathbb{C}^{m,n}$ que denota una matriz $m \times n$ de componentes complejas con $m \geq n$, y r como el rango de la matriz A. Entonces existen las matrices unitarias $U \in \mathbb{C}^{m,r}$ y $V \in \mathbb{C}^{r,n}$ con columnas ortonormales tal que:

$$A = U\Sigma V^H \text{ con } \quad \Sigma = \begin{bmatrix} \Sigma_r & 0_{r,n-r} \\ 0_{m-r,r} & 0_{m-r,n-r} \end{bmatrix} \in \mathbb{R}^{m,n}, \tag{2}$$

$$\Sigma_r = \text{diag}(\sigma_1, \ldots, \sigma_r), \text{ donde } \sigma_1 \geq \sigma_2 \geq \sigma_3, \ldots, \sigma_r \geq 0$$

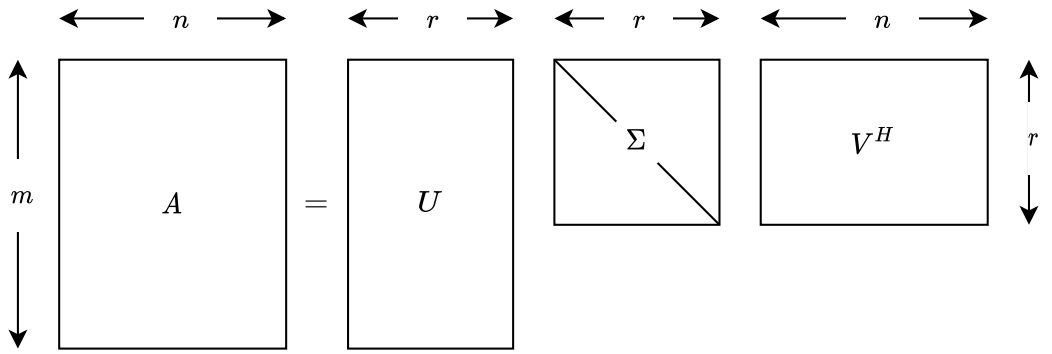


Figura 4: Descomposición de valores singulares. Elaboración propia

Si tenemos en cuenta que $A^H A \in \mathbb{C}^{n,n}$ es hermitiana y $\text{rango}(A^H A) = \text{rango}(A) = r$, por lo tanto la matriz $A^H A$ tiene exactamente r autovalores positivos $\lambda_1, \ldots, \lambda_r$, mientras que los $n - r$ restantes son cero. Entonces se define:

$$\sigma_j = \lambda_j^{1/2}, \; j = 1, \ldots, r \tag{3}$$

Los elementos diagonales de la matriz Σ_r son llamados valores singulares. Las columnas de U son llamados los vectores singulares izquierdos de A, mientras que las columnas de V son llamados vectores singulares derechos de A. Los valores singulares son definidos como las raíces cuadradas positivas de los autovalores de $A^H A$. La demostración de la descomposición de valores singulares se encuentra en el Anexo D.

2.3. Modulación PSK y QAM

Modulación por desplazamiento de fase

Modulación por desplazamiento de fase (PSK), consiste en el desplazamiento de fase de los símbolos, siendo que se pueden modular *M* símbolos a partir de $log_2(M)$ bits de información. En este esquema de modulación, las formas más comunes de modular PSK es mediante la fase de la señal que es representada por el ángulo alrededor de un círculo, y la amplitud constante como la distancia del origen o centro del círculo.

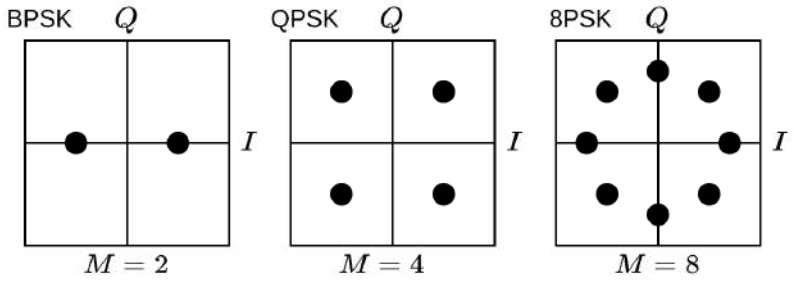


Figura 5: Constelaciones PSK de orden mayor. Elaboración propia

Modulación de amplitud en cuadratura

Modulación de amplitud en cuadratura (QAM), consiste en el desplazamiento de amplitud y de fase de forma independiente para cada símbolo, siendo que se pueden modular *M* símbolos a partir de $log_2(M)$ bits de información, este tipo de modulación es capaz de transmitir altas tasas de datos a comparación de los demás esquemas de modulación ordinarios, como PSK o ASK, ya que su diagrama de constelación presenta varios símbolos que pueden ser utilizados. A pesar de que los datos son binarios, las formas más comunes de QAM son formar una constelación cuadrada con un número de símbolos equivalente a potencias de 2, por ejemplo: 4QAM, 16QAM, 64QAM, etc.

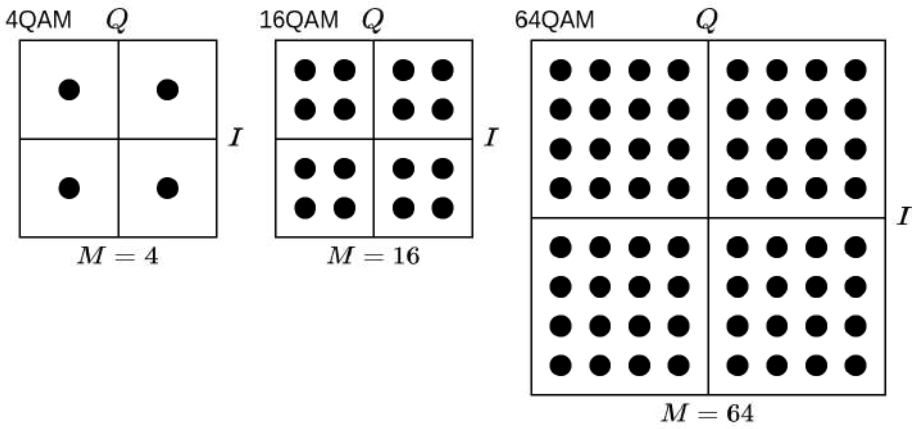


Figura 6: Constelaciones QAM de orden mayor. Elaboración propia

2.4. MIMO

MIMO es una tecnología de comunicaciones inalámbricas, donde se tienen múltiples antenas para trasmitir como para recepcionar, que puede mejorar las velocidades de datos mediante la multiplexación o mejorar el rendimiento por diversidad. Estas ganancias de eficiencia espectral usualmente requiere tener un conocimiento preciso del canal en el receptor y a veces también en el transmisor. El costo de mejorar el rendimiento obtenido mediante técnicas MIMO es el costo adicionado de desplegar múltiples antenas, el espacio y requerimiento de circuitos de potencia extra de cada antena, y también la complejidad requerida para procesamiento de señales multidimensional. [3]

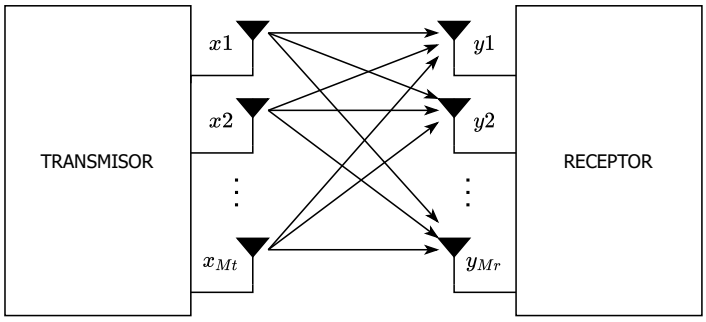


Figura 7: Sistema MIMO. [3]

Su funcionamiento se basa en aprovechar la diversidad de caminos de propagación generados por el uso de múltiples antenas, para mejorar el rendimiento, cobertura y confiabilidad. Aunque este tipo de sistemas provean una alta velocidad de transmisión, dependen de diversas estrategias o técnicas que mejoran los esquemas de transmisión o la fiabilidad del enlace, siendo estas: diversidad y multiplexación espacial.

Ganancia por diversidad espacial

Utilizando diversidad se tiene una mejora en la fiabilidad del sistema, donde se mejora el SNR promediado con el tiempo. Esta técnica transmite flujos de información iguales por cada una de las antenas, por lo que cada uno sufrirá desvanecimiento de manera distinta. Se garantiza que al menos una de las copias de la señal sufrirá menos desvanecimiento que las demás, por tanto, se tiene una probabilidad mayor de que la información se recepcione adecuadamente.

Ganancia por multiplexación espacial

Utilizando multiplexación espacial se tiene la maximización de la tasa de transmisión, es decir, la eficiencia espectral. Esta técnica transmite flujos de información independientes por cada antena de manera simultánea, con todos ocupando el mismo ancho de banda y la misma ranura temporal (time slot). Siendo que los flujos sufren desvanecimiento de manera distinta, con esta técnica se corre el riesgo de perder información en la comunicación.

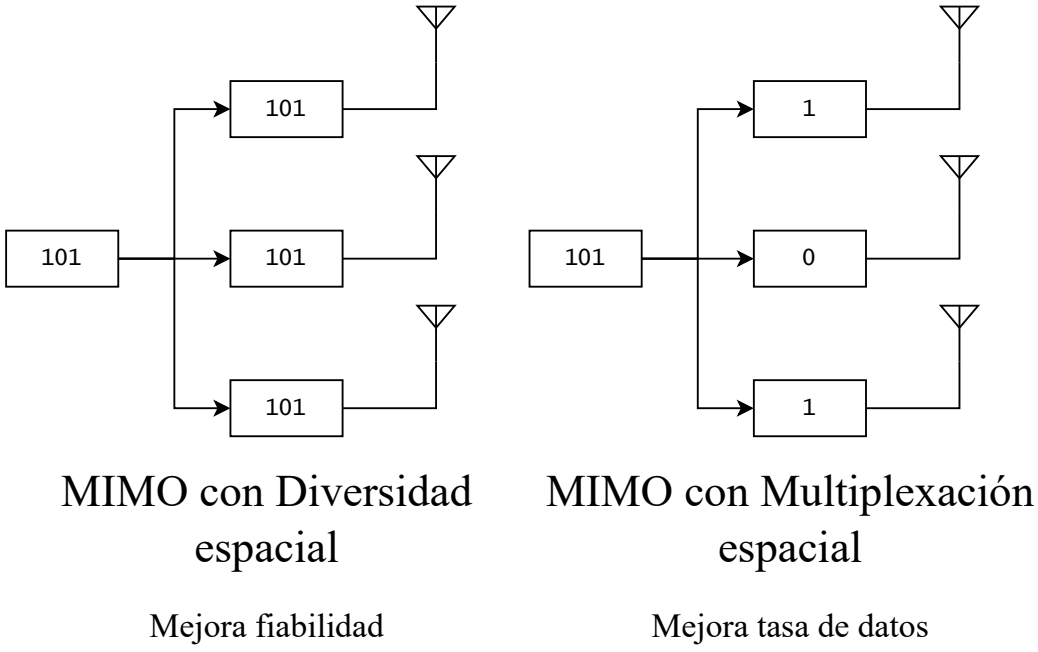


Figura 8: Comparación diversidad espacial y multiplexación espacial. Elaboración propia.

2.4.1. Modelo de canal MIMO

Para modelar un sistema MIMO haremos uso de un sistema de transmisión de punto a punto de banda estrecha con M_t antenas transmisoras y M_r antenas receptoras como se muestra en la figura 7. Este sistema puede ser representado por el siguiente modelo en tiempo discreto [3].

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{Mr} \end{bmatrix} = \begin{bmatrix} h_{11} & \cdots & h_{1Mt} \\ \vdots & \ddots & \vdots \\ h_{Mr1} & \cdots & h_{MrMt} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{Mt} \end{bmatrix} + \begin{bmatrix} n_1 \\ \vdots \\ n_{Mr} \end{bmatrix} \tag{4}$$

O simplemente como:

$$\mathbf{y = Hx + n}. \tag{5}$$

Donde:

- **x** representa los símbolos M_t-dimensional transmitidos
- **n** es el vector de ruido M_r-dimensional
- **H** es la matriz $M_r \times M_t$ de ganancias h_{ij} del canal para la antena receptora i y la antena transmisora j

Asumimos que el ancho de banda de canal **B** y ruido complejo Gaussiano de media cero y matriz de covarianza $\sigma_n \mathbf{I}_{M_r}$ donde $\sigma_n = N_0/2$, es la densidad de potencia espectral del ruido. También asumimos para una constante de potencia P con potencia de ruido σ_n^2, la relación señal a ruido (SNR) promedio para cada antena receptora bajo una ganancia de canal es:

$$\rho = P/\sigma_n^2 \tag{6}$$

2.4.2. Descomposición paralela de canal MIMO

La ganancia por multiplexación resulta del hecho de que un canal MIMO puede ser descompuesto en un número de R canales paralelos independientes. Mediante la multiplexación de los datos en estos canales independientes, se consigue multiplicar por R la tasa de datos en comparación a un sistema con una sola antena en el transmisor y receptor. Este incremento de la tasa de datos es llamado ganancia por multiplexación.

Consideramos un canal MIMO con una matriz de ganancia de canal **H** de $M_r \times M_t$ que es

conocida tanto en el transmisor y en el receptor. Y sea R_{H} el rango de $\mathbf{H}$, entonces para cualquier matriz $\mathbf{H}$ se puede obtener la descomposición de valores singulares (SVD) como:

$$\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{H} \tag{7}$$

Donde:

- $\mathbf{U}$ y $\mathbf{V}$ son matrices unitarias, de dimensiones $M_r \times M_r$ y $M_t \times M_t$ respectivamente.
- $\boldsymbol{\Sigma}$ es una matriz diagonal $M_r \times M_t$ de los valores singulares σ_i de $\mathbf{H}$

Estos valores singulares singulares tienen la propiedad de que $\sigma_i = \sqrt{\lambda_i}$ para cada λ_i siendo i el i-ésimo autovalor de $\mathbf{H}\mathbf{H}^{H}$ y R_{H} de estos autovalores son positivos y los $M_t - R_{\mathrm{H}}$ restantes son cero.

La descomposición paralela del canal es obtenido definiendo una transformación en el canal por medio de los siguientes procedimientos: **precodificación en el transmisor** donde la entrada $\mathbf{x}$ a las antenas es generada por una transformación lineal en el vector de entrada $\tilde{\mathbf{x}}$ como $\mathbf{x} = \mathbf{V}\tilde{\mathbf{x}}$ y la **decodificación en el receptor** en el que a la salida del canal $\mathbf{y}$ se multiplica por $\mathbf{U}^{H}$. como se muestra en la figura 9.

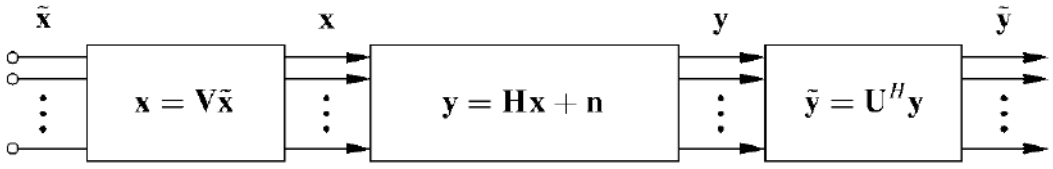


Figura 9: Precodificación en el transmisor y decodificación en el receptor. [3]

Este procedimiento de precodificación en el transmisor y decodificación en el receptor transforma el canal MIMO en un canal SISO de $R_{\mathbf{H}}$ canales paralelos de entrada $\tilde{\mathbf{x}}$ y salida $\tilde{\mathbf{y}}$, a partir de la SVD tenemos que:

$$\begin{aligned}
\tilde{\mathbf{y}} &= \mathbf{U}^{H}(\mathbf{H}\mathbf{x}+\mathbf{n}) \\
&= \mathbf{U}^{H}(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{H}\mathbf{x}+\mathbf{n}) \\
&= \mathbf{U}^{H}(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{H}\mathbf{V}\tilde{\mathbf{x}}+\mathbf{n}) \\
&= \mathbf{U}^{H}\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{H}\mathbf{V}\tilde{\mathbf{x}}+\mathbf{U}^{H}\mathbf{n} \\
&= \boldsymbol{\Sigma}\tilde{\mathbf{x}}+\tilde{\mathbf{n}}
\end{aligned} \tag{8}$$

Donde $\tilde{\mathbf{n}} = \mathbf{U}^H\mathbf{n}$ y $\mathbf{\Sigma}$ es la matriz de valores singulares de $\mathbf{H}$. Cabe mencionar que la multiplicación por una matriz unitaria no cambia la distribución del ruido, esto quiere decir que, $\mathbf{n}$ y $\tilde{\mathbf{n}}$ poseen distribución idéntica. Por tanto, la precodificación en el transmisor y la decodificación en el receptor transforman el canal MIMO en $R_\mathbf{H}$ canales paralelos independientes, donde el i-ésimo canal tiene entrada $\tilde{x}_i$, salida $\tilde{y}_i$, ruido $\tilde{n}_i$ y ganancia de canal σ_i. Esta descomposición paralela es mostrada en la figura 10.

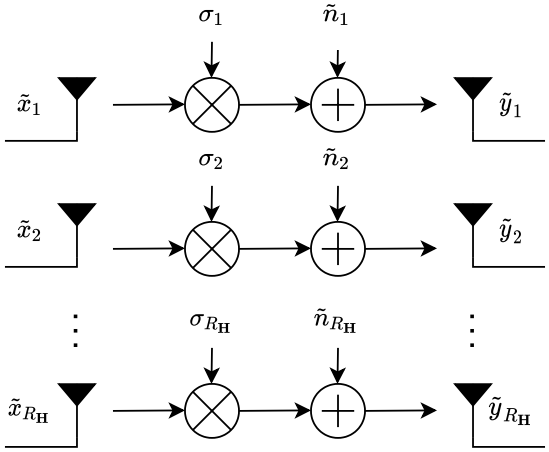


Figura 10: Descomposición paralela del canal MIMO. [3]

2.4.3. Canal conocido en el transmisor: Estrategia de water-filling

La descomposición MIMO descrita permite una simple caracterización de la capacidad del canal MIMO para una matriz $\mathbf{H}$ conocida en el transmisor y receptor. Específicamente, la capacidad es igual a la suma de las capacidades de cada canal paralelo independiente con la potencia de transmisión óptima asignada entre estos canales.

$$C = \max_{\rho_i:\sum_i \rho_i \leq \rho} \sum_{i=1}^{R_\mathbf{H}} B\log_2(1+\sigma_i^2\rho_i) \tag{9}$$

donde $R_\mathbf{H}$ es el número de valores singulares σ_i^2 diferente de cero de $\mathbf{H}$. Ya que el canal MIMO descompone en $R_\mathbf{H}$ canales paralelos, se puede decir que tiene $R_\mathbf{H}$ grados de libertad. Desde que $\rho = P/\sigma_n^2$, la capacidad 9 puede se expresado en términos de potencia asignada P_i del i-ésimo canal paralelo como:

$$C = \max_{\rho_i:\sum_i \rho_i \leq \rho} \sum_{i=1}^{R_\mathbf{H}} B\log_2(1+\frac{\sigma_i^2 P_i}{\sigma_n^2}) = \max_{\rho_i:\sum_i \rho_i \leq \rho} \sum_{i=1}^{R_\mathbf{H}} B\log_2(1+\frac{P_i\gamma_i}{P_n}) \tag{10}$$

donde $\gamma_i = \sigma_i^2 P/\sigma_n^2$ es el SNR asociado con el i-ésimo canal con potencia completa. Esta expresión indica que, con un SNR alto, la capacidad incrementa de manera lineal con el número de grados de libertad en el canal. En cambio, con un SNR bajo, toda la potencia será asignada al canal paralelo con el mejor SNR. La asignación de potencia óptima utilizando estrategia de waterfilling para un canal MIMO con un valor de corte γ_0 es:

$$\frac{P_i}{P} = \begin{cases} 1/\gamma_0 - 1/\gamma_i & \gamma_i \geq \gamma_0 \\ 0 & \gamma_i < \gamma_0 \end{cases} \tag{11}$$

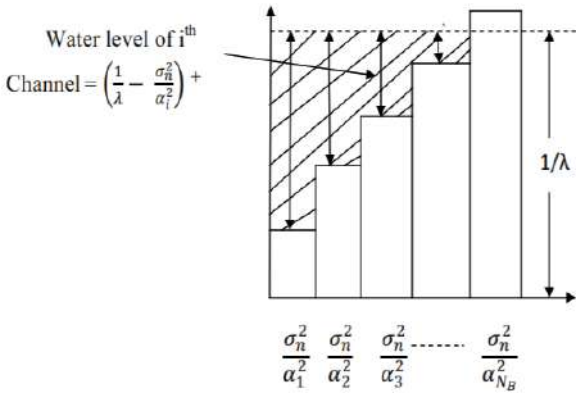


Figura 11: Diagrama de *water-filling* para sistemas MIMO. [6]

2.5. Multiplexación por división de frecuencias ortogonales

Multiplexación por división de frecuencias ortogonales (OFDM) es un tipo de modulación multiportadora, este tipo de modulación no utiliza un filtros individuales limitados en banda u osciladores para cada canal, el espectro de las subportadoras son superpuestas para hacer un uso eficiente del ancho de banda. La ortogonalidad entre señales es posible aplicando la transformada discreta de Fourier (DFT) y la transformada inversa discreta de Fourier (IDFT), las cuales pueden ser implementadas de manera eficiente utilizando la transformada rápida de Fourier (FFT) y la transformada inversa rápida de Fourier (IFFT), respectivamente [9].

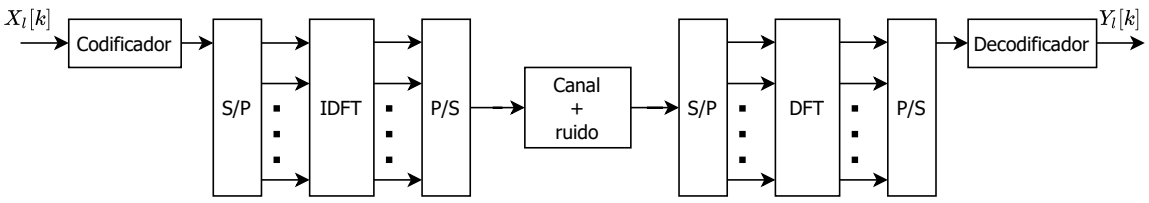


Figura 12: Implementación de OFDM usando IDFT/DFT. [9]

Como todas las portadoras son de duración finita T, el espectro de la señal OFDM se puede considerar como la suma de funciones *sinc* en el dominio de la frecuencia, desplazadas como se muestra en la figura 13, espaciadas por $1/T$.

Cada subportadora es limitada en el tiempo para cada símbolo. Una señal OFDM puede producir radiación fuera de su banda, en consecuencia provoca interferencia entre símbolos (ISI), como se muestra en la figura 14, donde los lóbulos laterales no son tan pequeños en comparación a los lóbulos principales.

Para mitigar el ISI, interferencias y efectos de multitrayecto se agrega un intervalo de guarda entre símbolos OFDM en el dominio del tiempo, que es el prefijo cíclico, del cual se hablará más adelante.

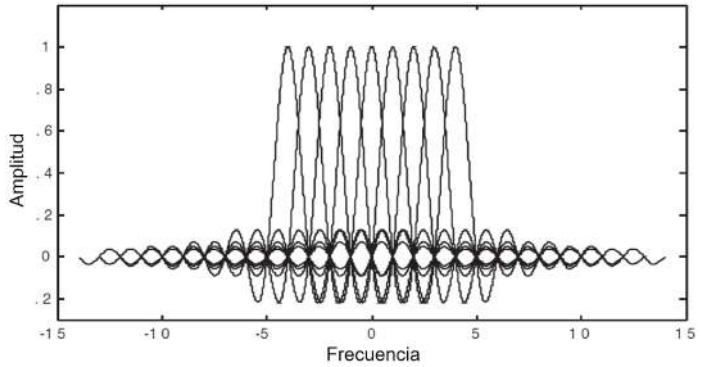


Figura 13: Espectro de una señal OFDM. [9]

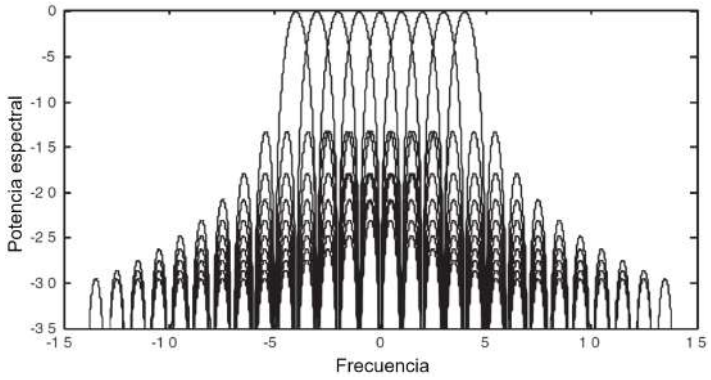


Figura 14: Potencia espectral de una señal OFDM. [9]

2.5.1. Modulación y demodulación OFDM

Principio de ortogonalidad

Se define que 2 señales son ortogonales si la integral de los productos para su periodo fundamental es cero, esto es:

$$\frac{1}{T_{sym}}\int_0^{T_{sym}} e^{j2\pi f_k t} e^{-j2\pi f_i t} dt = \frac{1}{T_{sym}}\int_0^{T_{sym}} e^{j2\pi \frac{k}{T_{sym}} t} e^{-j2\pi \frac{i}{T_{sym}} t} dt$$
$$= \begin{cases} 1, & \forall \text{ entero } k = i \\ 0, & \text{otro caso} \end{cases} \tag{12}$$

La ortogonalidad es una condición esencial para que la señal OFDM sea libre de interferencia entre subportadoras.

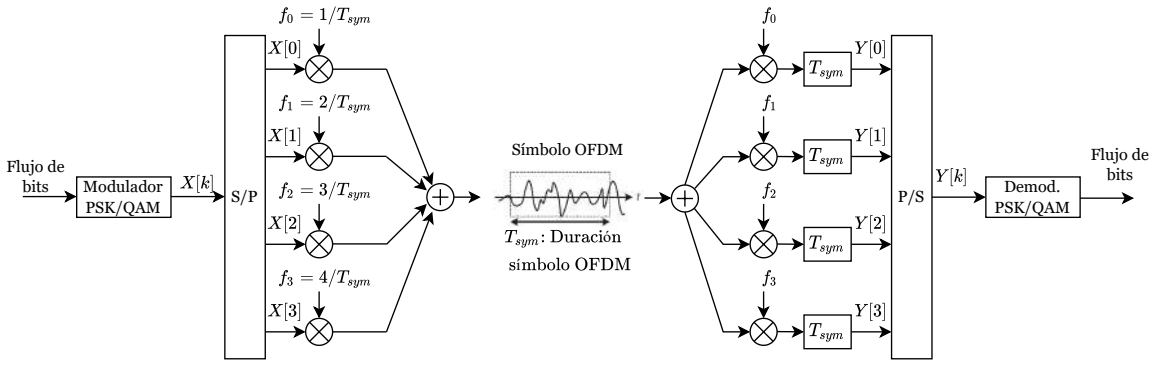
Modulación y demodulación OFDM

El transmisor OFDM modula una secuencia de bits en símbolos PSK o QAM, teniendo un flujo de símbolos complejos $X[k]$ el cual subsecuentemente es separado en N flujos paralelos. Cada uno de los N símbolos después de una conversión serial a paralelo les es asignado una subportadora de frecuencia $f_k = k/T_{sym}$. Los símbolos poseen una duración de T_s, así que la duración del tiempo de transmisión se extiende a $T_{sym} = NT_s$. Por tanto, los N símbolos a la salida del conversor serial a paralelo son los componentes discretos en frecuencia del modulador OFDM. En orden para generar símbolos OFDM, las componentes en frecuencia son convertidos en muestras temporales mediante la IFFT de N puntos. [3] [9]

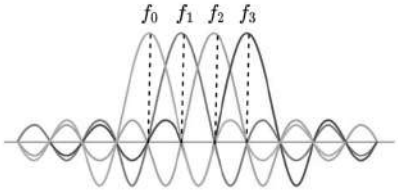
Una secuencia de símbolos OFDM en banda base está dado por la siguiente ecuación:

$$x[n] = \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N},\ 0 \leq n \leq N-1 \tag{13}$$

La modulación y demodulación pueden ser ilustradas por el diagrama de bloques en la figura 15a, la cual muestra que los símbolos en el dominio de la frecuencia $X[k]$ modulan una subportadora diferente. La señal OFDM puede ser recepcionada haciendo uso de la ortogonalidad entre las subportadoras, como se muestra en la figura 15b.



(a) Modulación y demodulación OFDM.



(b) Ortogonalidad entre subportadoras

Figura 15: Diagrama de bloques ilustrativo para una modulación y demodulación OFDM con 4 subportadoras. [9]

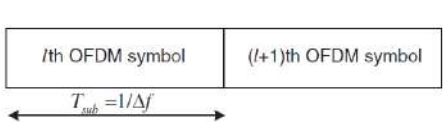
Intervalo de guarda y prefijo cíclico

Uno de los efectos en el canal inalámbrico es la interferencia entre símbolos (ISI) debido al multitrayecto del canal. El efecto de ISI por multitrayecto puede ser reducido en el símbolo OFDM extendiendo la duración del símbolo *N* veces. Sin embargo, el efecto aun permanece como un factor negativo que puede romper la ortogonalidad entre subportadoras. Por tanto un intervalo de guarda entre 2 símbolos OFDM consecutivos es esencial.

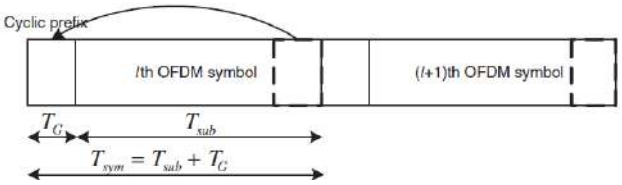
El intervalo de guarda puede ser insertado a manera de extensión cíclica del símbolo OFDM, conocido como prefijo cíclico (CP), el cual extiende el símbolo OFDM copiando las ultimas *G* muestras al inicio. En la figura 16b se muestran 2 símbolos OFDM con CP consecutivos donde T_G se denota como la longitud del CP en términos de muestras, T_{sub} es la duración de las subportadoras y $T_{sym} = T_G + T_{sub}$ es la duración del símbolo OFDM extendido.

La longitud del prefijo cíclico debe ser mayor al retardo de la respuesta del canal multitrayecto, así manteniendo la ortogonalidad entre subportadoras. Sin embargo el uso del CP significa supone una pérdida de energía al transmitir símbolos redundantes.

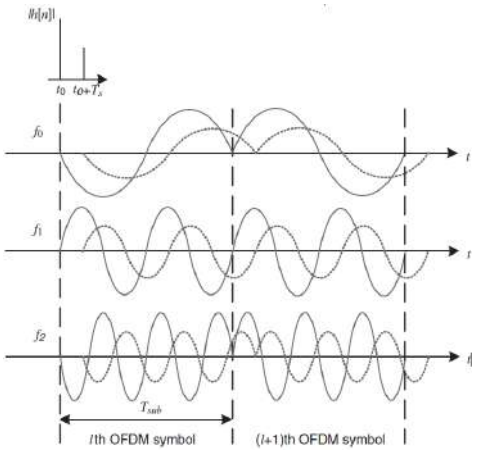
Finalmente se tiene un transmisor y receptor OFDM convencional como se muestra en la figura 17 que incluye las operaciones de IFFT y FFT.



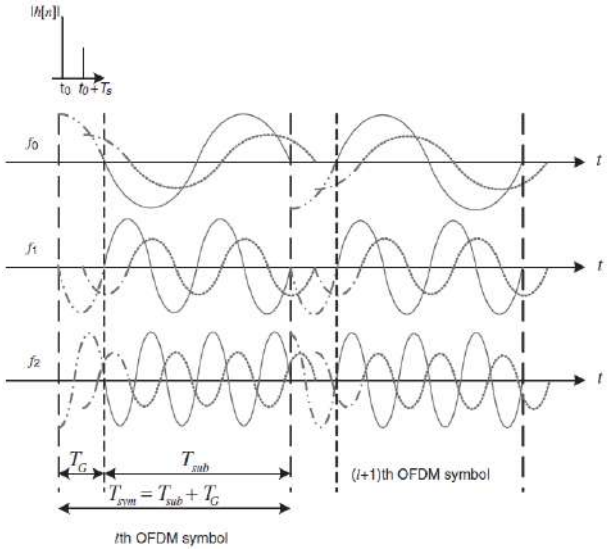
(a) Símbolos OFDM sin intervalo de guarda



(b) Adición del prefijo cíclico a símbolos OFDM.



(c) Efecto ISI en señal OFDM sin intervalo de guarda



(d) Efecto ISI en señal OFDM con prefijo cíclico

Figura 16: Comparación del efecto del multitrayecto del canal en señal OFDM sin intervalo de guarda y con prefijo cíclico. [9]

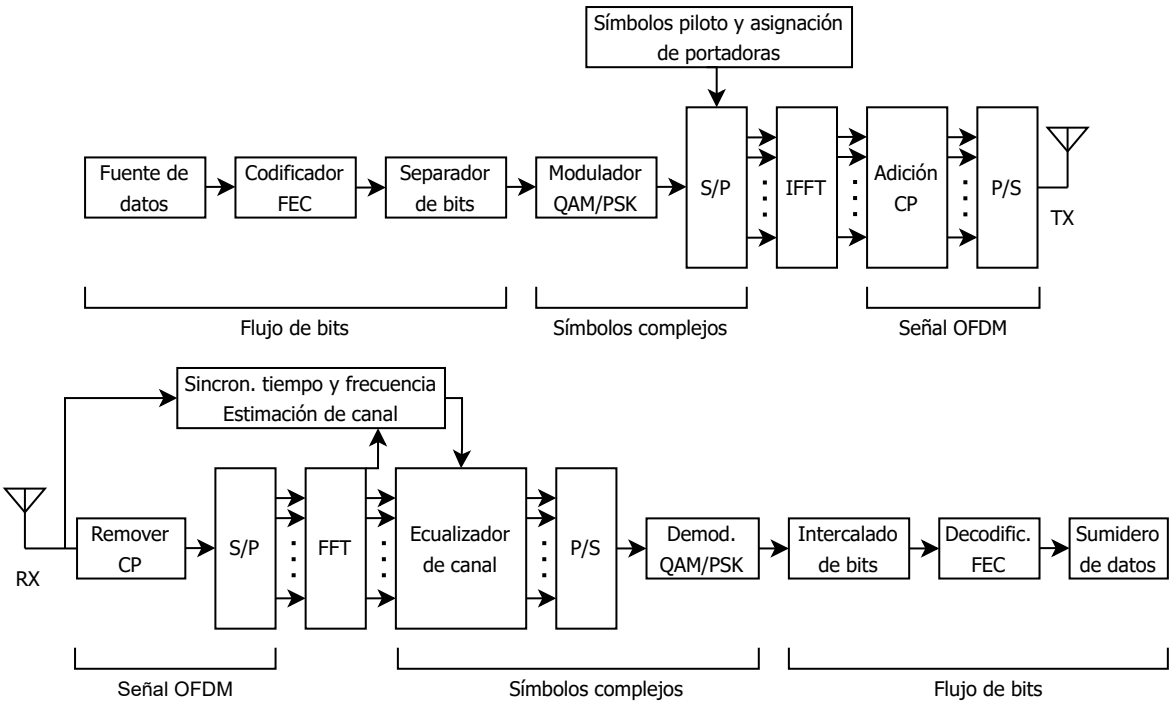


Figura 17: Diagrama de bloques de transmisión y recepción OFDM. [9]

2.6. MIMO-OFDM

MIMO-OFDM viene a ser el uso conjunto de las tecnologías de Múltiples antenas con la modulación OFDM, pues este tipo de modulación de por sí presenta una mejora sustancial en cuanto a tasas de transmisión por su esquema multiportadora, ahora resultando en la mejora de la capacidad de canal, cobertura y fiabilidad de datos.

2.6.1. Modelo de un sistema MIMO-OFDM

Consideramos un sistema MIMO-OFDM con M_t antenas transmisoras y M_r antenas receptoras. Cuando se aplica la técnica de multiplexación espacial la codificación puede ser aplicada en conjunto sobre las múltiples ramas de transmisión o en cada una.

En la parte de la transmisión se utilizan múltiples transmisores OFDM paralelos, los cuales tienen de entrada un subflujo de bits anteriormente multiplexado para cada rama, que luego se transmiten por sus respectivas antenas. En la parte de la recepción, los distintos flujos transmitidos tienen que pasar por una etapa de sincronización donde se distingue el inicio y fin de los paquetes OFDM, que luego se separan en subflujos donde se procesan paralelamente para demodular la información. Se utiliza una etapa de estimación de canal y detección para determinar la señal original de cada subflujo que luego se demultiplexa para recuperar la información en un solo flujo de datos.

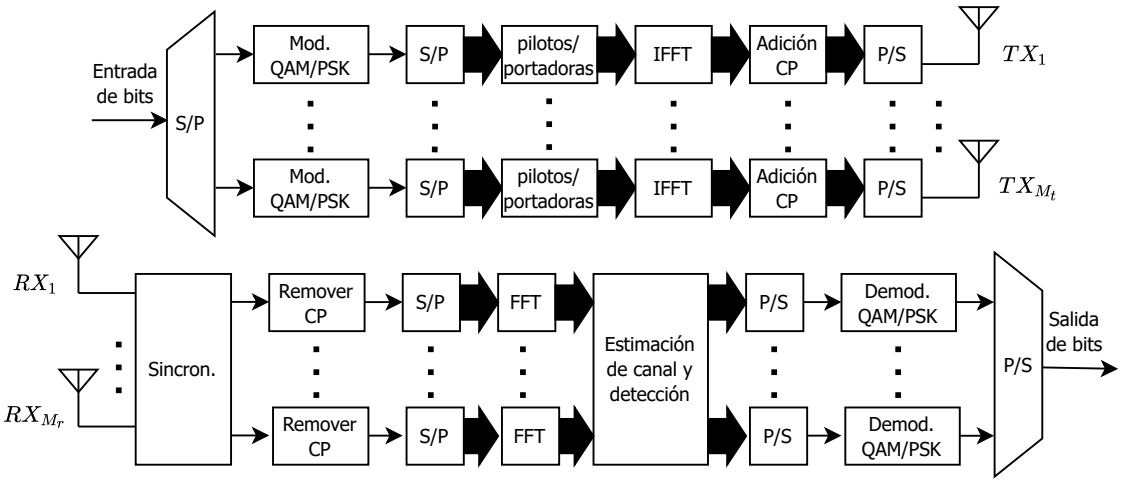


Figura 18: Diagrama de bloques general de transmisor y receptor MIMO-OFDM $M_r x M_t$. Elaboración propia.

2.6.2. Trama para MIMO-OFDM

El estándar IEEE 802.11n en comparación a los anteriores estándares 802.11a/g, incorpora transmisiones MIMO. Las señales OFDM son una buena opción para este tipo de comunicaciones debido a que las transmisiones para una subportadora generalmente son caracterizadas por una matriz compleja singular.

En la demodulación de una trama 802.11n se hace uso de un preámbulo de alto rendimiento (HT) que caracteriza los canales de transmisión. La salida de esta caracterización es la matriz del canal MIMO para cada subportadora. Esta matriz es usada para convertir múltiples señales recibidas (cada una puede tener interferencia de las demás flujos de datos transmitidos) en distintos flujos de datos. Para la demodulación de una señal MIMO, la matriz de canal MIMO debe ser invertida. Esta inversión puede ser realizada cuando los canales son lo suficientemente diferentes. [10]

802.11n ofrece dos características que incrementan el uso del espectro de radio: canales de 20MHz y 40MHz, a diferencia de los otros estándares mencionados, en 802.11n se tiene un mayor número de portadoras a utilizar, 56 portadoras (52 para datos y 4 de piloto). [11]

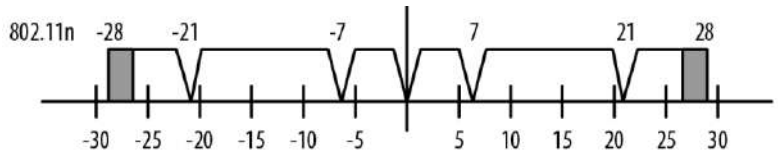


Figura 19: Canal de 20MHz 802.11n. [12]

El estándar 802.11n que hace uso de modulación OFDM para transmitir todos los datos, MIMO,define 3 modos de operación:

- **Non-HT o modo de legado:** Contiene 2 campos de preámbulo seguido por un campo de señal y campo de datos.
- **HT-mixed:** Este modo tiene contiene el primer formato de preámbulo de legado seguido de un campo de señal y múltiples campos de preámbulo HT y luego campos de datos.
- **HT-greenfield:** Este modo no utiliza el formato de legado, contiene un preámbulo HT corto seguido de un preámbulo HT largo, un campo de señal, múltiples campos de preámbulo HT y datos HT. La trama resultante es poco más pequeña que el modo HT

mixto, por lo que la transmisión en este modo es un poco más eficiente.

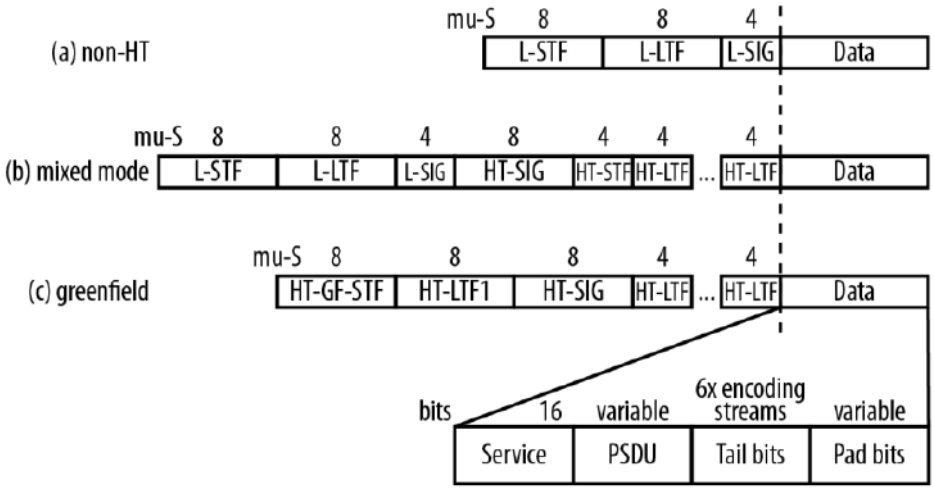


Figura 20: Formato de trama HT 802.11n. [12]

Campo	Descripción
L-STF	Non-HT Short Training Field
L-LTF	Non-HT Long Training Field
L-SIG	Non-HT Signal
HT-SIG	HT Signal
HT-STF	HT Short Training Field
HT-LTF	HT Long Training Field

Tabla 1: Descripción de campos de preámbulo. [12]

Estimación del canal

En un sistema MIMO-OFDM, el canal juega un rol importante para poder recuperar la información de manera confiable, por lo que necesita ser interpretado de la mejor manera posible. Para hacer esto, una comprensión exacta del canal puede ser adquirido mediante su estimación y mediante su detección, lo cual es seguido de una compensación de sus efectos en las señales recibidas. Uno de los métodos más conocidos es el uso de símbolos piloto, los cuales pueden aprovechar la ortogonalidad en el dominio de la frecuencia o utilizar el dominio temporal para tener una estimación del canal en una antena en el receptor. Pero la eficiencia espectral es reducida al tener que transmitir símbolos que no son parte de la información. [13]

Para estimar el canal haciendo uso de los símbolos piloto se utiliza el método de Least Square (LS), que no requiere mucha complejidad computacional para su implementación. Donde el

objetivo es minimizar la distancia cuadrada entre la señal recibida y la señal original. [14] [9]

LS estima el canal sobre las subportadoras piloto que puede ser obtenido mediante la ecuación.

$$\hat{H}_{LS} = (X_P)^{-1} Y_P \tag{14}$$

Donde $\hat{H}_{LS}$ es la matriz de canal estimada, X_P es la matriz de símbolos piloto conocidos y Y_P es la matriz de símbolos piloto que llegan al receptor.

Detección

Para la detección de la señal en el receptor se hace uso de la matriz de canal estimada H, el uso de una detección lineal trata a todas las señales transmitidas como interferencias excepto por la señal de interés proveniente del flujo de su antena. Por lo tanto, las señales de interferencia de otras antenas son minimizadas o anuladas en el proceso de detección. [9] Uno de los métodos de detección lineal que no requiere mucha complejidad computacional es la técnica de Zero-forcing (ZF), que se puede expresar como:

$$\tilde{\mathbf{x}}_{ZF} = \mathbf{H}^{-1}\mathbf{y} \tag{15}$$

Donde $\tilde{\mathbf{x}}_{ZF}$ representa un vector de $M_r \times 1$ cuyos elementos son los flujos de señales libres de interferencia e $\mathbf{y}$ es el vector $M_t \times 1$ en el que sus elementos son los flujos de señal entrantes en el receptor.

2.7. Radio Definida por Software

Las Radios Definidas por Software (SDR) son radios de propósito general con capacidad de soportar múltiples interfaces aire y protocolos inalámbricos a través del uso de antenas de banda ancha, conversiones de radiofrecuencia, conversores análogo-digital (ADC), digital-análogo (DAC). Se caracterizan por ser reconfigurables ya que su funcionalidad se encuentra definida por software, esto quiere decir que puede implementar diferentes aplicaciones utilizando el mismo hardware. Las principales limitantes de estas radios se deben a distintos factores que llevan desde el consumo energético o las elevadas especificaciones del procesador. [15]

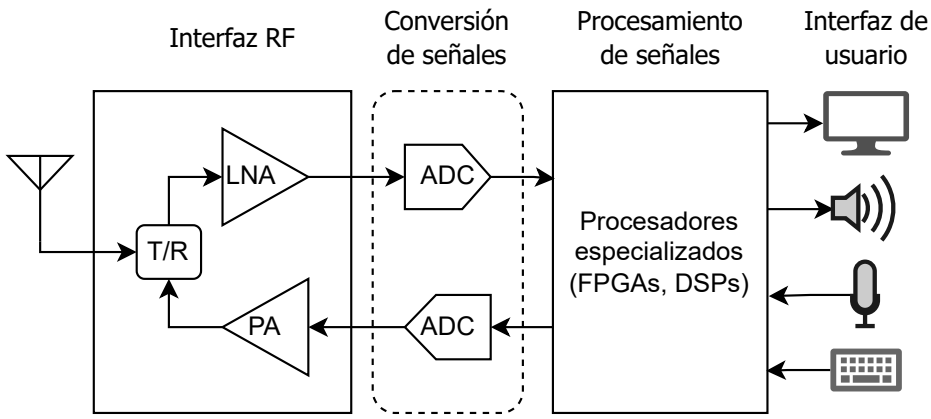


Figura 21: Diagrama de Radio definida por software. Elaboración propia.

2.7.1. Hardware USRP

Los USRP (Universal Software Radio Peripheral) proporcionan una arquitectura de RF definida por software para diseñar, crear prototipos e implementar rápidamente sistemas inalámbricos con procesamiento de señales personalizado. [16]

NI USRP 2920

El NI USRP 2920 o USRP N210 es usado para aplicaciones de comunicaciones demandantes que requieren desarrollo rápido. Su arquitectura incluye un FPGA Xilinx Spartan 3A DSP 3400, puede operar en el rango de frecuencias de 50 MHz a 2.2 GHz, posee un ancho de banda en banda base de 40 MHz tanto en transmisión como recepción, y una conectividad mediante interfaz Gigabit Etheret para el flujo de datos con ordenadores. Su diseño modular incluye un puerto de expansión que permite multiple USRP de la misma serie ser sincronizados y usarlos en configuración MIMO. [16]

Figura 22: NI USRP 2920. [16]

Las especificaciones técnicas del NI USRP 2920 son:

Transmisor	
Rango de frecuencia	50 MHz a 2.2 GHz
Paso de frecuencia	<1 KHz
Máxima potencia de salida	
50 MHz a 1.2 GHZ 1.2 GHZ a 2.2 GHz	50mW a 100mW (17 dBm a 20 dBm) 30mW a 70mW (15 dBm a 18 dBm)
Rango de ganancia	0 a 31 dB
Paso de ganancia	1.0 dB
Precisión de frecuencia	2.5 ppm
Máximo ancho de banda instantánea en tiempo real	
Muestras de 16 bits Muestras de 8 bits	20 MHz 40 MHz
Máxima tasa de muestreo I/Q	
Muestras de 16 bits Muestras de 8 bits	25 MS/s 50 MS/s
Conversor digital-análogo (DAC)	2 Canales, 400MS/s, 16 bit
DAC rango dinámico libre de espurios (sFDR)	80 dB

Tabla 2: Especificaciones del transmisor de NI USRP 2920

Receptor	
Rango de frecuencia	50 MHz a 2.2 GHz
Paso de frecuencia	<1kHz
Rango de ganancia	0 a 31.5 dB
Paso de ganancia	0.5 dB
Máxima potencia de entrada	0 dBm
Figura de ruido	5 dB a 7 dB
Precisión de frecuencia	2.5 ppm
Máximo ancho de banda instantánea en tiempo real	
Muestras de 16 bits Muestras de 8 bits	20 MHz 40 MHz
Máxima tasa de muestreo I/Q	
Muestras de 16 bits Muestras de 8 bits	25 MS/s 50 MS/s
Conversor análogo-digital (ADC)	2 Canales, 400MS/s, 14 bit
ADC rango dinámico libre de espurios (sFDR)	88 dB

Tabla 3: Especificaciones del receptor de NI USRP 2920

NI USRP 2950R

El NI USRP 2950R o USRP X310 es una plataforma de Radio definida por software escalable de alto rendimiento para diseño y despliegue de sistemas inalámbricos de siguiente generación. La arquitectura de hardware combina 2 tarjetas madre de ancho de banda ex-

tendido con un rango de frecuencia de 50 Mhz a 2.2 GHz y un ancho de banda en banda base de 120 MHz, posee múltiples interfaces de conexión de alta velocidad (PCIe, 10 GigE dual y 1 GigE dual) y un FPGA Kintex-7 410T programable. En adición para proveer el mejor rendimiento en su clase, su arquitectura de software de código abierto provee soporte del controlador UHD multiplataforma haciéndolo compatible con un gran número de dispositivos, espacios de trabajo y proyectos de código abierto. [16]

Figura 23: NI USRP 2950R. [16]

Sus especificaciones del NI USRP 2950R son:

Transmisor	
Número de canales	2
Rango de frecuencia	50 MHz a 2.2 GHz
Paso de frecuencia	<1kHz
Máxima potencia de salida	
50 MHz a 1.2 GHz 1.2 GHZ a 2.2 GHz	50 mW a 100 mW (17 dBm a 20 dBm) 30 mW a 70 mW (15 dBm a 18 dBm)
Rango de ganancia	0 a 31 dB
Paso de ganancia	1.0 dB
Máximo ancho de banda instantánea en tiempo real	120MHz
Máxima tasa de muestreo I/Q	200 MS/s
Conversor digital-análogo (DAC)	
Resolución sFDR	16 bit 80 dB

Tabla 4: Especificaciones del transmisor de NI USRP 2950R

Receptor	
Número de canales	2
Rango de frecuencia	50 MHz a 2.2 GHz
Paso de frecuencia	$<$1kHz
Rango de ganancia	0 a 37.5 dB
Paso de ganancia	0.5 dB
Máxima potencia de entrada	-15 dBm
Figura de ruido	5 dB a 7 dB
Máximo ancho de banda instantánea en tiempo real	120MHz
Máxima tasa de muestreo I/Q	200 MS/s
Conversor análogo-digital (ADC)	
Resolución sFDR	14 bit 88 dB

Tabla 5: Especificaciones del receptor de NI USRP 2950R

2.8. Ubuntu

Ubuntu es una distribucion Linux de código abierto y libre, tiene gran compatibilidad con la mayoría de aplicaciones desarrolladas para Linux y presenta un entorno amigable para el usuario, además presenta una comunidad grande entre desarrolladores y usuarios que dan soporte constante al sistema operativo. [17]

La versión de Ubuntu utilizada es 20.04 LTS de 64 bits funcionando sobre una máquina Workstation Z4 G4. Tiene habilitada 2 interfaces de conexión: 1Gb Ethernet (2 puertos) y PCI (4 puertos).

Figura 24: Logo de Ubuntu

2.9. GNU Radio

GNU Radio es una plataforma de desarrollo en software con licencia GNU (Licencia pública general) que provee bloques de procesamiento de señal para implementar Software Radios, que puede ser utilizado en conjunto con equipo RF externo para hacer Radios Definidas por Software, o utilizarlo como un entorno de simulación sin uso de hardware. Es ampliamente

usado en la industria y aprendizaje académico para dar soporte a investigaciones en comunicaciones inalámbricas y sistemas de radio aplicables en el mundo real. [18] La versión de GNU Radio utilizada para este trabajo es 3.8.1.

Figura 25: Logo de GNU Radio

GNU Radio utiliza 2 lenguajes de programación para su funcionamiento, el primero es Python, el cual se provee una interfaz gráfica orientada al uso de diagramas de bloques para trabajar, y el segundo es C++ que es el lenguaje que utilizan los bloques para procesar las señales, esto debido al rendimiento del lenguaje C sobre Python. La interacción entre ambos lenguajes se debe a la librería de Pybind11 que permite utilizar funciones de C++ en Python y viceversa.

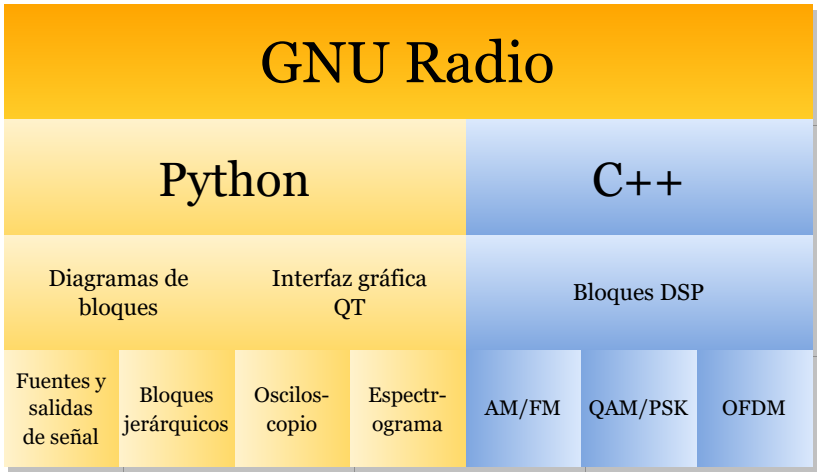


Figura 26: Arquitectura de GNU Radio. Elaboración propia.

Entre las principales ventajas del uso de GNU Radio con los SDR se tiene que:

- A diferencia de Simulink, GNU Radio permite trabajar paralelamente con varios conjuntos de muestras digitales en todos sus bloques, lo que representa un mejor rendimiento al momento de usar los SDR.
- GNU Radio permite añadir nuestras propias funciones y bloques en lenguaje Python

o C++ al código fuente o utilizar un bloque personalizado que permite usar e implementar bloques no tan complejos.

Y las desventajas que se tiene en GNU Radio son:

- GNU Radio no posee bloques para todas las aplicaciones de Telecomunicaciones ya que existen diversos enfoques y métodos para implementar cada uno, por lo que es más efectivo hacer uso de alguna librería de bloques externa ya desarrollada o sino desarrollar una nueva.
- Varios bloques quedan obsoletos debido a que no tienen la programación adecuada de acuerdo al enfoque que se les da y terminan siendo modificados o reemplazados en nuevas actualizaciones.

La guía de uso de GNU Radio Companion se puede encontrar en el Anexo E.

CAPITULO III

3. Diseño e implementación del sistema de comunicación en plataforma SDR

En este capítulo se describe el diseño e implementación del sistema de comunicación MIMO-OFDM 2x2 haciendo uso del algoritmo de *water-filling* para asignación de potencia óptima, utilizando GNU Radio para la parte en software y los dispositivos USRP para la parte en hardware como se muestra en el diagrama de la figura 27.

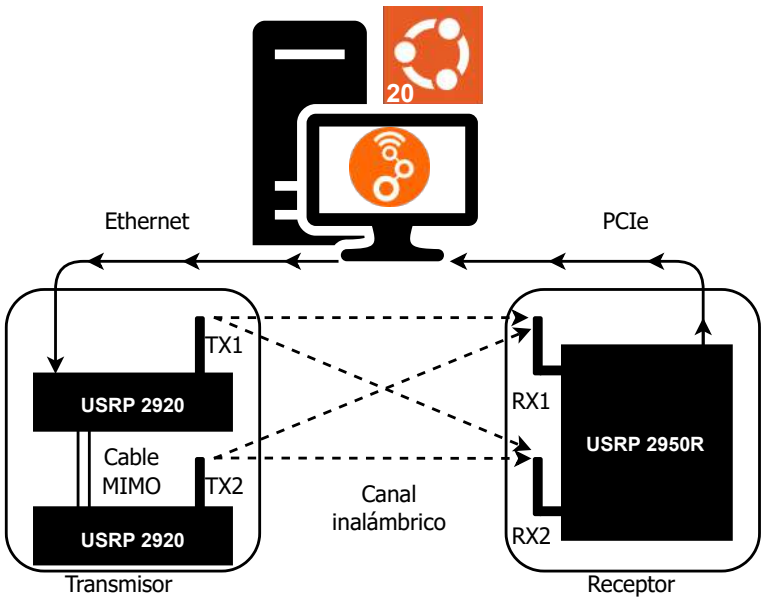


Figura 27: Diagrama de sistema de comunicación OFDM 2x2 con USRP. Elaboración propia

3.1. Transmisor MIMO-OFDM 2x2

3.1.1. Diseño de sistema de transmisión

A continuación se muestra un diagrama de bloques del transmisor OFDM 2x2 en la figura 28.

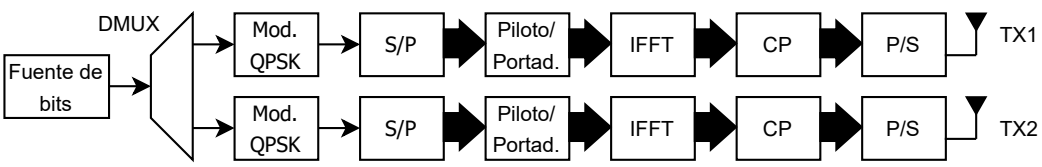


Figura 28: Diagrama de bloques del transmisor OFDM 2x2. Elaboración propia.

Se tiene una fuente de bits que es demultiplexada en 2 flujos diferentes, que posteriormente son modulados en símbolos QPSK y convertidos en subflujos paralelos por medio de conversores serial a paralelo. Después se les asignan portadoras virtuales e introducen símbolos piloto, luego se efectúa la IFFT para convertir al plano temporal y se adiciona un prefijo cíclico a cada flujo. Finalmente se tienen señales OFDM en banda base para trasmitir utilizando una interfaz RF.

3.1.2. Implementación del sistema de transmisión MIMO-OFDM 2x2

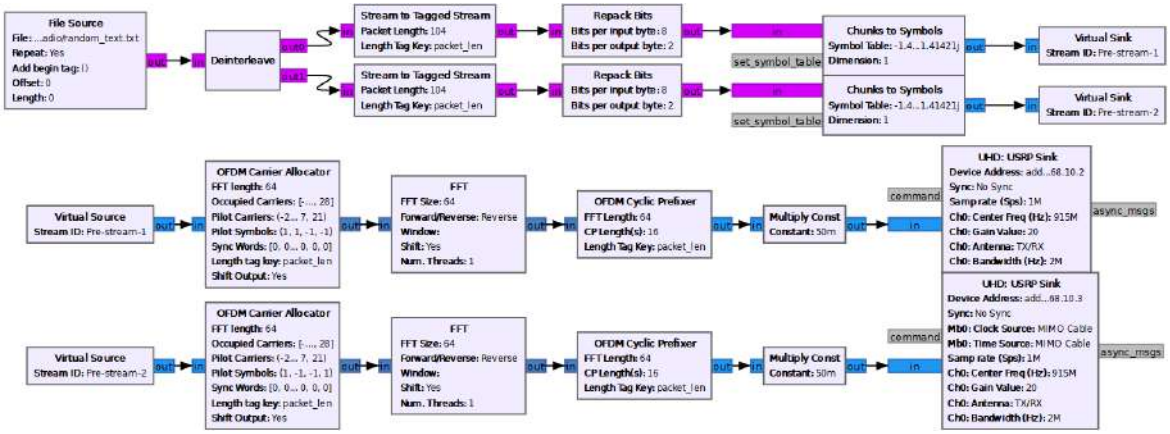


Figura 29: Transmisor OFDM 2x2 en GNU Radio.

Fuente de información y flujo de bits

Primero para tener un flujo bits en GNU Radio se utiliza un archivo digital, como un mensaje de texto o una imagen. Para utilizaro a modo de fuente de bits se utiliza el bloque *File source* y se configura el parámetro *Output type* para que a la salida se tengan Bytes o paquetes de 8 bits.

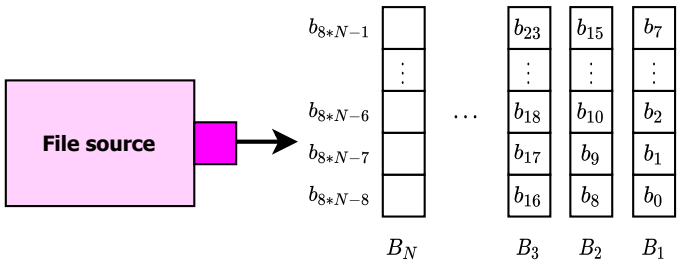


Figura 30: Bloque *File source* para un archivo de N bytes

Para generar los subflujos hacia las antenas se hace uso del bloque *Deinterleave*, que funciona como un demultiplexor de bytes como se muestra en la figura 31, donde se modifica

el parámetro *block size* para demultiplexar en grupos de 2 bytes.

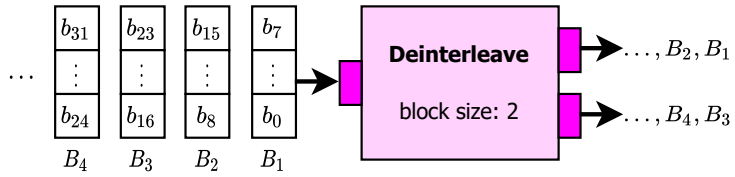


Figura 31: Bloque *Deinterleave*

Ahora se define el tamaño de la trama de información. Para ello se utilizó el bloque *Stream to Tagged Stream*, cuya función es etiquetar el flujo de bytes con un valor y un nombre identificador o Tag del cual los siguiente bloques harán uso. De acuerdo al estándar 802.11n, con 52 subportadoras de información, cada una con 1 símbolo QPSK con 2 bits de información, se tendrían 13 Bytes, y para 8 símbolos OFDM se tiene 104 Bytes de datos y se etiqueta con el Tag: "`packet_len`".

Figura 32: Bloque *Stream to Tagged Stream*

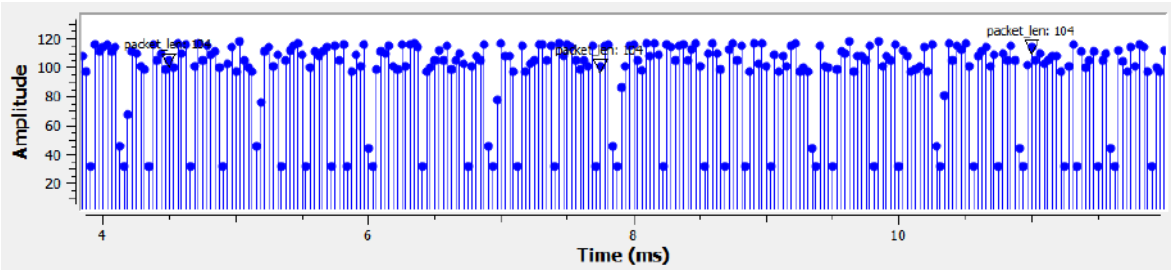


Figura 33: Señal discreta de un flujo de bytes en sistema decimal etiquetada cada 104 bytes

Modulación QPSK

Para modular los bits en símbolos QPSK se define el tipo de modulación por medio de un bloque *Variable*, que tiene como identificador "`pld_mod`". Además, se hace uso de la librería *digital* de GNU Radio que contiene funciones para crear objetos de modulación PSK o QAM, en este caso se utiliza la función "`digital.constellation_qpsk()`" como se muestra en la figura 34.

Después se continúa con el procesamiento de los bits, para lo cual ante una modulación QPSK, se requiere una entrada de 2 bits, siendo que se tiene de entrada del tipo Byte. Para

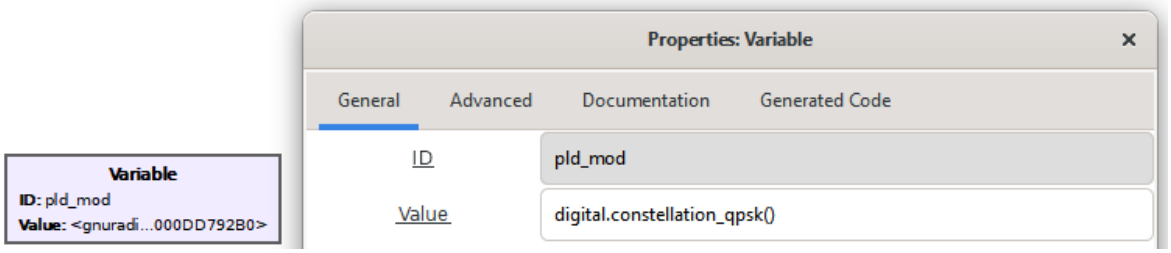


Figura 34: Definición de modulador QPSK

ello se utilizará el bloque *Repack bits*, el cual se configura para una entrada de 8 bits por byte y una salida de 2 bits por byte, y también hace uso del Tag "`packet_len`", la función del bloque se muestra en la figura 35.

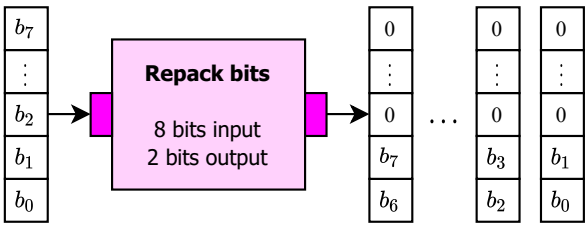


Figura 35: Bloque *Repack bits*

Para el mapeado de símbolos se utiliza el bloque *Chunks to symbols*, que se configura para una entrada del tipo Byte y de salida del tipo compleja, la tabla de símbolos utiliza los puntos de la modulación mediante la función "`payload_mod.points()`" como se muestra en la figura 36.

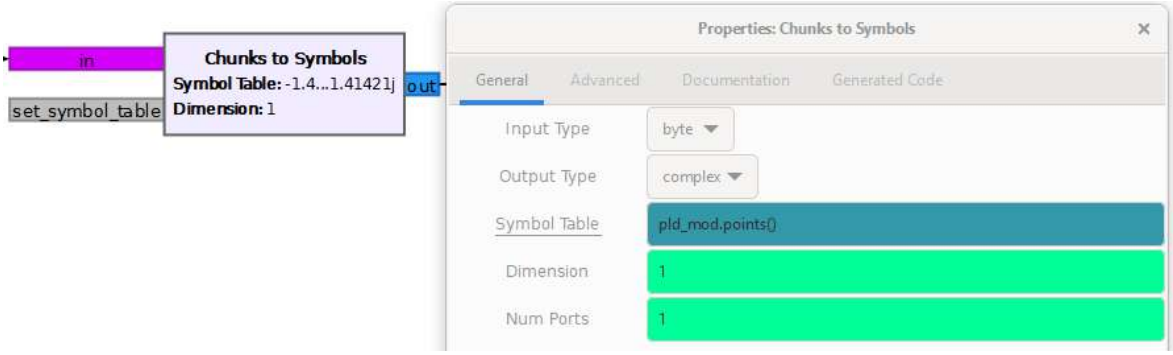


Figura 36: Bloque *Chunks to symbols*

En la figura 37 se ve una ventana de constelación con símbolos QPSK a la salida del bloque *Chunks to symbols*, con valores $(1.4142+1.4142j);(-1.4142+1.4142j);(-1.4142-1.4142j);(1.4142-1.4142j)$ cuyo módulo es 2.

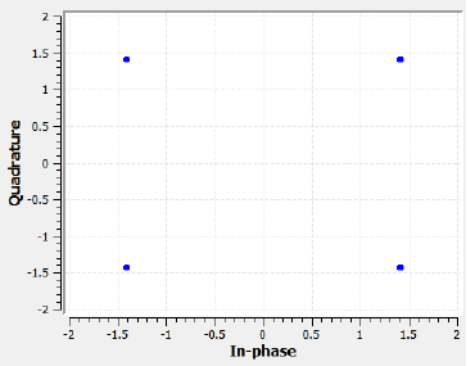


Figura 37: Símbolos QPSK en GRC

Modulación OFDM

Una vez se haya modulado los bits de información, siguen las etapas de conversor serial a paralelo para generar flujos paralelos de símbolos QPSK y la asignación de portadoras virtuales a cada uno de estos flujos. Para estas 2 etapas se utiliza el bloque *OFDM Carrier Allocator*, que se muestra en la figura 38.

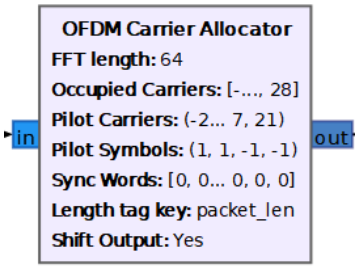


Figura 38: Bloque *OFDM Carrier Allocator*

Este bloque tiene de entrada tiene un flujo de símbolos complejos, los cuales son distribuidos en tiempo y frecuencia, así también se insertan símbolos piloto y símbolos de sincronismo para formar la trama de información.

La asignación de subportadoras de acuerdo al estándar 802.11n [12] establece 64 subportadoras:

- Componente continua (1 portadora central): 0.
- Datos (52 portadoras): (-28 a -22), (-20 a -8), (-6 a -1), (1 a 6), (8 a 20), (22 a 28). [12]
- Piloto (4 portadoras): (-21,-7,7,21). [12]
- Sin usar (7 portadoras): (-32 a -29), (29 a 31).

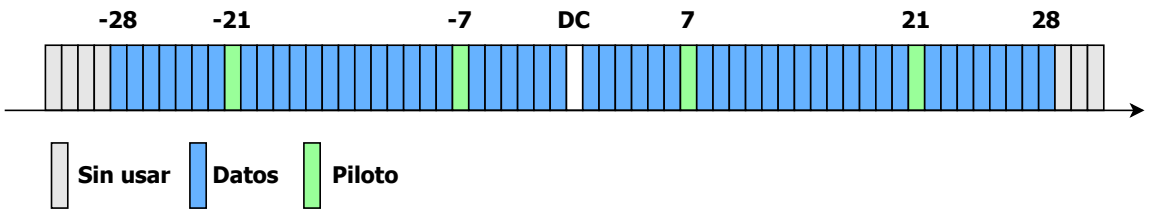


Figura 39: Asignación de subportadoras. Elaboración propia

Los símbolos piloto vienen a ser los símbolos que se encuentran dentro de cada uno de los símbolos OFDM a transmitir, estos deben ser ortogonales entre sí. Para un sistema MIMO-OFDM 2x2 los símbolos piloto utilizados son símbolos BPSK $[1,1,-1,-1]$, $[1,-1,-1,1]$ para el flujo 1 y 2 respectivamente, desarrollado en [19].

En el formato del preámbulo se forma una trama *HT_mixed* desarrollada en [19] que utiliza campos HT-LTF de símbolos de entrenamiento con el cual el receptor será capaz de estimar el canal MIMO. Los campos HT-LTF para cada flujo estarán compuestos por 2 símbolos HT-DLTF como se ve en la figura 40, que son símbolos utilizados para la demodulación de los datos.

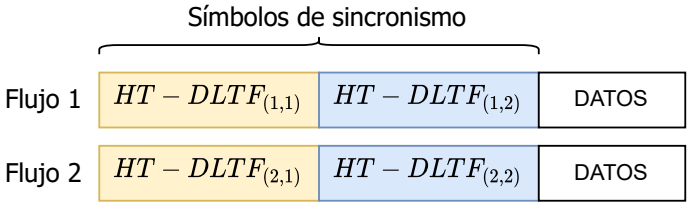


Figura 40: Formato de preámbulos. [19]

Los símbolos de sincronismo están constituidos por 57 elementos que componen las subportadoras de datos, subportadoras piloto y la portadora central, mientras que en GNU Radio se completa con ceros para tener 64 elementos, de acuerdo a la asignación se tiene 7 subportadoras sin usar, 4 al inicio y 3 al final.

Los símbolos de sincronismo *HT-DLTF* para el primer flujo son:

$$HT\text{-}LTF_{(1,1)} = [1,1,1,1,\text{-}1,\text{-}1,1,1,\text{-}1,1,\text{-}1,1,1,1,1,1,1,1,\text{-}1,\text{-}1,1,1,\text{-}1,1,\text{-}1,1,1,$$
$$1,1,0,1,\text{-}1,\text{-}1,1,1,\text{-}1,1,1,1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,1,1,\text{-}1,\text{-}1,1,\text{-}1,1,\text{-}1,1,1,1,1,\text{-}1,\text{-}1] \quad (16)$$

$$HT\text{-}LTF_{(1,2)} = [\text{-}1,\text{-}1,\text{-}1,\text{-}1,1,1,\text{-}1,\text{-}1,1,\text{-}1,1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,1,1,\text{-}1,\text{-}1,1,\text{-}1,1,\text{-}1,$$
$$\text{-}1,\text{-}1,\text{-}1,0,\text{-}1,1,1,\text{-}1,\text{-}1,1,\text{-}1,1,\text{-}1,1,1,1,1,1,\text{-}1,\text{-}1,1,1,\text{-}1,1,\text{-}1,1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,1,1] \quad (17)$$

Los símbolos de sincronismo *HT-DLTF* para el segundo flujo son:

$$HT\text{-}LTF_{(2,1)} = [\text{-}1,1,\text{-}1,1,1,1,1,1,1,\text{-}1,\text{-}1,1,1,\text{-}1,1,\text{-}1,1,1,1,1,1,\text{-}1,\text{-}1,1,1,\text{-}1,$$
$$1,\text{-}1,0,1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,1,1,\text{-}1,\text{-}1,1,\text{-}1,1,\text{-}1,1,1,1,1,\text{-}1,\text{-}1,1,1,1,1,\text{-}1,\text{-}1,1,1] \quad (18)$$

$$HT\text{-}LTF_{(2,2)} = [\text{-}1,1,\text{-}1,1,1,1,1,1,1,\text{-}1,\text{-}1,1,1,\text{-}1,1,\text{-}1,1,1,1,1,1,\text{-}1,\text{-}1,1,1,\text{-}1,$$
$$1,\text{-}1,0,1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,\text{-}1,1,1,\text{-}1,\text{-}1,1,\text{-}1,1,\text{-}1,1,1,1,1,\text{-}1,\text{-}1,1,1,1,1,\text{-}1,\text{-}1,1,1] \quad (19)$$

En la figura 41 se visualiza la salida del bloque *OFDM Carrier Allocator* en una ventana de instrumentación como si fuera un único flujo, obteniendo una trama de 640 símbolos entre BPSK y QPSK. Los primeros 128 símbolos BPSK son de sincronismo y los 512 símbolos QPSK restantes son de datos, y vuelve a repetirse el formato de la trama.

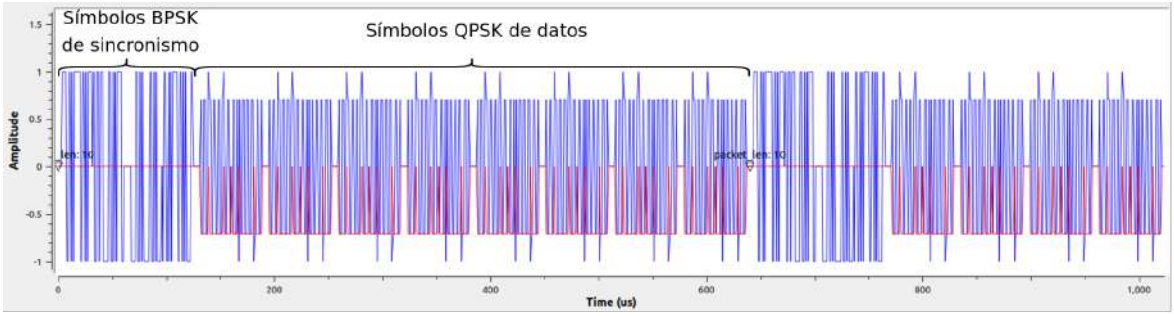


Figura 41: Símbolos de salida del bloque *OFDM Carrier Allocator*

La siguiente etapa consiste en la operación de la IFFT para transformar los 64 flujos paralelos del dominio discreto de la frecuencia al dominio discreto temporal. Con lo cual se utiliza el bloque *FFT*, que es configurado en el modo *Reverse* para aplicar la transformada inversa como se muestra en la figura 42.

Figura 42: Bloque *FFT* en modo *Reverse*

Finalmente para adicionar el prefijo cíclico se utiliza el bloque *OFDM Cyclic Prefixer* que es mostrado en la figura 43, donde de acuerdo al estándar 802.11n el prefijo cíclico consta de la cuarta parte de las subportadoras totales.

En la figura 44 se muestra primero la señal OFDM resultante de aplicar la IFFT, obteniendo

10 símbolos en total, 2 de sincronización y 8 de datos con un total de 640 muestras. Después, al adicionar el prefijo cíclico el número de muestras aumenta de 640 a 800.

Figura 43: Bloque *OFDM Cyclic Prefixer*

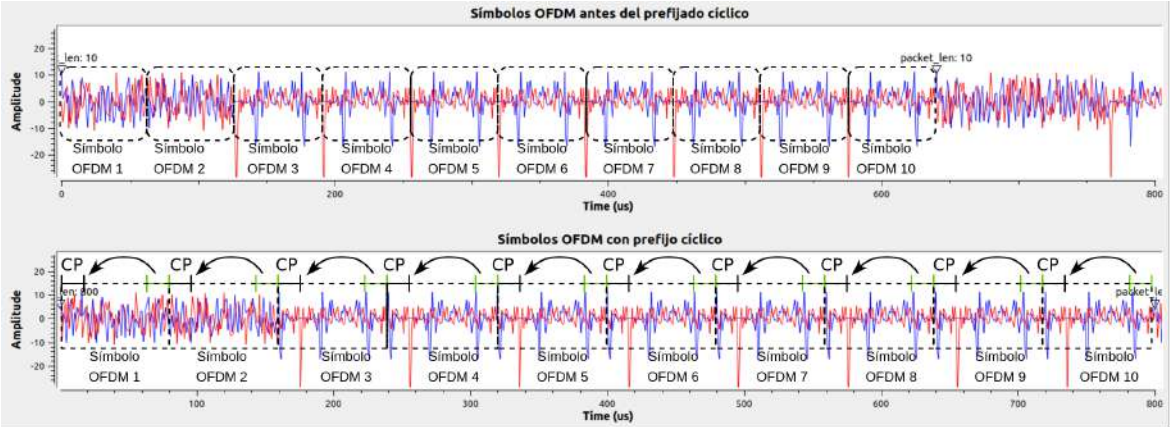


Figura 44: Extensión de símbolos OFDM con el prefijo cíclico

Con esto se concluye la implementación del sistema transmisor MIMO-OFDM 2x2 que está listo para ser utilizado en simulación y posteriormente implementar el enlace de la comunicación inalámbrica mediante hardware SDR.

3.2. Receptor MIMO-OFDM 2x2

3.2.1. Diseño de sistema de recepción

A continuación se muestra un diagrama de bloques del receptor OFDM 2x2.

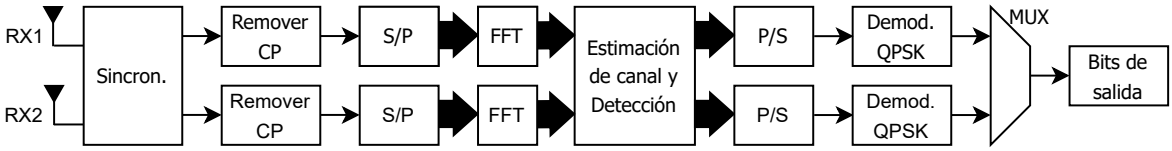


Figura 45: Diagrama de bloques del receptor OFDM 2x2. Elaboración propia.

Primero comienza con la recepción de las señales OFDM que pasan por una etapa de sincronización de ambos flujos y luego se procede a remover el prefijo cíclico de cada una. Después, pasan por un conversor serial a paralelo para aplicar la FFT, que transforma los flujos del dominio del tiempo al dominio de la frecuencia, obteniendo flujos de símbolos

QPSK. Para ecualizar el canal se pasa por una etapa de estimación del canal y detección que mitiga las interferencias de los símbolos recibidos, finalmente son demodulados y se utiliza un multiplexor que permita reconstruir la información recibida.

Implementación del sistema de recepción MIMO-OFDM 2x2

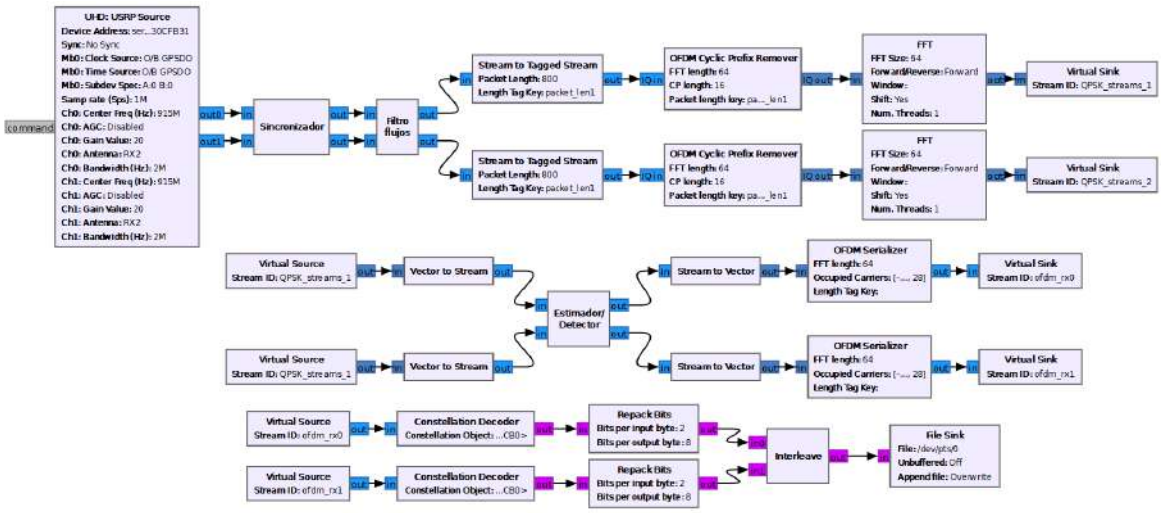


Figura 46: Receptor MIMO-OFDM 2x2 en GNU Radio

Sincronización

Para la etapa de sincronización se utilizan 2 bloques que fueron desarrollados en [19], los cuales basan en la técnica de ventanas deslizantes [9]. De acuerdo a la técnica mencionada, para detectar el inicio y final de los símbolos OFDM, se toma una ventana de muestras y utilizando el prefijo cíclico se procede a efectuar la correlación de señales desde el inicio de un símbolo OFDM hasta detectar una copia que indicaría el final del mismo. Ya que ambos flujos no llegan en el mismo instante de tiempo por el retardo de trayecto múltiple, habiendo utilizado el procedimiento anterior, los paquetes son sincronizados en un mismo tramo temporal para ser procesados.

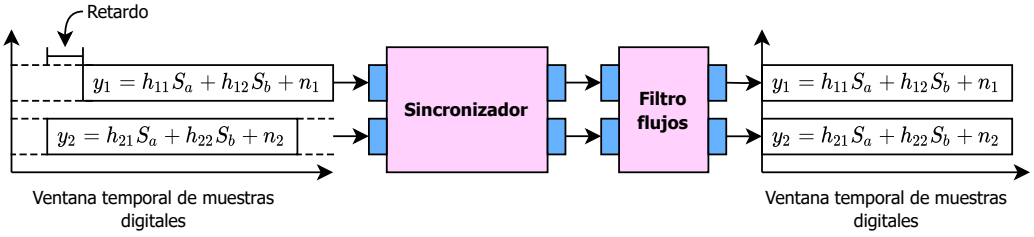


Figura 47: Bloques *Sincronizador* y *Filtro flujos*. [19]

Una vez sincronizado los flujos se procede a etiquetar la señal con un Tag, que marca las muestras de interés a procesar. Para este caso como los 10 símbolos OFDM constan de 800 elementos con las que se procesaron en la etapa de transmisor, se utiliza el bloque *Stream to Tagged Stream* para señales complejas y etiquetar con el Tag "`packet_len1`".

Figura 48: Bloque *Stream to Tagged Stream* para señales complejas

La siguiente etapa para los flujos sincronizados es remover el prefijo cíclico, para esto se hace uso del bloque *OFDM Cyclic Prefix Remover* que se muestra en la figura 49, este bloque pertenece a la librería externa *gr-radar* [20]. Tiene de parámetros el número de subportadoras, la longitud del prefijo cíclico y la etiqueta de señal Tag, a la salida actúa como un conversor serial a paralelo.

Figura 49: Bloque *Cyclic Prefix Remover*

En la figura 50 se puede ver la comparación de la señal OFDM recepcionada y como después se le es removido el prefijo cíclico, reduciendo el número de muestras.

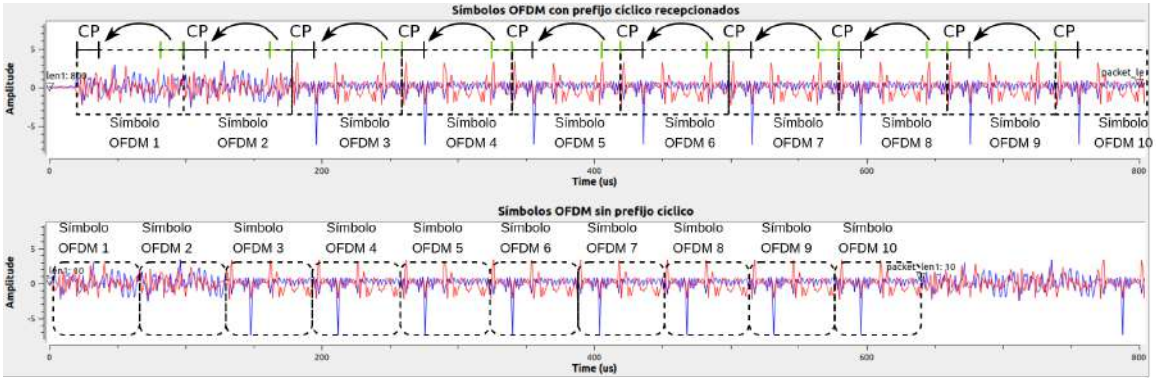


Figura 50: Extracción del prefijo cíclico de los símbolos OFDM recepcionados

Después de remover el prefijo cíclico en cada símbolo OFDM, se retorna al procesamiento de 640 elementos. Para convertir la señal OFDM del plano discreto temporal al plano discreto de la frecuencia se utiliza el bloque *FFT* pero configurado en modo *Forward* tal como se muestra en la figura 51.

Figura 51: Bloque *FFT* en modo Forward

Mediante este proceso se vuelven a tener 64 flujos paralelos de símbolos BPSK y QPSK afectados por el canal como se muestra en la figura 52.

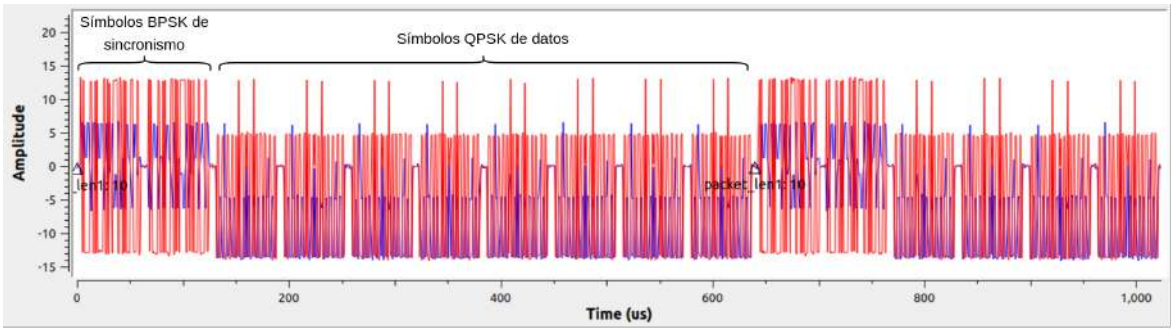


Figura 52: Señal de salida después de efectuar la FFT en la señal recepcionada

Estimación de canal y detección

Para el la estimación y detección se hace uso del bloque *Estimador/Detector*, desarrollado en [19], que es un bloque embebido de 2 entradas y salida de tipo compleja que basa su funcionamiento en el algoritmo *Least Square* (LS) y *Zero forcing*(ZF) de las ecuaciones 14 y 15 respectivamente. Este bloque se encarga de estimar el canal MIMO **H** caracterizado como una matriz compleja 2x2 mediante el uso vectores de símbolos de entrenamiento.

Los símbolos de entrenamiento utilizados para los flujos 1 y 2 se muestra en el siguiente segmento de código:

```
1 self.p1 = numpy.array([0, 0, 0, 0, 1, 1, 1, 1,-1,-1, 1, 1,-1, 1,-1,
    1, 1, 1, 1, 1, 1,-1,-1, 1, 1,-1, 1,-1, 1, 1, 1, 1, 0, 1,-1,-1, 1,
    1,-1, 1,-1, 1,-1,-1,-1,-1,-1, 1, 1,-1,-1, 1,-1, 1,-1, 1, 1, 1,
    1,-1,-1, 0, 0,  0, 0, 0, 0, 0,-1,-1,-1,-1, 1, 1,-1,-1, 1,-1,
    1,-1,-1,-1,-1,-1,-1, 1, 1,-1,-1, 1,-1, 1,-1,-1,-1,-1, 0,-1, 1,
    1,-1,-1, 1,-1, 1,-1, 1, 1, 1, 1, 1,-1,-1, 1, 1,-1, 1,-1,
    1,-1,-1,-1,-1, 1, 1, 0, 0, 0])
2 self.p2 = numpy.array([0, 0, 0, 0,-1, 1,-1, 1, 1, 1, 1, 1, 1,-1,-1,
    1, 1,-1, 1,-1, 1, 1, 1, 1, 1,-1,-1, 1, 1,-1, 1,-1, 0,
    1,-1,-1,-1,-1,-1, 1, 1,-1,-1, 1,-1, 1,-1, 1, 1, 1, 1,-1,-1, 1, 1,
    1, 1,-1,-1, 1, 1, 0, 0, 0, 0, 0, 0, 0,-1, 1,-1, 1, 1, 1, 1, 1,
    1,-1,-1, 1, 1,-1, 1,-1, 1, 1, 1, 1, 1,-1,-1, 1, 1,-1, 1,-1, 0,
    1,-1,-1,-1,-1,-1, 1, 1,-1,-1, 1,-1, 1,-1, 1, 1, 1, 1,-1,-1, 1, 1,
    1, 1,-1,-1, 1, 1, 0, 0, 0])
```

Segmento de código 1: Definición de los símbolos de entrenamiento

GNU Radio al trabajar con flujos paralelos utiliza arreglos anidados que contienen vectores de los 64 flujos paralelos por lo que se utiliza el bloque *Vector to Stream* para convertir el vector de muestras paralelas a un solo flujo, y también después de procesar la estimación de canal y su ecualización para los símbolos, para regresarlo a un procesamiento paralelo se utiliza el bloque *Stream to Vector* como se muestra en la figura .

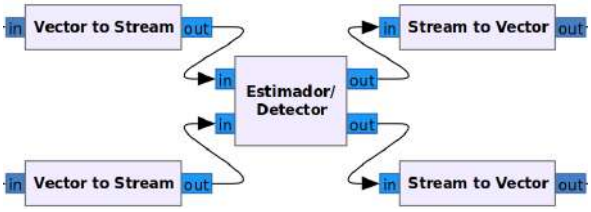


Figura 53: Bloque *Estimdor/Detector*. [19]

Una vez eliminada la interferencia del canal se procede a extraer las subportadoras utilizando el bloque *OFDM Serializer*, que realiza la operación inversa hecha por el bloque *OFDM Carrier Allocator*. A la salida del bloque *OFDM Serializer* se tiene un único flujo de símbolos QPSK los cuales ya pueden ser demodulados.

Figura 54: Bloque *OFDM Serializer*

Demodulación

Para demodular los símbolos QPSK se utiliza el bloque *Constellation Decoder* que tiene de entrada señal compleja y tiene por salida bytes, se necesita configurarlo para conocer el tipo de modulación utilizada, en este caso se le da a conocer la modulación con el comando "`payload_mod.base()`".

Figura 55: Bloque *OFDM Serializer*

A la salida se tiene un flujo de 2 bits por Byte, por lo que para reconstruir la información de un flujo se debe volver a 8 bits por Byte, para lo cual se utiliza el bloque *Repack bits*.

Finalmente para multiplexar los flujos de bits se utiliza el bloque *Interleave*, donde se tiene como entrada los 2 flujos de Bytes y a la salida se utiliza el bloque *File Sink* que permite guardar la información obtenida en un archivo.

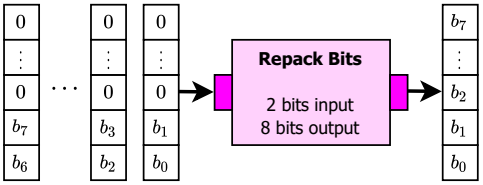


Figura 56: Bloque *Repack Bits*

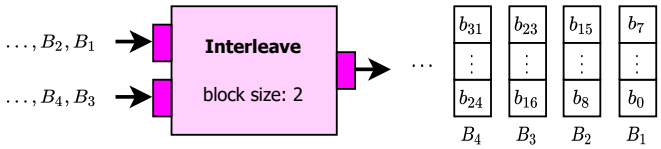


Figura 57: Bloque *Interleave* como multiplexor de bloques 2 Bytes

Figura 58: Bloque *File sink*

3.3. Algoritmo de *Water-filling*

3.3.1. Diseño de sistema de transmisión CSI conocido en el transmisor

A continuación en la figura 59 se ve un diagrama de bloques del diseño de asignación de potencia en las antenas transmisoras por medio de un bloque de realimentación que utiliza el algoritmo de *water-filling*.

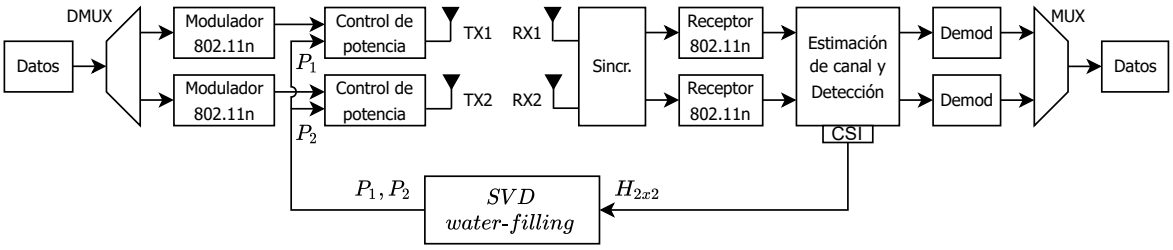


Figura 59: Diagrama de bloque de asignación de potencia por estrategia de *water-filing*. Elaboración propia.

3.3.2. Implementación de sistema MIMO-OFDM con CSI conocido en el transmisor

Para implementar un bloque de realimentación se utilizará el bloque *Embedded Python Block* que se muestra en la figura 60.

Para configurar el bloque embebido se necesita de un interprete de Python. Dentro del bloque permite utilizar editores de texto o de código que permiten modificar el funcionamiento del bloque dentro de GNU Radio, al crear el bloque se genera el siguiente segmento de código:

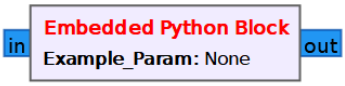


Figura 60: Bloque *Embedded Python Block*

```python
import numpy as np
from gnuradio import gr

class blk(gr.sync_block):
    def __init__(self, example_param=1.0):
        gr.sync_block.__init__(
            self,
            name='Embedded Python Block',
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        self.example_param = example_param
    def work(self, input_items, output_items):
        output_items[0][:] = input_items[0] * self.example_param
        return len(output_items[0])
```

Segmento de código 2: Código por defecto del bloque embebido de python

Algo importante a considerar es que la etapa de transmisión y recepción con uso de Hardware real, no se encuentran sincronizados, por lo que el uso de un bloque habitual para realimentar la información del canal no sería la mejor opción al momento de ejecutar el programa, puesto que debería efectuar el control a cada instante en la ejecución del programa, lo cual requeriría establecer un funcionamiento más complejo de su autonomía.

Para hacer la realimentación más eficiente, se utilizan mensajes asíncronos, que a diferencia del flujo habitual que se ha estado utilizando, este tipo de datos puede permitir la comunicación entre bloques, tanto con etapas continuas como con etapas anteriores, y actúa de manera asíncrona a la ejecución del programa, lo cual facilita el control. Para utilizar mensajes asíncronos en GNU Radio se utiliza PMT (*Polymorphic Types*), una librería integrada en GNU Radio que es usado como portador de datos entre bloques.

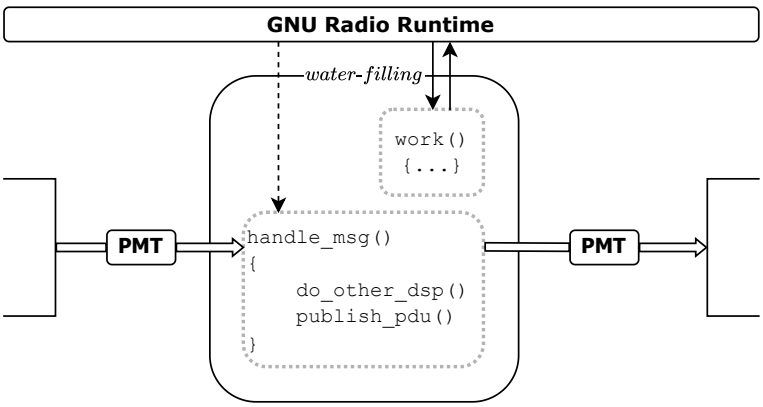


Figura 61: Funcionamiento de mensajería asíncrona. Elaboración propia.

Primero, para enviar la información de la matriz de canal MIMO, se modificó el bloque *Estimador/Detector* para que envíe las matrices de canal estimadas por un puerto PMT. Luego se creó el bloque *SVD Water-filling* que recepciona y procesa las matrices de canal, así también calcula los valores de potencia óptima a asignar a los transmisores. Los bloques *UHD: USRP Sink* admiten mensajes PMT para ser reconfigurados, y para modificar la potencia de transmisión se modifica la ganancia de transmisión en dB, por tanto los valores de potencia óptima son enviados como ganancias en formato PMT.

La creación de los puertos de mensajería utiliza el siguiente segmento de código:

```
import pmt # importacion de libreria PMT
class blk(gr.sync_block):
    def __init__(self):
        gr.sync_block.__init__(...) # Parametros del bloque
        # Creacion de puerto de transmision
        self.message_port_register_out(pmt.intern("out"))
        # Creacion de puerto de recepcion
        self.message_port_register_in(pmt.intern("in"))
        # Funcion para recepcion de mensajes
        self.set_msg_handler(pmt.intern("csi"), self.msg_handler)
```

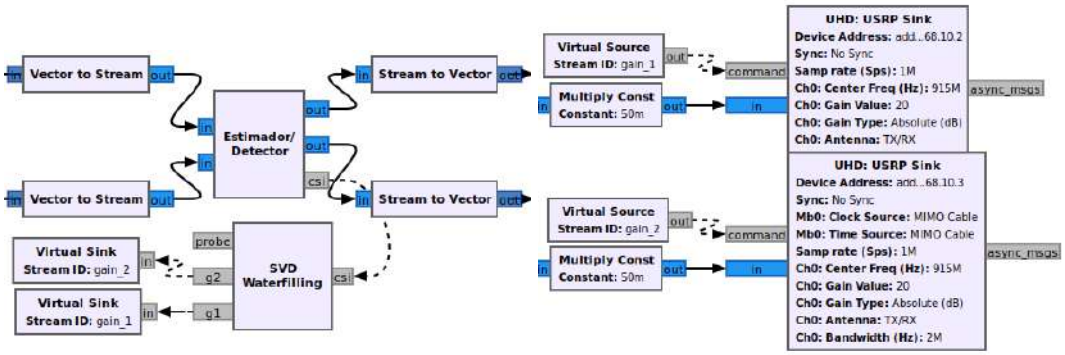
Segmento de código 3: Creación de puertos de mensajería PMT

El envío de datos en formato PMT utiliza el siguiente segmento de código:

```
pmt_msg = pmt.to_pmt(chans)       # Conversion a formato pmt
self.message_port_pub(pmt.intern("csi"),pmt_msg) # envio de mensaje
```

Segmento de código 4: Envío de mensaje PMT

En la figura 62 se muestra la conexión del bloque *SVD Water-filling*.



(a) Bloque *SVD Water-filling* en el receptor

(b) Reconfiguración de bloques *UHD: USRP Sink* en el transmisor

Figura 62: Bloque *SVD Water-filling* de realimentación al transmisor

El algoritmo de asignación de potencia óptima con algoritmo de *water-filling* implementado en el bloque *SVD water-filling* se muestra en la figura 63.

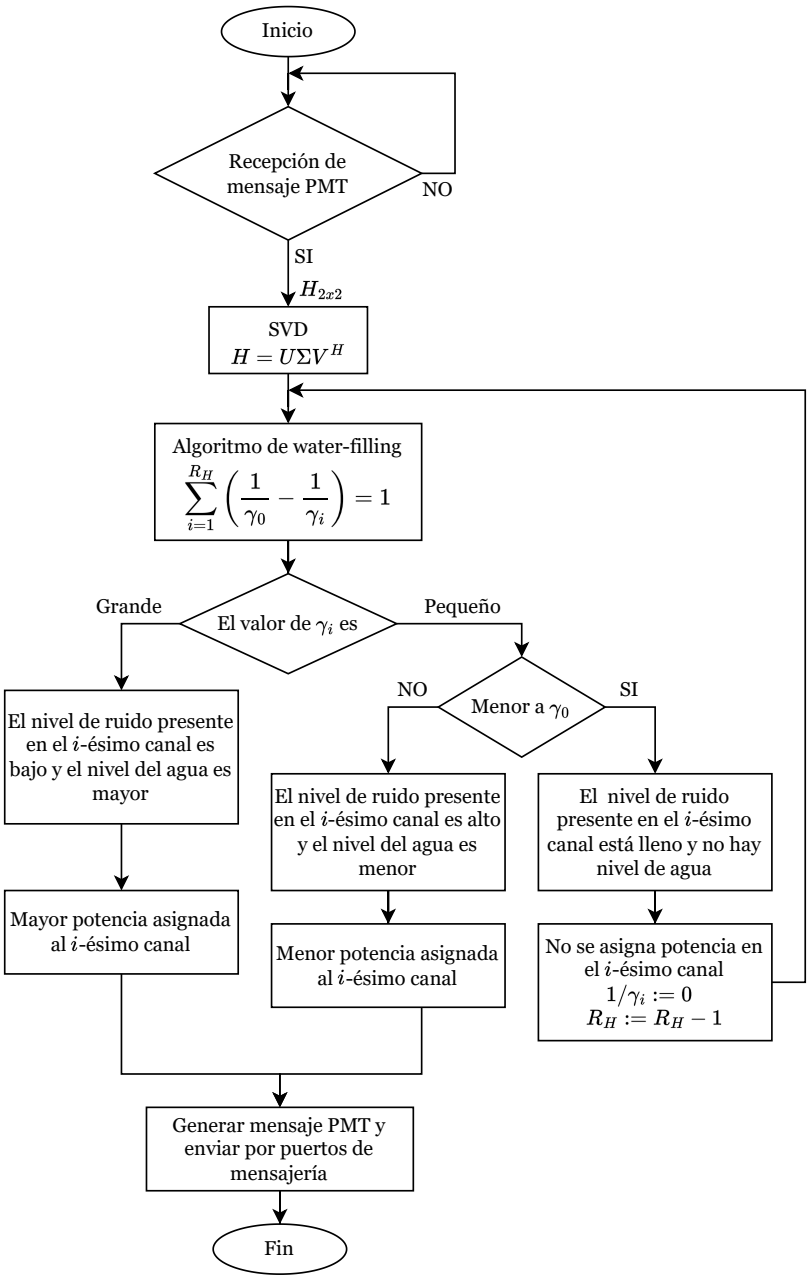


Figura 63: Algoritmo de *water-filling*. Elaboración propia.

Cada el bloque *SVD water-filling* recepciona un mensaje PMT, se ejecuta la función de control de eventos de recepción "`msg_handler`", en donde se desarrolla todo el algoritmo. En orden para procesar un mensaje PMT, primero se extrae el contenido convirtiendo el formato del mensaje a una matriz de la librería *numpy*.

```
def msg_handler(self, msg):
        pmt_msg = pmt.to_python(msg)    # Conversion de PMT a Python
        chans = []
        for chan_msg in pmt_msg:
            Hm = np.array([[abs(chan_msg[0]), abs(chan_msg[1])],
                           [abs(chan_msg[2]), abs(chan_msg[3])]])
            chans.append(Hm)
```

```
8         chans = np.asarray(chans)  # lista de matrices H en magnitud
```

Segmento de código 5: Recepción y conversión de mensaje PMT a matriz numpy

Teniendo las matrices en un formato más manejable, se aplica la descomposición de valores singulares mediante la función ”svd” de *numpy*. Seguidamente, se aplica el algoritmo de *water-filling*, que calcula γ_0 y γ_i para obtener los factores de asignación de potencia.

```
pwrs_vec = []
for Hm in chans:
    rows, cols = Hm.shape
    U, D, V = np.linalg.svd(Hm)
    gammas = [(sigma**2)*self.rho for sigma in D]
    # Gamma cero
    gamma_zero = self.gammaZero(gammas)
    # Verificando incoherencia
    gammas, new_gamma_flag = self.checkGamma(gamma_zero, gammas)
    # nuevo valor de gamma cero
    if new_gamma_flag:
        gamma_zero = self.gammaZero(gammas)
    # factor de asignacion de potencia
    mu = 1/gamma_zero
    factors = self.pwrFactors(gammas, gamma_zero, rows)
    pwrs_vec.append(factors)
pwrs_vec = np.asarray(pwrs_vec)
```

Segmento de código 6: Algoritmo de water-filling

Para evitar que toda la potencia se asigne a un solo transmisor, se restringe el umbral de potencia asignable mediante el siguiente segmento de código.

```
avr0 = 0
avr1 = 0
c = 0
for factors in pwrs_vec:
    avr0 = (avr0 + factors[0]) if factors[0] <= 0.9 else (avr0 + 0.9)
    avr1 = (avr1 + factors[1]) if factors[0] <= 0.9 else (avr0 + 0.1)
    c = c + 1
avr0 = avr0/c
avr1 = avr1/c
```

Segmento de código 7: Factores de asignación de potencia

Después de calcular los factores de asignación de potencia, estos son convertidos a un valor de ganancia en dB admisible a partir de las especificaciones del USRP 2920 como transmisor, de acuerdo a la tabla 2. Además, se define un parámetro de referencia en dB que representa el 50 % de potencia asignada inicialmente, esto debido a que los transmisores no comparten potencia y ayudan a definir la potencia máxima que se les puede asignar a cada uno.

```
Pt_max_dBm = 20 # [dBm] potencia maxima en cada canal
```

```
Gain_ref = self.Gain_ref # [dB] Referencia de 50% de potencia
gain_n210 = [i for i in range(0,31+1)] # Rango de ganancias USRP 2920
Pt_dBm_0dB = Pt_max_dBm - gain_n210[len(gain_n210)-1]
Power_ref = self.dBmToWatts(Pt_dBm_0dB + Gain_ref)
Pt_max = 2*Power_ref  # [mW] Restriccion de potencia para ambos USRP
P1 = pwr_factor*Pt_max
P2 = (1-pwr_factor)*Pt_max
P2_dBm = self.WattsTodBm(P2)
P1_dBm = self.WattsTodBm(P1)
gain1 = np.round(P1_dBm - gain_0dB,0)
gain2 = np.round(P2_dBm - gain_0dB,0)
```

Segmento de código 8: Calculo de ganancia óptima en dB

Finalmente, los valores de ganancia calculados son convertidos a formato PMT y enviados por medio de los puertos de mensajería a los bloques USRP. Los parámetros de ganancia de los USRP son reconfigurados y se modifica la potencia de transmisión de cada uno.

```
# puerto "probe" factores/ganancias/umbrales
pmt_probe = pmt.to_pmt([mu,avr0,avr1])
self.message_port_pub(pmt.intern("probe"),pmt_msg)
# Comandos de configuracion de ganancia para los USRP
ant_msg = "TX/RX"
pmt_g1 = pmt.to_pmt({'antenna': ant_msg, 'gain': gain1, 'chan': 0})
pmt_g2 = pmt.to_pmt({'antenna': ant_msg, 'gain': gain2, 'chan': 0})
self.message_port_pub(pmt.intern("g1"),pmt_g1)
self.message_port_pub(pmt.intern("g2"),pmt_g2)
```

Segmento de código 9: Reconfiguración de USRP por mensaje asíncrono

El programa completo del bloque *SVD water-filling* se encuentra en el Anexo F.

3.4. Uso de los dispositivos USRP

En esta parte se describirá como se hizo la conexión y comunicación de los dispositivos USRP en Ubuntu 20.04 LTS.

Antenas y frecuencia de transmisión

Para la transmisión se utilizará las antenas VERT900 que puede operar en el rango de frecuencias de 824 a 960 MHz y 1710 a 1990 MHz.

Figura 64: Antena Vert900

USRP Hardware Driver (UHD)

Para efectuar la comunicación de los dispositivos de radio definida por software USRP con el ordenador, es necesario instalar los controladores de código abierto UHD que es compatible con todos los modelos USRP.

El método para instalar los drivers de manera rápida y eficiente es a partir de los repositorios públicos de Ubuntu, por lo que se utilizan los siguientes comandos en una temrinal:

```
1 $ sudo apt update
2 $ sudo apt install libuhd-dev uhd-host
```

Segmento de código 10: Instalación controladres UHD

3.4.1. Conexión y uso del NI USRP 2920

Primero para establecer la conexión del USRP 2920 al ordenador se utiliza el puerto de conexión 1Gb Ethernet del panel frontal como se muestra en la figura 65.

Figura 65: Conexión del USRP 2920 por 1GbE

Ahora se debe establecer la configuración del adaptador del puerto conectado en la misma subred que el dispositivo, ya que por diseño de fábrica todos los dispositivos inicialmente

tienen la dirección “192.168.10.2” por lo se que se configura una dirección estática como se muestra en la figura 66.

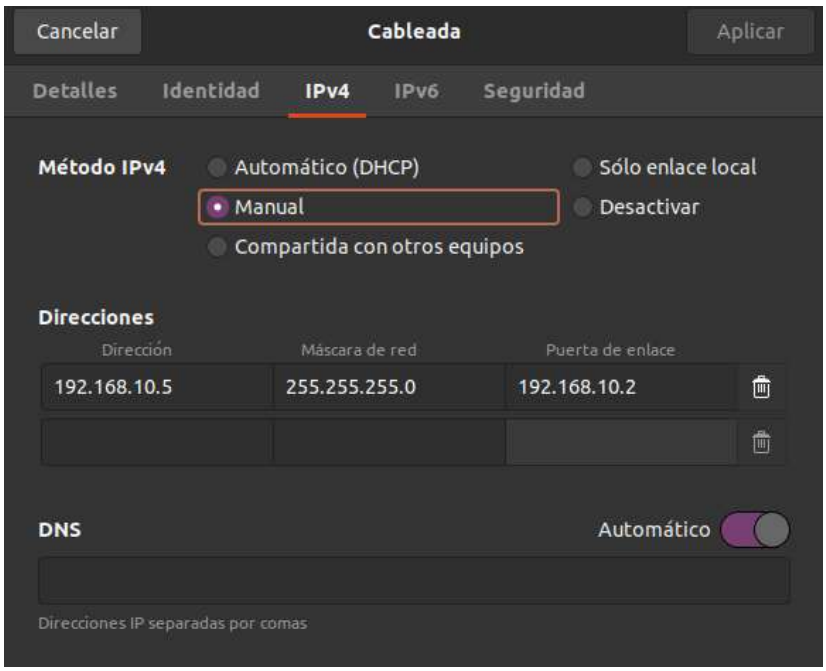


Figura 66: Configuración de IP estática

Ahora es posible detectar la conexión del USRP por lo que se puede utilizar un comando para medir la latencia de la conexión, en una terminal se ejecuta el siguiente comando:

```
1 $ ping 192.168.10.2
```

Segmento de código 11: Medición de latencia a dirección IP

Como resultado se debe tener:

Figura 67: Latencia de la conexión con el USRP

Antes de utilizar la verificación el controlador UHD se descargan las imágenes de FPGA de todos los modelos de USRP utilizando el siguiente comando en una terminal:

```
$ sudo uhd_images_downloader
```

Segmento de código 12: Actualización de imágenes de los FPGA

El controlador UHD permite detectar y verificar el funcionamiento de los dispositivos USRP mediante los siguientes comandos:

```
$ uhd_find_devices
$ uhd_usrp_probe
```

Segmento de código 13: Comandos UHD

Figura 68: Detección del NI USRP 2920

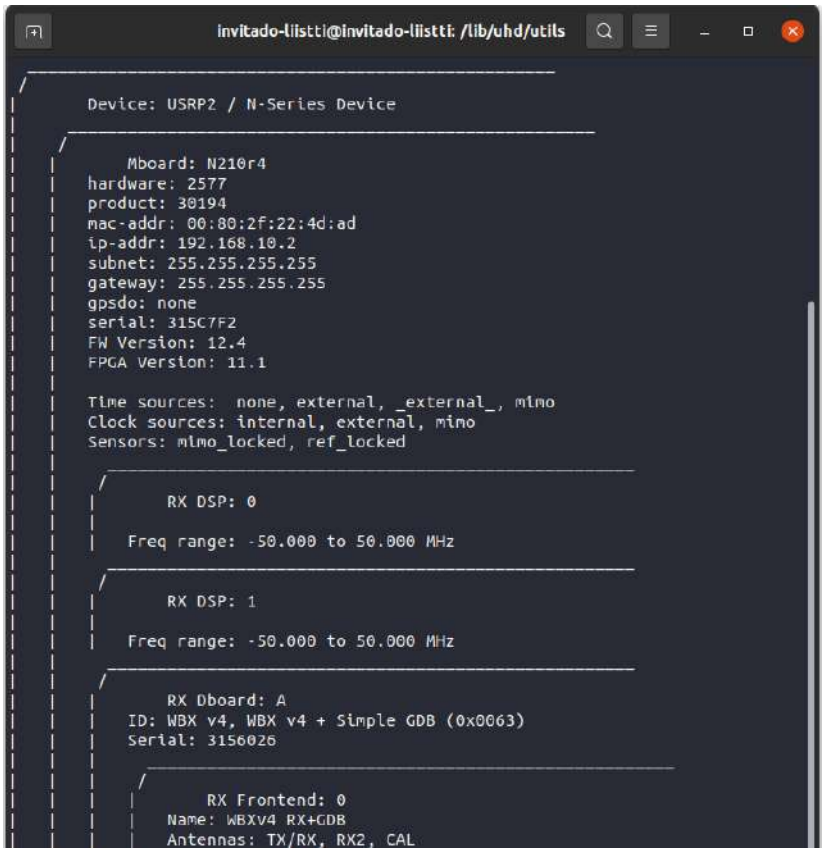


Figura 69: Prueba del NI USRP 2920

Antes de conectar ambos USRP 2920, es necesario hacer un cambio de dirección IP debido a que vienen con la misma dirección por defecto. Entonces, se hace uso de la aplicación que viene con UHD: "*usrp_burn_mb_eeprom*", que está ubicado en "*/lib/uhd/utils*". A continuación se utilizan los siguientes comandos en una terminal:

```
$ cd /lib/uhd/utils
$ ./usrp_burn_mb_eeprom --args="addr=192.168.10.2" --values="ip-addr
    =192.168.10.3"
```

Segmento de código 14: Cambio de dirección IP

Luego se utiliza el cable de conexión MIMO, que se muestra en la figura 70, que actúa como una configuración maestro-esclavo entre los USRP 2920. El cable MIMO permite compartir un mismo reloj de referencia, sincronizar los dispositivos y conexión Ethernet de ambos.

Figura 70: Cable MIMO

Finalmente se utilizan los puertos TX1 con las antenas VERT900 teniendo los USRP como se muestra en la figura 71.

Figura 71: Transmisor MIMO 2x2 con NI USRP 2920 y antenas VERT900

Y en la terminal ya son detectables ambos USRP utilizando el mismo comando de detección de dispositivos de UHD tal como ve en la figura 72.

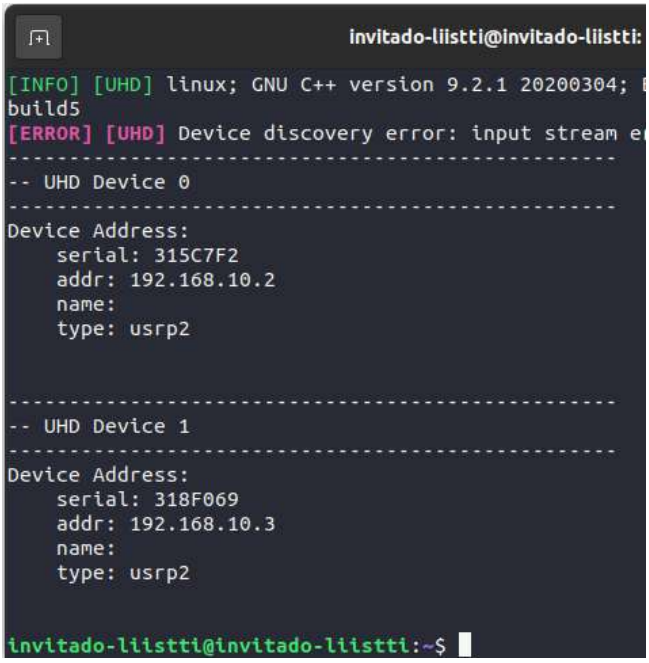


Figura 72: Detección de ambos NI USRP 2920

Ahora para programar los USRP como transmisores desde GNU Radio se utiliza el bloque *UHD: USRP Sink* que tiene de entrada una señal compleja como se muestra en la figura 73.

(a) Bloque *UHD:USRP Sink* para el primer USRP

(b) Bloque *UHD:USRP Sink* para el segundo USRP

Figura 73: Bloques de transmisión USRP

En el apartado de *General* para el primer bloque se tienen los siguientes parámetros:

- **Device address:** “addr=192.168.10.2” (Dirección IP del primer USRP 2920)
- **Sync:** No Sync (No se usa señal de referencia para sincronizar tiempo y reloj de los dispositivos)
- **Clock Source:** Default
- **Time Source:** Default

- **Samp rate (Sps):** *samp_rate* (Frecuencia de muestreo)

En el apartado de *General* para el segundo bloque se tienen los siguientes parámetros:

- **Device address:** "addr=192.168.10.3" (Dirección IP del segundo USRP 2920)
- **Sync:** No Sync (No se usa señal de referencia para sincronizar tiempo y reloj de los dispositivos)
- **Clock Source:** MIMO Cable (Comparte reloj de referencia con el dispositivo maestro)
- **Time Source:** MIMO Cable (Sincronización temporal con el dispositivo maestro)
- **Samp rate (Sps):** *samp_rate* (Frecuencia de muestreo)

En el apartado de *RF* se tienen los siguientes parámetros:

- **Center Freq (Hz):** *Frequency* (Frecuencia de transmisión)
- **Gain Value:** *gain_tx1* (Ganancia de transmisión para el bloque 1)/ *gain_tx2* (Ganancia de transmisión para el bloque 2)
- **Gain type:** Absolute(dB)
- **Antenna:** TX/RX (puerto de transmisión)
- **Bandwitdth:** *BW* (Ancho de banda)

3.4.2. Conexión y uso del NI USRP 2950R

Para el USRP 2950R, se utiliza la interfaz por PCI Express haciendo uso del kit de conectividad PCIe que se muestra en la figura 74a, y su conexión por medio de su puerto PCIe como se muestra en la figura 74b.

(a) Kit de conectividad PCI Express

(b) Conexión PCIe del NI USRP 2950R

Figura 74: Conexión por PCI de USRP

Para verificar la conexión PCI se utiliza el comando en la terminal:

```
1 $ lspci
```

Segmento de código 15: Detección PCI

Se obtiene un listado de todos los dispositivos conectados por PCI y entre ellos el de la marca National Instruments que confirma la conexión como se muestra en la figura 75.

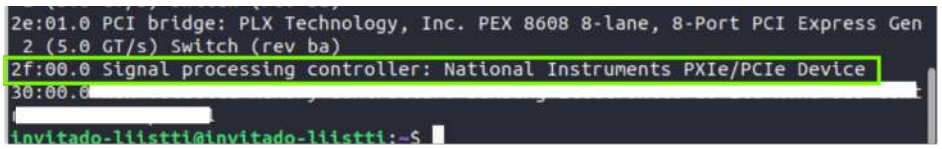


Figura 75: Listado de conexiones PCI

Para el uso de una interfaz por PCI Express requiere de un controlador adicional llamado *NI-RIO*, a continuación se describe el procedimiento de su instalación.

Primero se descargan los controladores de la página oficial de National Instruments, para esta tesis se utilizó la versión 2021Q3. Uno de los problemas encontrados al momento de instalar este controlador es la incompatibilidad con los kernel actuales de Ubuntu, por lo que se optó por utilizar el kernel de baja latencia, el cual se instala con el siguiente comando:

```
$ sudo apt-get install linux-lowlatency
```

Segmento de código 16: Instalación del kernel de baja latencia

Y también se recomienda desinstalar cualquier versión del kernel *OEM*, en caso se tenga por defecto, y toda versión del kernel *generic* por delante de la versión 5.13 utilizando el aplicativo *mainline*, y mediante el GRUB de arranque del ordenador se escoge únicamente el kernel *lowlatency*.

Una vez teniendo descargado los archivos del controlador *NI_RIO*, se descomprime y se busca el archivo de nombre "*ni-ubuntu2004firstlook-drivers-2021Q3.deb*", se abre el archivo y se instala o mediante la terminal se utiliza el comando:

```
$ sudo dpkg -i ni-ubuntu2004firstlook-drivers-2021Q3.deb
```

Segmento de código 17: Instalación del repositorio de NI-RIO

Ahora se actualizan los repositorios de Ubuntu y se instalan las cabeceras del kernel:

```
$ sudo apt update
$ sudo apt install linux-headers-$(uname -r)
```

Segmento de código 18: Instalación de cabeceras

Se procede a instalar el paquete de *ni-usrp-rio*:

```
$ sudo apt install ni-usrp-rio
```

Segmento de código 19: Instalación de controlador NI-RIO

Para resolver ante la incompatibilidad del programa "*ni-kal*" se recomienda reinstalar utilizando los siguientes comandos:

```
$ sudo touch /usr/src/linux-headers-XX.YY.ZZ-generic/include/config/
    modversions.h
$ sudo apt reinstall ni-kal
```

Segmento de código 20: Solución de incompatibilidad

Si la instalación no presentó algún problema, entonces el dispositivo ya puede ser detectado por el controlador UHD como se muestra en la figura 76, y la verificación de su funcionamiento en la figura 77. Se usan los mismos comandos que se utilizó para el NI USRP 2920.

Figura 76: Detección del NI USRP 2950R

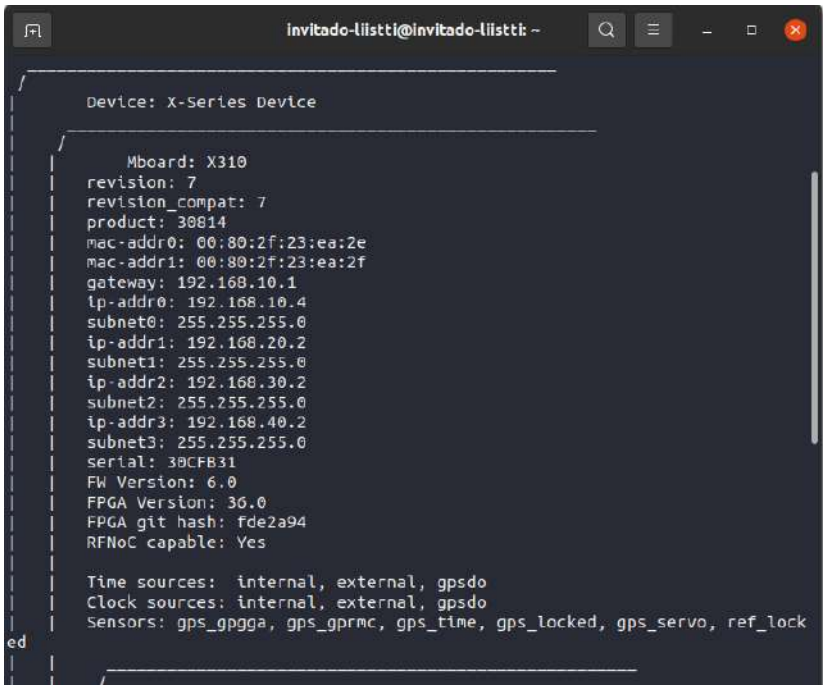


Figura 77: Prueba del NI USRP 2954R

Finalmente se utilizan los puertos RX2 de ambos canales del USRP 2950R con las antenas VERT900 quedando como se muestra en la figura 78.

Figura 78: Receptor MIMO 2x2 con NI USRP 2950R

Para programar el USRP 2954R desde GNU Radio como un receptor se utiliza el bloque *UHD: USRP Source* que tiene como salida una señal compleja como se muestra en la figura 79.

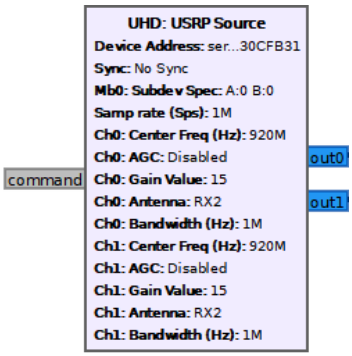


Figura 79: Bloque *UHD: USRP Source* de recepción

En el apartado de *General* para el bloque se tienen los siguientes parámetros:

- **Device address:** “serial=30CFB31” (Serial del USRP 2950R)
- **Sync:** No Sync (No se usa señal de referencia para sincronizar tiempo y reloj de los dispositivos)
- **Clock Source:** Default
- **Time Source:** Default
- **Subdev Spec:** “A:0 B:0” (Especificación de canales RF)
- **Num Channels:** 2 (Número de canales RF)
- **Samp rate (Sps):** *samp_rate* (Frecuencia de muestreo)

En el apartado de *RF* se tienen los siguientes parámetros para ambos bloques:

- **Ch0 Center Freq(Hz):** *frequency* (Frecuencia de recepción)

- **Ch0 AGC:** Disabled
- **Ch0 Gain Value:** *gain_rx1* (Ganancia de recepción)
- **Ch0 Gain type:** Absolute(dB)
- **Ch0 Antenna:** RX2 (puerto de transmisión)
- **Ch0 Bandwitdth:** *BW* (Ancho de banda)
- **Ch1 Center Freq(Hz):** *Frequency* (Frecuencia de recepción)
- **Ch1 AGC:** Disabled
- **Ch1 Gain Value:** *gain_rx2* (Ganancia de recepción)
- **Ch1 Gain type:** Absolute(dB)
- **Ch1 Antenna:** RX2 (puerto de transmisión)
- **Ch1 Bandwitdth:** *BW* (Ancho de banda)

Definición de parámetros

Como frecuencia portadora de transmisión y recepción, se utiliza la frecuencia de 920 MHz en una variable de nombre *frequency*, ya que la antena VERT900 puede trabajar en la banda de frecuencias ISM de 915 MHz.

Los valores de ganancia de cada USRP modifican la potencia tanto de transmisión como de recepción en los canales y los valores aceptados son en unidades dB, estos se guardan en las variables *gain_tx1*, *gain_tx2*, *gain_rx1* y *gain_rx2*, en base a las especificaciones referidas de las tablas 2 y 5.

Las máximas frecuencias de muestreo para los USRP 2920 y 2950R, son 400MS/s y 200MS/s respectivamente, pero en el caso del 2920 mediante una conexión por 1GbE solo permite hasta un máximo de 25MS/s a diferencia del 2950R que su conexión PCI si permite utilizar el máximo valor [16]. Para optimizar el procesamiento de la comunicación se utiliza un valor de 1MS/s en la variable *samp_rate*.

El ancho de banda se configura para 1 MHz ya que es el máximo ancho de banda utilizable al utilizar 1MS/s como frecuencia de muestreo y se guarda en la variable *BW*. En base a los parámetros de seleccionados y el estándar 802.11n se tendría una tasa de datos máxima de 2.6 Mbps para 1MHz de ancho de banda.

CAPITULO IV

4. Pruebas y Resultados del sistema implementado

En este capítulo se muestran las pruebas efectuadas y los resultados obtenidos del sistema implementado en plataforma de SDR, se realizaron distintos análisis ejecutándose instrucciones en el entorno de GNU Radio y se ilustra los resultados utilizando la herramienta computacional Octave.

4.1. Comunicación del enlace inalámbrico

Las pruebas del enlace punto a punto se realizaron utilizando la banda ISM en la frecuencia de 920 MHz, donde se usaron 3 equipos SDR. En la etapa de transmisión se tienen 2 equipos SDR sincronizados en configuración MIMO, que transmiten un mensaje de texto con modulación QPSK utilizando OFDM. En la parte del receptor se tiene un equipo SDR con 2 canales utilizados como receptor MIMO. Esto se muestra en la figura 80.

Figura 80: Sistema de comunicación MIMO 2x2 con Radios USRP

Los resultados y análisis se muestran a continuación.

4.1.1. Transmisor

Al momento de ejecutar el programa, se utilizan ventanas de instrumentación para ver la señal OFDM en el tiempo y su espectro de frecuencias en banda base antes de transmitirla mediante los equipos SDR, esto se puede ver en la figuras 81 y 82, donde el ancho de banda útil es de 894 Khz que es utilizado por las 57 portadoras: 52 de datos, 4 de pilotos, 1 portadora central; y además 7 portadoras laterales como bandas de guarda.

Cabe resaltar que la amplitud de las señales OFDM deben que tener una amplitud entre -1 a 1 para que las mismas puedan ser convertidas a señales analógicas.

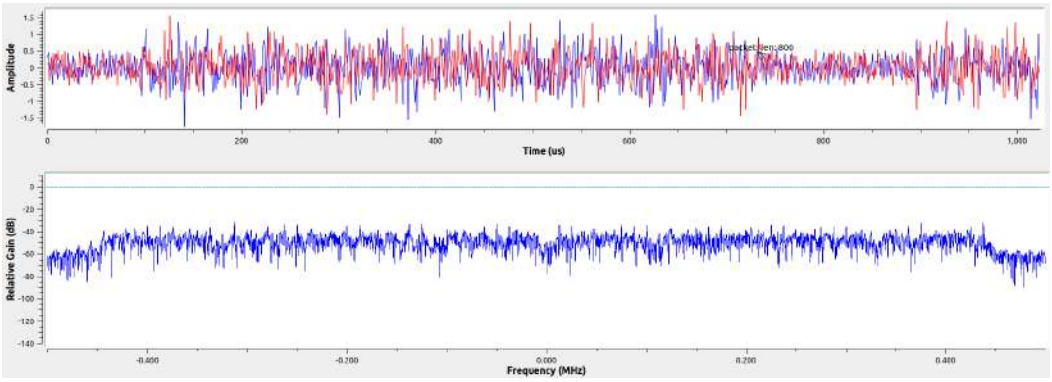


Figura 81: Ventanas en tiempo y frecuencia de señal OFDM en el transmisor 1

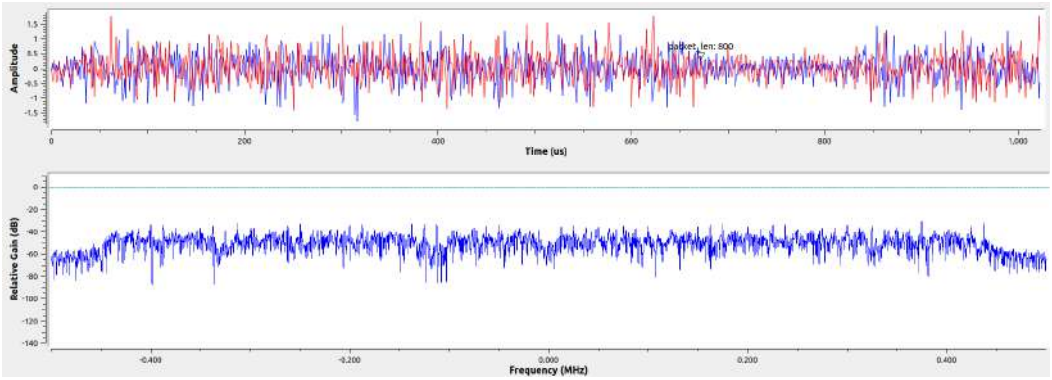


Figura 82: Ventanas en tiempo y frecuencia de señal OFDM en el transmisor 2

4.1.2. Receptor

Cuando se efectuó la recepción, se recibió por cada antena una superposición de ambas señales OFDM transmitidas y se utilizan ventanas en tiempo y frecuencia para ver las mismas, esto se muestra en la figuras 83 y 84, donde el ancho de banda utilizado es de 900 Khz que pasa a través de un filtro de 1 MHz configurado en el USRP y la amplitud digital máxima, después de la recepción, también es de -1 a 1.

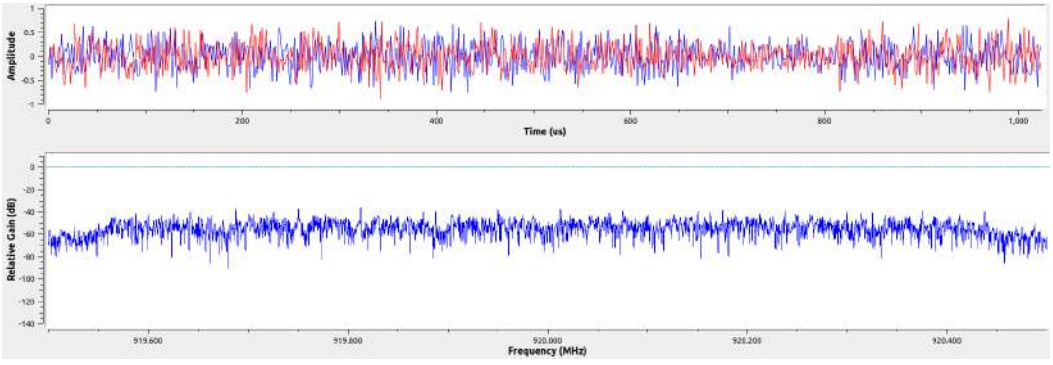


Figura 83: Ventanas en tiempo y frecuencia de señal sintonizada en el receptor 1

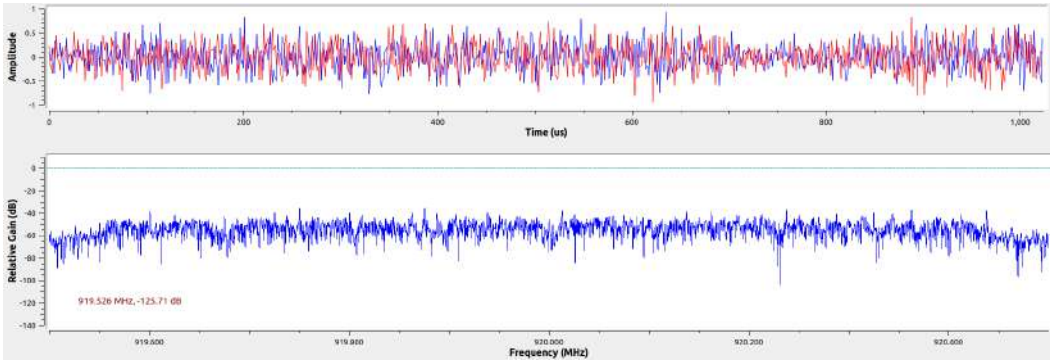


Figura 84: Ventanas en tiempo y frecuencia de señal sintonizada en el receptor 2

4.2. Tasas de error de bit

Para evaluar la información errada en la comunicación se utiliza un bloque comparativo de la información transmitida y la información recepcionada.

Debido a que el bloque *Estimador/Detector* [19] procesa el flujo de símbolos para extraer los símbolos de datos, los símbolos de sincronismo son reemplazados por ceros. Por lo que se utiliza el siguiente segmento de código para procesar el texto.

```
# Acceso a archivos
with open("file.txt",'r',encoding='utf-8',errors='ignore') as file:
    text1 = file.read()
# Rermoviendo espacios
text1 = text1.splitlines()
for i in range(0,len(text1)):
    text2[i] = text1[i].replace("\x00","")
    text2[i] = text1[i].replace("\t","")
    text2[i] = text1[i].replace(" ","")
# Reescribiendo los archivos de texto
with open("file_out.txt",'w',encoding='utf-8') as file:
    for line in text1:
        file.write(line)
```

Segmento de código 21: Procesamiento de texto

Después de haber corregido el texto, para comparar los archivos se utiliza el entorno GRC. Primero se cargan los archivos con el bloque *File Source* y se generan flujos de bits por medio del bloque *Repack Bits* configurado de 8 bits/Byte a 1 bit/Byte. A continuación se calcula la tasa de error de bit utilizando un bloque de Python de nombre BER. La funcionalidad del bloque BER se basa en la comparación de los flujos de bits, se guarda en un contador el número de muestras procesadas, en otro contador el número de muestras erradas y se calcula la tasa de error de bits como una relación entre el número de errores y el total de muestras procesadas. El segmento de código utilizado es el siguiente:

```
count = 0
errors = 0
```

```
BER = 1
Errors_max = int(1e6)
Sym_max = int(10e6)
if errors < Errors_max:
    if count < Sym_max:
        count += len(in0)  # Cuenta las muestras de la primera entrada
        errors += np.count_nonzero(in0-in1) # Compara las muestras con la segunda entrada
        BER= errors / count # Tasa de error
out2[:]=BER
out1[:]=errors
out0[:]=count
```

Segmento de código 22: Cálculo de la tasa de error de bit

En la figura 85 se muestra el diagrama de bloque para calcular la BER en entorno de GRC.

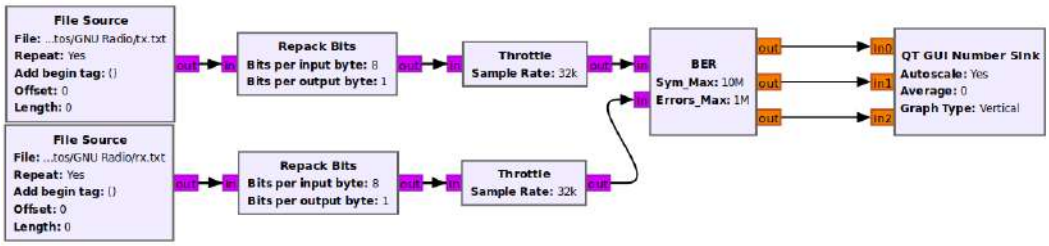


Figura 85: Cálculo de la BER en GNU Radio

4.2.1. Pruebas en simulaciones

Para las simulaciones se plantean 3 escenarios modelando distintos canales MIMO 2x2 para verificar el funcionamiento de los algoritmos.

Para simular un canal como una matriz de ganancias se utilizan los siguientes bloques:

- *Multiply const*: Multiplica la señal por un valor de ganancia real o complejo.
- *Noise Source*: Genera una fuente de ruido de distribución Gaussiana o uniforme.
- *Delay*: Produce un retardo añadiendo ceros al inicio de una señal.
- *Tag Gate*: Evita propagar las etiquetas de la señal ya que estas no se transmiten en Hardware real.

Como primer escenario se simuló un canal MIMO 2x2 ideal sin retardo y sin ruido, con ganancias de canal cercanas al valor de 1 para que altere la señal mínimamente y a la vez la transmisión tenga trayectorias con ganancias diferentes como para no ser considerada una matriz singular, esto está dado por la ecuación 20 donde se representa al canal MIMO como una matriz 2x2 y se implementa en simulación como se ve en la figura 86.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1.2 & 0.98 \\ 0.9 & 1.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{20}$$

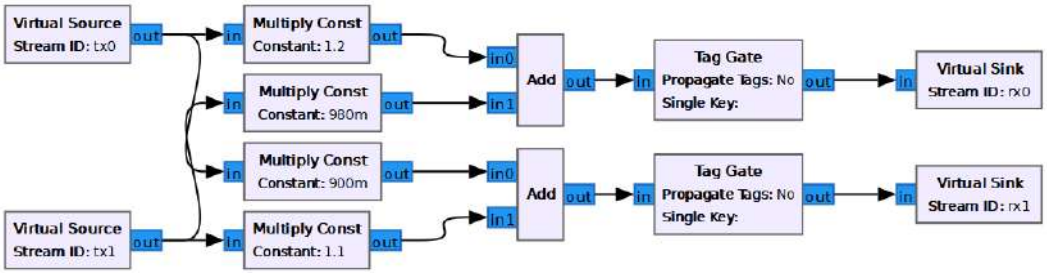


Figura 86: Canal MIMO 2x2 simulado en GNU Radio

Como segundo escenario se simuló un canal MIMO 2x2 con ganancias complejas, efecto de retardo temporal y ruido con distribución Gaussiana adicionado a cada receptor. Este canal está dado por la ecuación 21, y su implementación en simulación se muestra en la figura 87.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0.8 & 0.21 - 0.72j \\ -0.2j & -0.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \tilde{n}_1 \\ \tilde{n}_2 \end{bmatrix} \tag{21}$$

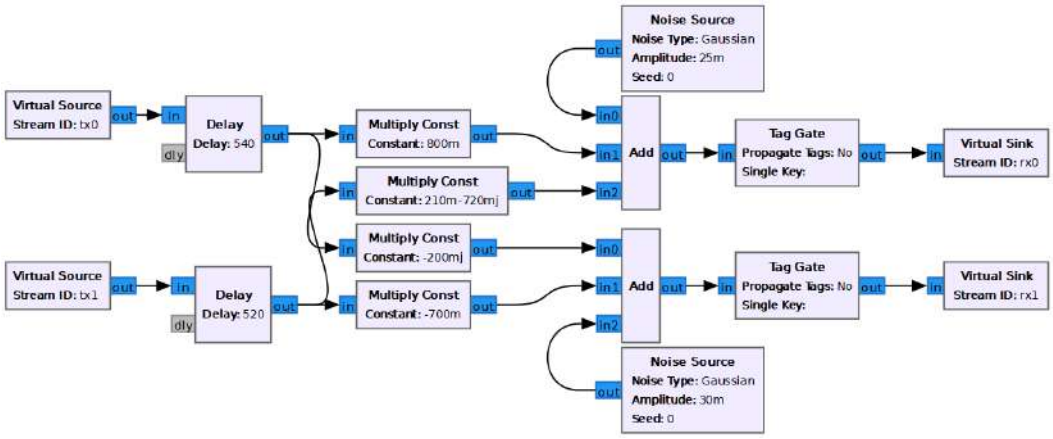


Figura 87: Canal MIMO 2x2 con retardo temporal y ruido Gaussiano simulado en GNU Radio

Para el tercer escenario, utilizando el mismo canal con efectos de retardo y ruido, se aplica la asignación de potencia óptima, donde se utilizó el puerto *probe* del bloque de realimentación junto con un bloque *Message Debug* que imprime en la terminal los coeficientes de asignación de potencia.

Los valores de salida tienen el siguiente formato: (umbral, factor1, factor2), donde *factor1* y *factor2* tienen que sumar 1, como se ve en la figura 89.

En la figura 90 se ve como el algoritmo de *water-filling* calcula los factores de asignación de potencia óptima. Para cada caso de simulación se considera que la amplitud de salida

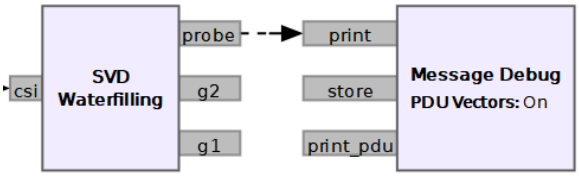


Figura 88: Obtención de factores de potencia óptima

de cada flujo utiliza un 50 % de la potencia de transmisión total. Los factores de asignación de potencia estimados son aproximadamente de 0.6 y 0.4, lo que quiere decir que al primer transmisor se le asigna una potencia del 60 % y para el segundo un 40 %, que en caso de simulación modifica únicamente la amplitud.

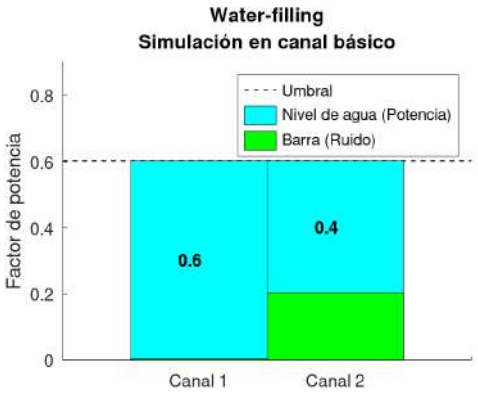
```
******* MESSAGE DEBUG PRINT ********
#(0.60298 0.599536 0.400464)
**********************************
******* MESSAGE DEBUG PRINT ********
#(0.60298 0.599536 0.400464)
**********************************
******* MESSAGE DEBUG PRINT ********
#(0.60298 0.599536 0.400464)
**********************************
******* MESSAGE DEBUG PRINT ********
#(0.60298 0.599536 0.400464)
**********************************
******* MESSAGE DEBUG PRINT ********
#(0.60298 0.599536 0.400464)
**********************************

>>> Done
```

(a) Factores para canal básico

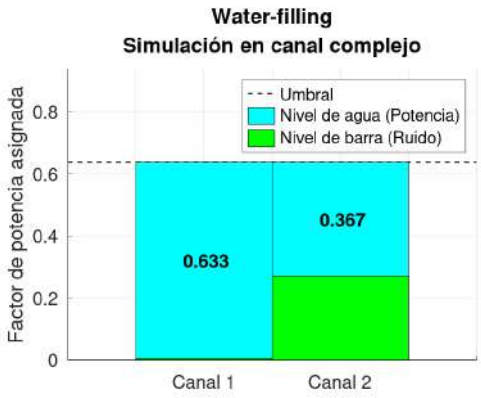
```
******* MESSAGE DEBUG PRINT ********
#(0.646844 0.640982 0.359018)
**********************************
******* MESSAGE DEBUG PRINT ********
#(0.629095 0.622964 0.377036)
**********************************
******* MESSAGE DEBUG PRINT ********
#(0.637754 0.632536 0.367464)
**********************************
```

(b) Factores para canal complejo

Figura 89: Valor de factores de asignación de potencia para el transmisor



(a) Nivel de agua en el canal básico



(b) Nivel de agua en el canal complejo

Figura 90: *water-filling* para los canales de simulación

Water-filling	**Sin WF**		**Con WF**	
Canal	Básico	Complejo	Básico	Complejo
BER	28.68×10^{-3}	196.2×10^{-3}	5.9×10^{-3}	102.28×10^{-3}

Tabla 6: Mediciones de la BER en simulaciones

En base a los resultados obtenidos de la tabla 6 de la tasa de error de bits, se comprueba que al añadir efectos de canal, se comprueba que en un canal complejo la BER de 0.1962 es superior a un canal básico con 0.028, al aplicar *water-filling* se reduce la tasa de error de bits en el canal básico a 0.0059 y en el canal complejo a 0.10, con *water-filling* las BER son inferiores comparado a cuando no se utiliza *water-filling*.

4.2.2. Pruebas con hardware USRP

Para evaluar el desempeño con hardware USRP, se realizaron pruebas de distancia con distintas ganancias de transmisión, a distancias de 1, 1.5, 2 y 3 metros. En cada caso, primero se comunicó como un simple enlace de bajada y luego con la información de canal realimentada aplicando el algoritmo de *water-filling*.

Tomando en cuenta de que los equipos SDR de transmisión no comparten potencia, se toma una ganancia de referencia para ambos como indicador del 50 % de la potencia total utilizada. Cuando la potencia de recepción alcanza el límite, que es de -15dBm, la amplitud de las muestras de la señal digitalizada son limitadas en el rango de -1 a 1 en el entorno de GNU Radio.

En las pruebas realizadas para 1 metro de distancia con 15 dB de ganancia, tanto en la transmisión como en la recepción, la potencia recibida no alcanzó el límite de -15dBm, por lo que en la figura 91 se ve las señales OFDM muestreadas de manera correcta.

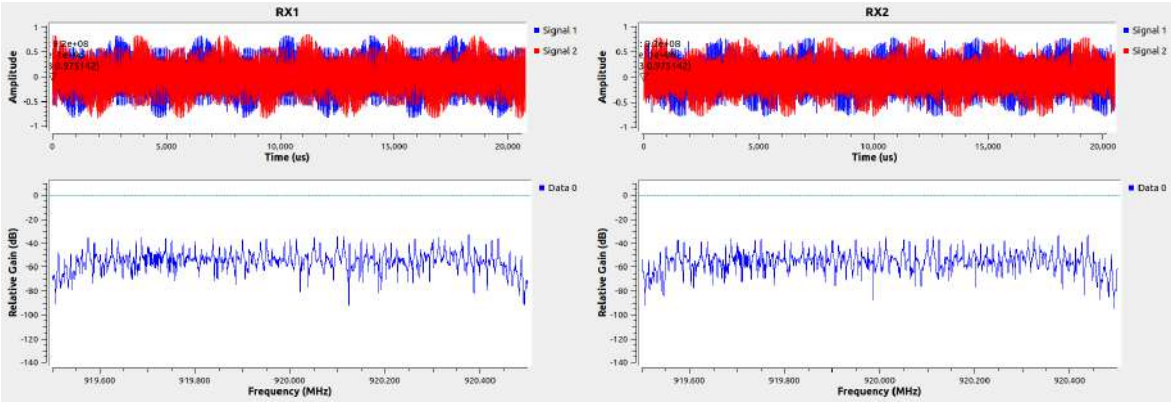


Figura 91: Señal recibida en tiempo y frecuencia con 15 dB de ganancia

Pero al utilizar valores de ganancia de transmisión superiores a 20 dB a 1 metro, se alcanza el límite de potencia de recepción, por lo que las señales recibidas que se muestran en la figura 92 son recortadas en amplitud, perdiendo parte de la información y en consecuencia generando errores de recepción.

Repitiendo el mismo procedimiento para las distancias de 1, 1.5, 2 y 3 metros se determinaron las ganancias de referencia experimentales 15, 18, 22, 24 dB respectivamente, pudiendo aplicar la asignación de potencia sin exceder la potencia de recepción. Entonces la potencia

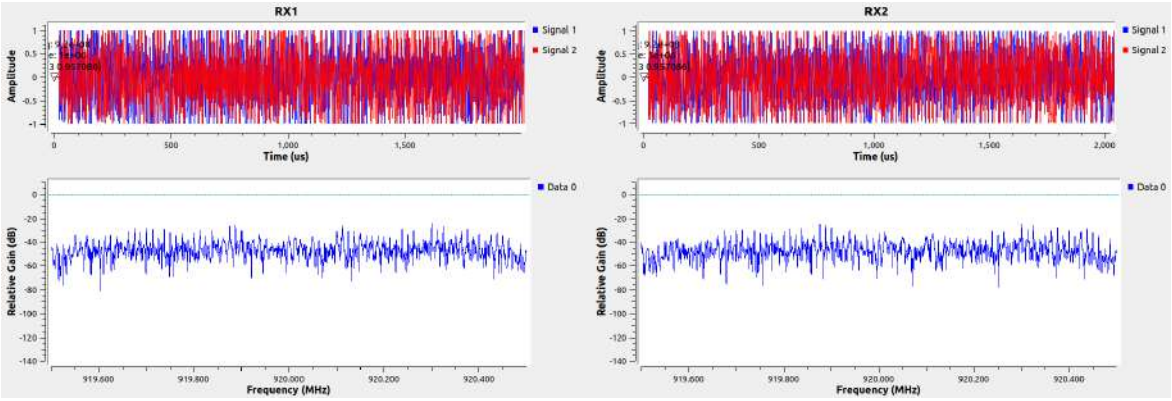


Figura 92: Señal recibida en tiempo y frecuencia con 20 dB de ganancia

asignada según la ganancia de referencia se ve en la figura 93.

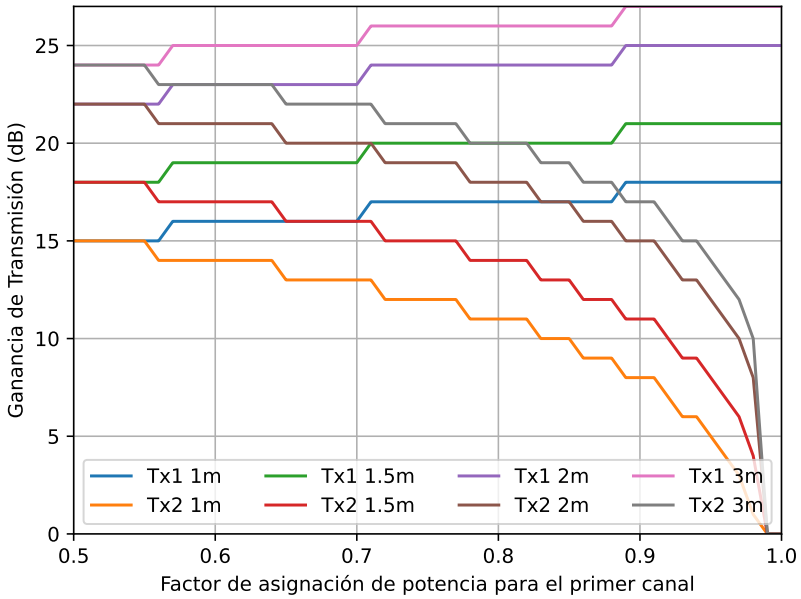


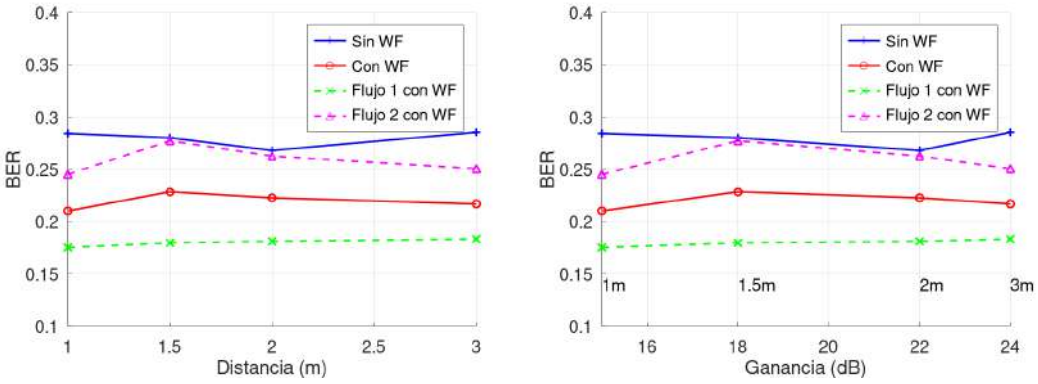
Figura 93: Factores de asignación de potencia calculada en cada prueba

Los resultados de la tasa de error de bits se muestran en la tabla 7 y en la figura 94, donde en promedio se obtiene una reducción del 25 % de la BER después de aplicar el algoritmo de *water-filling*.

El resultado gráfico de aplicar el algoritmo de *water-filling* en los canales se muestra en la figura 95, donde para a la mayoría de canales MIMO se estima una asignación de potencia del 70 % a 80 % para el primer transmisor y 20 % a 30 % para el segundo transmisor.

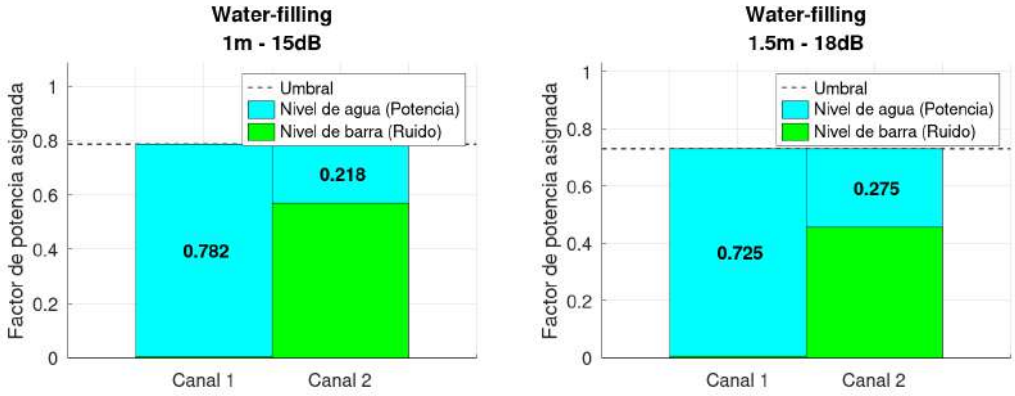
Distancia [m]	Ganancia TX [dB]	BER			
		Sin WF	Con WF	WF: Flujo 1	WF: Flujo 2
1	15	0.284	0.2101	0.1752	0.245
1.5	18	0.2799	0.2283	0.1796	0.277
2	22	0.2678	0.2225	0.181	0.2624
3	24	0.2851	0.2168	0.1832	0.25036

Tabla 7: Tasas de error de bit

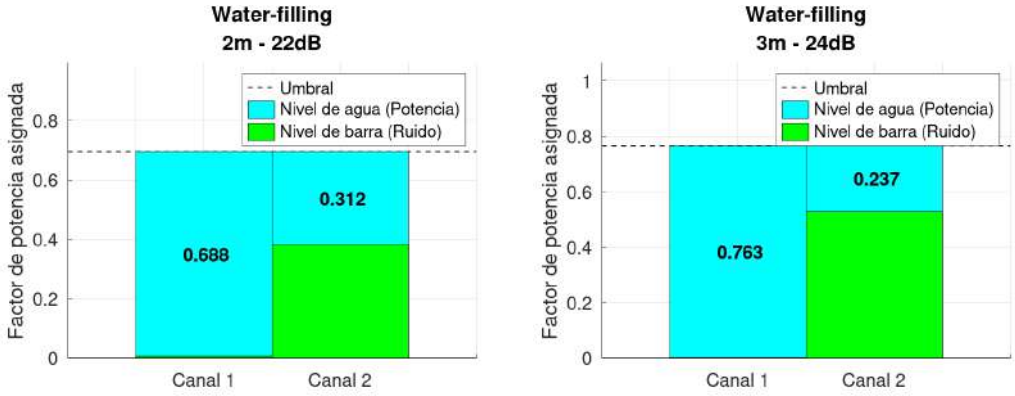


(a) Tasa de error de bit en base a la distancia (b) Tasa de error de bit en base a la ganancia

Figura 94: Gráficas de tasa de error de bit



(a) Nivel de agua para canal de 1m (b) Nivel de agua para canal de 1.5m



(c) Nivel de agua para canal de 2m (d) Nivel de agua para canal de 3m

Figura 95: *water-filling* para los canales inalámbricos en las pruebas

Discusión

- En la simulación con un canal básico se obtuvieron resultados favorables con la asignación de potencia óptima reduciendo la tasa de error aprovechando la descomposición del canal. En el primer escenario se obtienen factores idénticos en la salida de la terminal pero en el segundo escenario de simulación se obtienen factores similares mas no iguales; esto debido a los efectos de canal y en base a las tramas recepcionadas. Pero el bloque tiene una buena estimación de canal.
- El algoritmo de *water-filling* utilizado está programado para priorizar la asignación de potencia al primer transmisor. Si el canal posee una alta interferencia entonces toda la potencia sería asignada a un solo transmisor, así de esta manera el sistema en teoría asegura que solo el 50 % de la información sea correctamente recibida. Por lo que analizando individualmente la tasa de error en cada flujo de recepción, el primer flujo es el que tendrá una menor tasa de error comparado al segundo flujo.
- Se obtuvieron tasas de error de bit elevadas debido a las siguientes razones: Primero, en base al trabajo de [19], se utilizó otros modelos de radio USRP, banda de frecuencias más baja, software y ordenador diferente para desarrollar el prototipo de comunicación. Por lo que no fue posible replicar del todo los mismos resultados en simulación como en experimentación real y se tuvo que adecuar al entorno. Segundo, las técnicas utilizadas en los filtros no presentan suficiente robustez para recuperar la señal sin un número bajo de errores, por tanto se pueden tener pérdidas de paquetes OFDM al no detectarse correctamente.
- Las tasas de error fueron obtenidas a partir de muestras de archivos de más de 1000 Bytes, por lo que se obtienen tasas de error de bit no menor a 10^{-3}. Esto debido a que el diagrama de flujo en GNU Radio presenta una optimización parcial de como se ejecutan los algoritmos, puesto que se procesan muchos bloques de datos de señales en tiempo real lo que conlleva a una gran carga de procesamiento para el ordenador. Entonces, se puede presentar lo que es desbordamiento de datos, que es cuando no se pueden procesar todas las muestras lo suficientemente rápido, perdiendo rendimiento en el proceso y con la posibilidad de colapsar la ejecución del programa.
- Los factores de asignación resultantes de la estimación del canal, no mantienen los mismos valores a diferentes distancias. Puesto que se estiman diferentes canales MIMO. Sin embargo, en base a los resultados de los factores de asignación, estos indica que el canal a dicha frecuencia no presenta una alta interferencia como para asignar un porcentaje mayor del 90 % de la potencia a uno de los transmisores; aunque este puede ser limitado en la configuración del bloque de GNU Radio.

Conclusiones

- Se logró aplicar el algoritmo de *water-filling* en un prototipo de comunicación MIMO 2x2 para la asignación de potencia óptima en los canales inalámbricos.
- Para poder aplicar el algoritmo de *water-filling* a los sistemas MIMO, se debe realizar una descomposición paralela del canal MIMO, lo que permite determinar los parámetros de relación señal a ruido y poder efectuar una asignación de potencia óptima.
- Se diseñó e integró técnicas de procesamiento digital de señales para tecnología MIMO sobre plataforma de SDR, y se concluyó en el desarrollo de un sistema MIMO 2x2 que utiliza el estándar 802.11n con modulación QPSK.
- Los canales SISO resultantes de la descomposición del canal MIMO se manejaron independientemente con su propia asignación de potencia de acuerdo al algoritmo de *water-filling*.
- A partir de los resultados del algoritmo de *water-filling* se demuestra una mejora del 25 % de la tasa de error de bits de acuerdo a la tabla 7, resultando en un aumento de la capacidad y uso de la potencia de manera eficiente, en un sistema MIMO-OFDM 2x2.

Recomendaciones

- En sistemas Ubuntu los drivers del NI USRP 2950R son publicados periódicamente y pueden quedar obsoletos en sistemas recientes por lo que se recomienda utilizar conexión por 1Gb o 10Gb Ethernet, que a diferencia del kit de conexión PCI Express, no necesita driver y solo se maneja direcciones IP que facilitan la conexión, pero la velocidad de datos para sistemas mucho mas complejos se reduce, perdiendo rendimiento.
- Se recomienda utilizar algunos herramientas de calibración provistas por los drivers de UHD para cada modelo de USRP ya que no operan con la misma exactitud, estas herramientas pueden ayudar a minimizar el desbalance de las muestra IQ y la desviación de frecuencia del oscilador local con la sintonización para cualquier rango de frecuencias accesible por el dispositivo.
- Se recomienda utilizar versiones estables de GNU Radio en Ubuntu, ya que no siempre las más recientes son compatibles con los paquetes o librerías utilizadas, al menos en esta tesis, resultando en un esfuerzo mayor tratando de adaptarlos.
- Se recomienda utilizar métodos de sincronización y detección más robustos y que contrarresten los efectos de canal de manera más efectiva para una mejor recepción de las tramas.
- Se debe tener en cuenta la potencia de transmisión de la señal para un entorno donde no se supere el desvanecimiento y la potencia de recepción no se sature en un canal inalámbrico, así manteniendo la amplitud digital de las muestras de -1 a 1, por lo que se recomienda utilizar atenuadores de señal en los conectores de las antenas o hacer pruebas en distancias mayores.
- Para trabajar con Radio definida por software se debe medir la complejidad del proyecto debido a que la mayor parte del procesamiento depende del ordenador, por lo que los diagramas que poseen más complejidad requieren mayor capacidad de procesamiento.

Referencias

[1] Y. H. Hernández and F. M. Rizo, "Sistema mimo empleando antenas adaptativas," *Telemática*, vol. 12, no. 2, pp. 23–34, 2013.

[2] C. J. Pertuz, "Capacidad de canal y asignación de potencia con propagación rayleigh haciendo uso del algoritmo water-filling," 2014.

[3] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[4] K. Kumar and M. Singh, "Proposed water filling model in a mimo system," *International Journal of Emerging Technology and Advanced Engineering*, vol. 1, no. 2, pp. 127–131, 2011.

[5] E. Wineberger and O. Amrani, "Adaptive loading for mimo and mimo-ofdm channels," in *2006 IEEE 24th Convention of Electrical & Electronics Engineers in Israel*, pp. 403–407, IEEE, 2006.

[6] N. Kumar and D. Kaur, "Enhance the capacity of mimo wireless communication channel using svd and optimal power allocation algorithm," *International Journal of Electronics and Telecommunications*, vol. 65, no. 1, pp. 71–78, 2019.

[7] L. Grira and R. Bouallegue, "Using water filling technique and svd decomposition for cooperative node selection in wsn," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 149–153, IEEE, 2020.

[8] H. Gurdasani *et al.*, "Channel capacity enhancement of mimo system using water-filling algorithm," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 12, pp. 192–201, 2021.

[9] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *MIMO-OFDM wireless communications with MATLAB*. John Wiley & Sons, 2010.

[10] Keysight, "802.11n HT OFDM Overview." https://rfmw.em.keysight.com/, 2022. Último acceso: 22.03.23.

[11] A. Caiza and D. Fernando, "Estudio comparativo teórico-práctico entre los estándares wlan 802.11n y 802.11ac," B.S. thesis, Sangolquí, 2018.

[12] M. Gast, *802.11 n: a survival guide*. O'Reilly Media, Inc., 2012.

[13] H. Kaur, M. Khosla, and R. Sarin, "Channel estimation in mimo-ofdm system: a review," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 974–980, IEEE, 2018.

[14] A. Khlifi and R. Bouallegue, "Performance analysis of ls and lmmse channel estimation techniques for lte downlink systems," *arXiv preprint arXiv:1111.1666*, 2011.

[15] J. Á. A. Fundora and N. A. Torres, "Rds (radio definido por software). consideraciones para su implementación de hardware," *Telemática*, vol. 12, no. 2, pp. 56–68, 2013.

[16] Ettus Research, "Usrp products." https://www.ettus.com/, 2013. Último acceso: 15.10.22.

[17] Ubuntu, "What is Ubuntu?." https://help.ubuntu.com, 2020. Último acceso: 25.10.22.

[18] "GNU Radio." `https://www.gnuradio.org/`, 2020. Último acceso: 30.10.22.

[19] 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.

[20] S. Wunsch, "GNU Radio - Radar toolbox." https://github.com/kit-cel/gr-radar, 2022. Último acceso: 15.12.22.

Anexos

Anexo A

Diagrama de bloques del sistema de comunicación MIMO-OFDM 2x2 con USRP

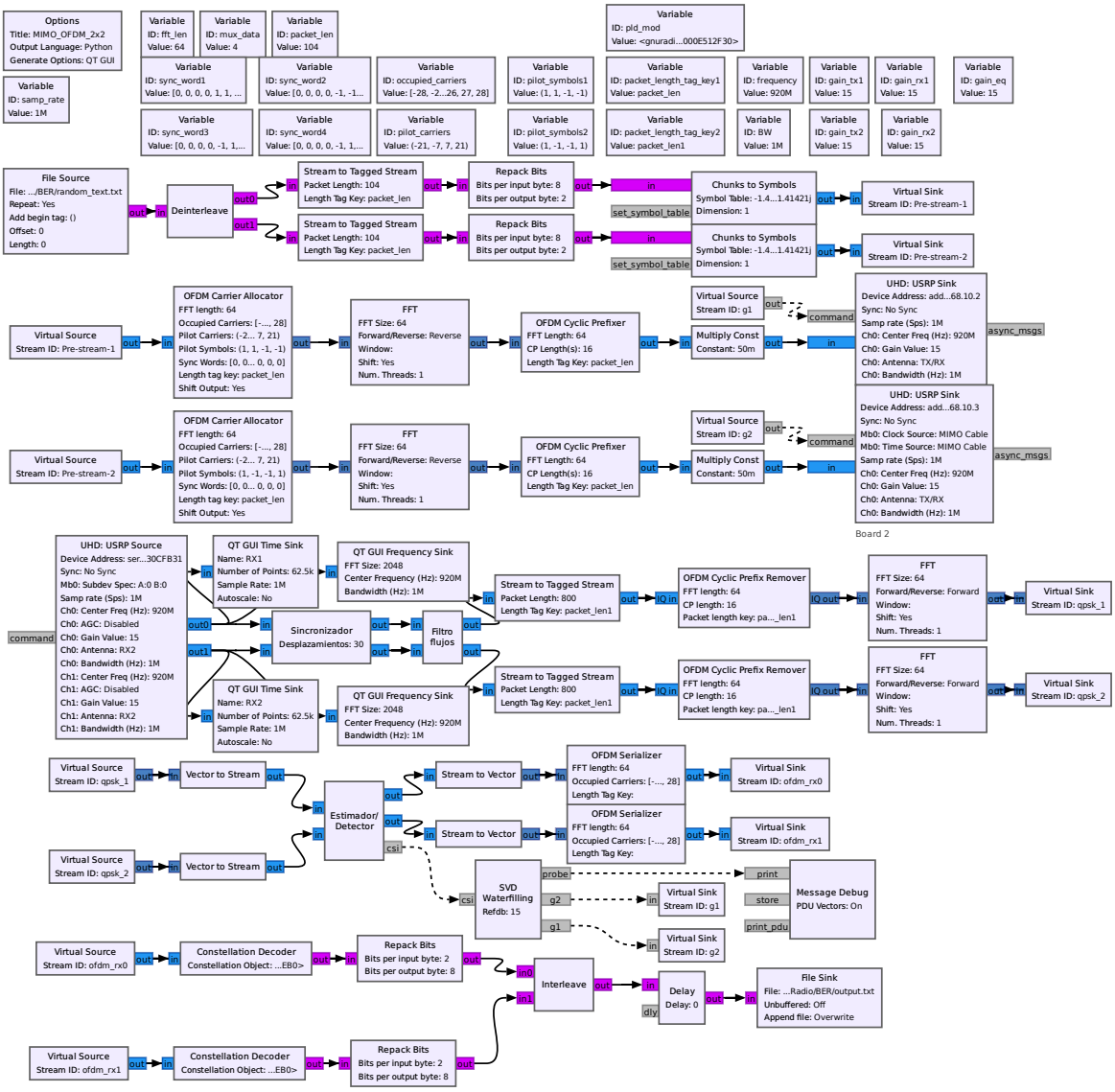


Figura 96: Diagrama de bloques de sistema de comunicación MIMO-OFDM 2x2

Anexo B

Diagrama de bloques del sistema de comunicación MIMO-OFDM 2x2 simulado en canal de ganancias

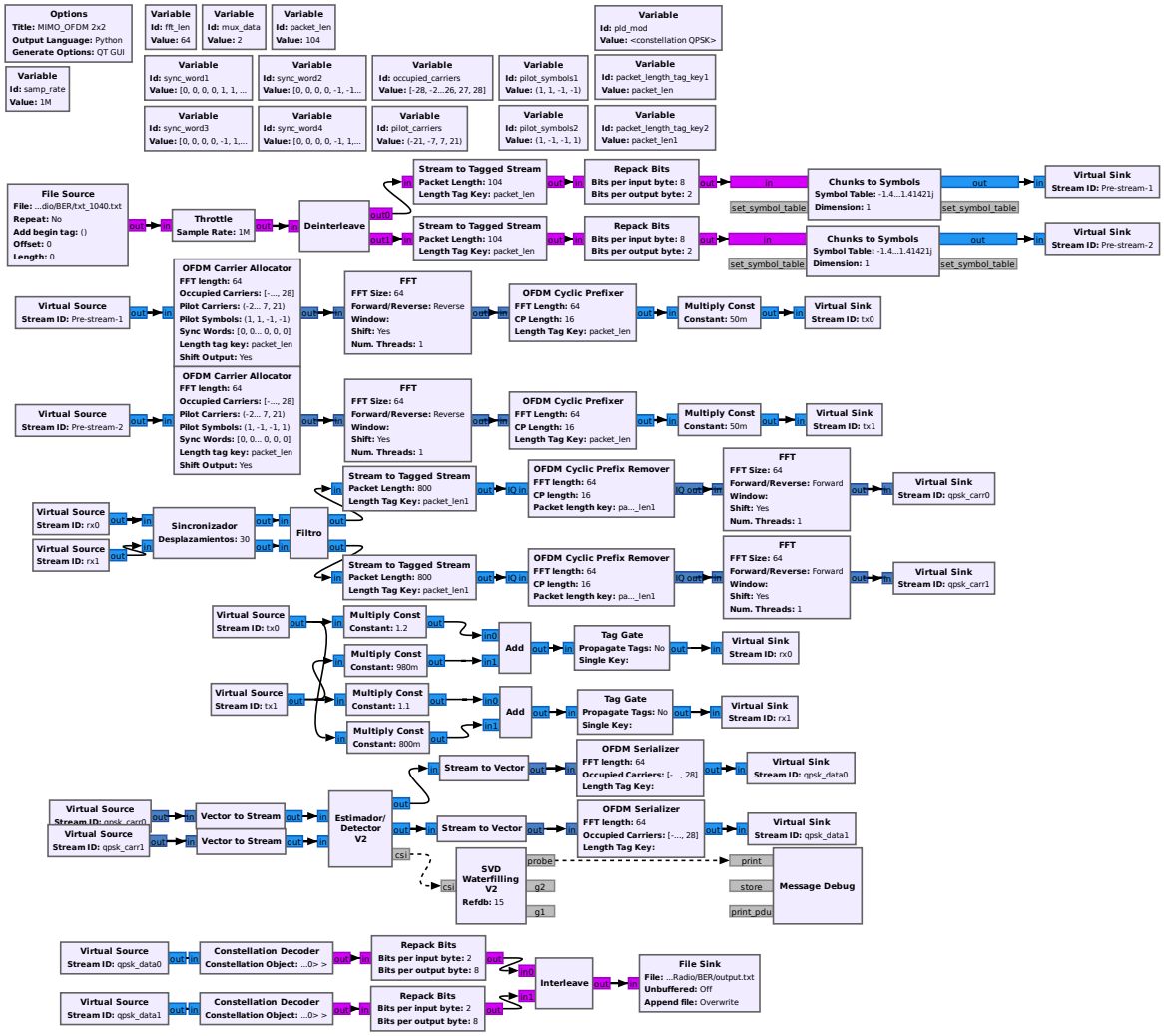


Figura 97: Diagrama de bloques de sistema de comunicación MIMO-OFDM 2x2

Anexo C

Diagrama de bloques del sistema de comunicación MIMO-OFDM 2x2 simulado en canal complejo con retardo

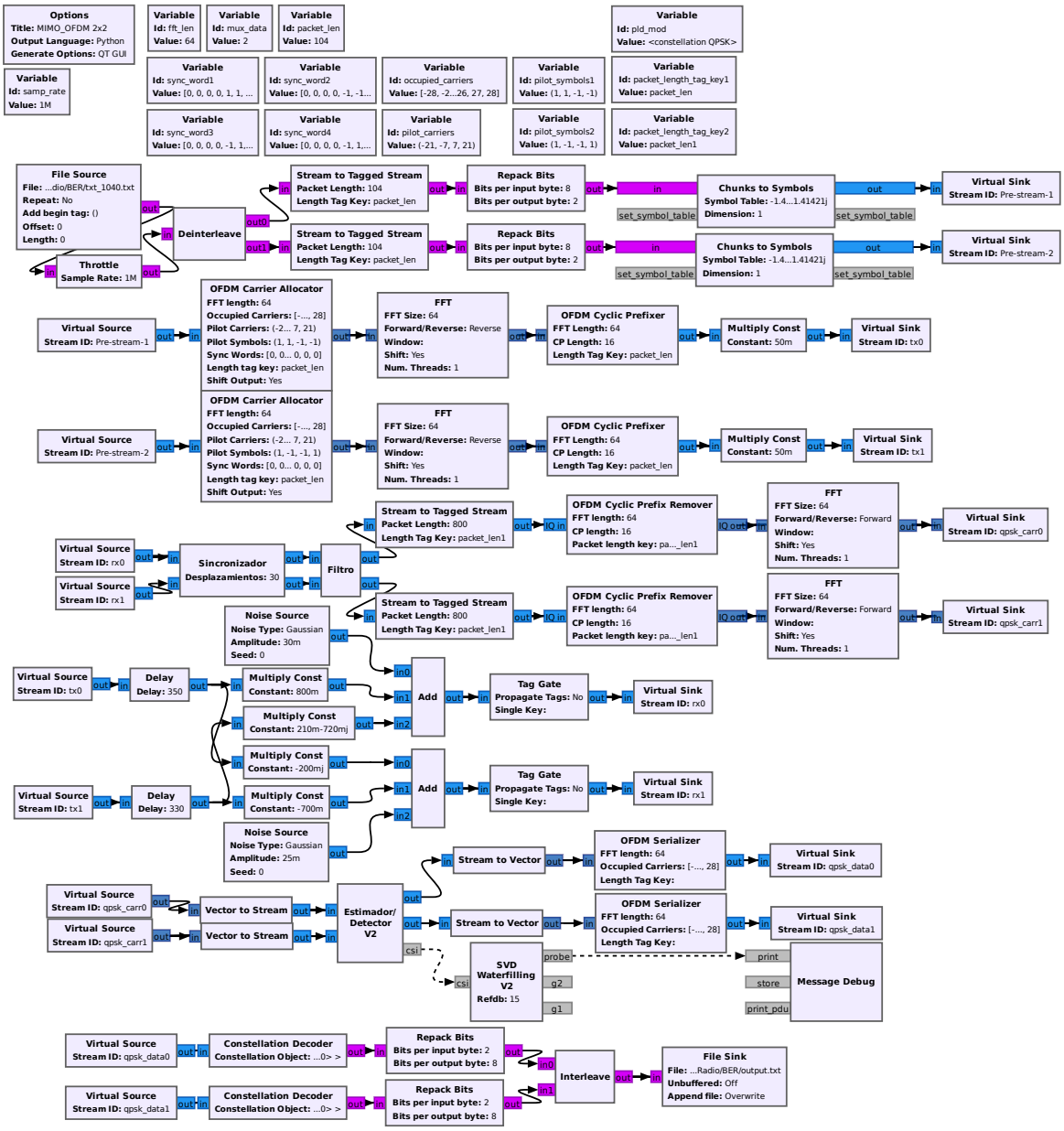


Figura 98: Diagrama de bloques de sistema de comunicación MIMO-OFDM 2x2

Anexo D

Autovalor y Valores Singulares

Un *eigenvalor* o *autovalor* y *eigenvector* o *autovector* de una matriz cuadrada A son un escalar λ y un vector x diferente de cero tal que:

$$Ax = \lambda x$$

Un *valor singular* y par de *vectores singulares* de una matriz cuadrada o rectangular A son escalares positivos σ y 2 vectores diferentes de cero u y v tal que:

$$Av = \sigma u,$$
$$A^H u = \sigma v$$

El exponente en A^H representa la transpuesta hermitiana y denota la compleja conjugada transpuesta de una matriz compleja. La ecuación característica o polinomio característico con lo que definimos los autovalores de A es:

$$\det(A - \lambda I) = 0$$

El grado del polinomio es del orden de la matriz, esto implica que una matriz de orden $n \times n$ tiene n autovalores. Sean $\lambda_1, \lambda_2, \ldots, \lambda_n$ los autovalores de la matriz A , y sean $x_1, x_2, \ldots, x_n$ un conjunto de autovectores correspondientes, y sea Λ una matriz diagonal de $n \times n$ de entrada λ_j, y sea X la matriz $n \times n$ cuya j-ésima columna es x_j. Entonces:

$$AX = X\Lambda$$

Asumiendo que los autovectores son linealmente independientes entonces X^{-1} existe, y con X siendo una matriz no singular, se tiene descomposición de autovectores en la forma:

$$A = X\Lambda X^{-1}$$

En caso los autovectores de A no sean linealmente independientes, entonces dicha descomposición no existe. Para cualquier matriz no singular T entonces se tiene:

$$A = TBT^{-1}$$

Que es una transformación de similitud, entonces se dice que A y B son similares. Si $Ax = \lambda x$ y $x = Ty$, entonces $By = \lambda y$, en otras palabras, una transformación de similitud preserva los autovalores. La descomposición de autovalores es un intento de encontrar transformaciones de similitud a una forma diagonal. Escrito en términos de matriz, la definición para valores

singulares y vectores singulares es:

$$AV = U\Sigma$$

Aquí Σ es una matriz diagonal del mismo tamaño que A. Resulta que los vectores singulares siempre pueden se elegidos para que sean perpendiculares el uno al otro, por lo cual las matrices U y V, cuyas columnas son los vectores singulares normalizados, satisfacen $U^H U = I$ y $V^H V = I$. En otras palabras, U y V son ortogonales si son reales, y unitarias en caso sean complejas. Consecuentemente:

$$A = U\Sigma V^H$$

Que es conocida como la descomposición de valores singulares de la matriz A.

Anexo E

Uso de GNU Radio Companion

Se dará a conocer como se utiliza GRC (GNU Radio Companion) en el sistema operativo Ubuntu. GNU Radio Companion es un editor visual para crear diagramas de bloques para procesamiento digital de señales.

Inicio de GNU Radio Companion

Para abrir GRC primero se abre una terminal en Ubuntu y luego se introduce el comando:

```
1 $ gnuradio-companion
```

Segmento de código 23: Inicio de GRC

Luego aparecerá la interfaz de GRC como se muestra en la figura 99. Esta interfaz posee distintas secciones para trabajar, primero se tiene la barra de herramientas con botones para abrir, guardar, ejecutar el programa, después se tiene la librería de bloques que contiene los bloques de procesamiento, luego está la ventana de variables donde se visualizan los valores y parámetros utilizados en los bloques, y finalmente una terminal integrada donde se muestran las salidas de algún comando o errores de compilación del programa.

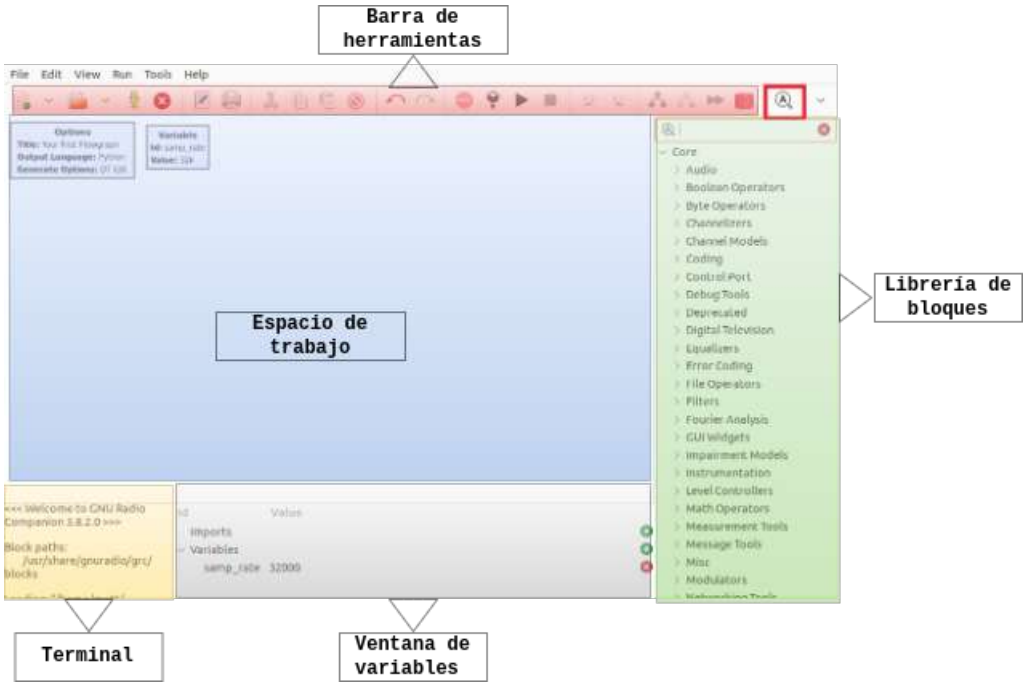


Figura 99: Interfaz de GRC

Para configurar las opciones del diagrama inicial se hace clic primario doble en el bloque *Options* que aparece inicialmente en cada inicio del programa, donde se mostrará los diversos parámetros de entrada que tiene, tal como son el titulo del programa, la descripción del mismo y el tipo de ejecución. Se procede a modificar sus propiedades, el parámetro Id de

espacio gris es el nombre con el cual se generará el programa y debe ser distinto del nombre de cualquier bloque ya existente, los espacios de color morado son entradas de texto para títulos, autores y descripciones adicionales como se muestra en la figura 100. Para guardar la configuración del bloque se presiona el botón OK.

Para guardar el diagrama se va la pestaña File y se presiona el botón “Save” como se muestra en la figura 101. Y finalmente se guarda con nombre de terminación .grc tal como se muestra en la figura 102.

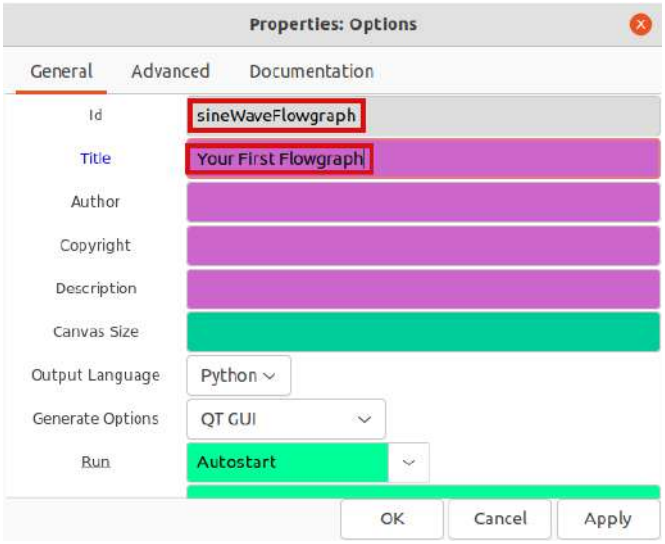


Figura 100: Nombrando el diagrama de bloques

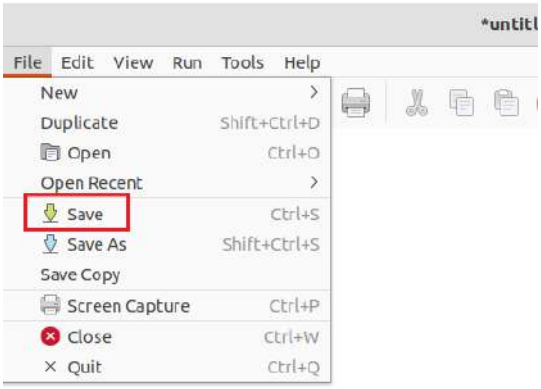


Figura 101: Guardado de archivo

Figura 102: Guardado de archivo .grc

Añadiendo bloques

Para añadir bloques al espacio de trabajo se utiliza el buscador en la librería haciendo uso del icono de la lupa de la barra de herramientas, y aparecerá un buscador encima de la librería de bloques y se escribe el nombre del bloque *Signal Source* que sirve como fuente de señal tal como se muestra en la figura 103. A partir del buscador se mantiene un Clic sobre el nombre del bloque y se arrastra o un doble clic para hacer aparecer el bloque en medio del espacio de trabajo, también se utiliza el bloque *Throttle*, que sirve de limitador del uso de recursos del ordenador cuando no se hace uso de hardware SDR, y para ver la etapa de la señal en una ventana de tiempo y otra en una ventana de frecuencia se añaden los bloques de QT, que son *Time Sink* y *Frequency Sink*.

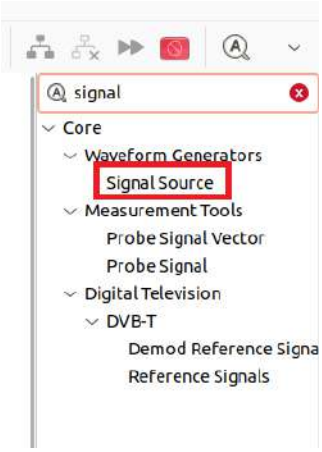


Figura 103: Librería de bloques

Para conectar los bloques se debe tener en cuenta el tipo de dato de salida y entrada, estos se distinguen por el color de los puertos en los bloques y se clasifican en: complejos (azules), flotantes (naranjas), enteros (verde), bits (morado) y mensajes asíncronos (gris) como se muestra en la figura 104.

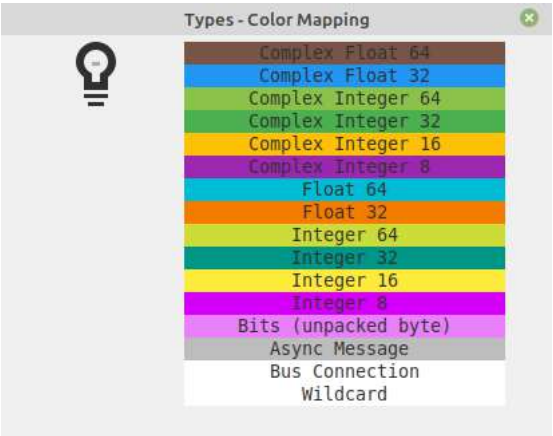


Figura 104: Tipos de datos en GRC

Luego se procede a conectar los bloques tal como se ve en la figura 105.

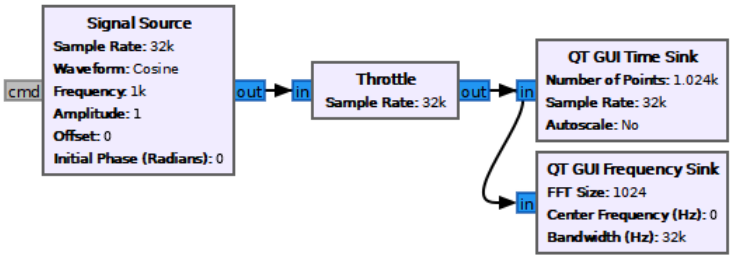


Figura 105: Conexión entre bloques

Para ejecutar el diagrama de bloques se utiliza el botón Play ubicado en la barra de herramientas.

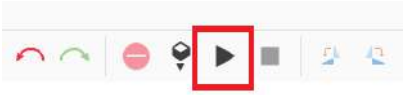


Figura 106: Ejecutando programa

Se ejecutará el programa y aparecerán ventanas de tiempo y frecuencia de la señal generada como se muestra en la figura 107.

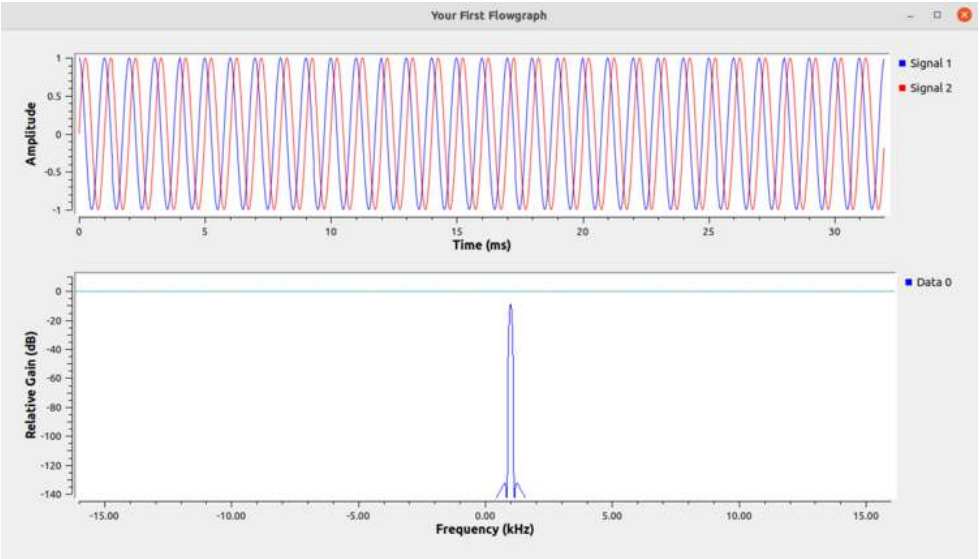


Figura 107: Ventanas de tiempo y frecuencia

Y finalmente donde se guardó el archivo .grc se generará un archivo en Python, el primero contiene la información para generar el diagrama en GRC y el archivo de Python contiene el código para ejecutar el diagrama de bloques.

Anexo F

Código python del bloque SVD Waterfilling

```
import numpy as np
from gnuradio import gr
import pmt

class blk(gr.sync_block):
    def __init__(self,RefdB=15):
        gr.sync_block.__init__(
            self,
            name='SVD\nWaterfilling',
            in_sig=[],
            out_sig=[]
        )
        self.Gain_ref = int(RefdB)
        self.pwrlimit = 0.9
        self.rho = 20
        self.message_port_register_in(pmt.intern("csi"))
        self.message_port_register_out(pmt.intern("g1"))
        self.message_port_register_out(pmt.intern("g2"))
        self.message_port_register_out(pmt.intern('probe'))
        self.set_msg_handler(pmt.intern("csi"), self.msg_handler)

    def gammaZero(self, gammas):
        c = 0
        i = len(gammas)
        sum = 1
        while c < len(gammas):
            if gammas[c] != 0:
                sum = sum + 1/gammas[c]
            else:
                i -= 1
            c += 1
        return i/sum

    def checkGamma(self, gamma_zero, gammas):
        redo = False
        for index, gamma in enumerate(gammas):
            if gamma < gamma_zero:
                gammas[index] = 0
                redo = True
        return [gamma for gamma in gammas], redo

    def pwrFactors(self, gammas, gamma_zero, n):
        pwrs = np.zeros(n)
        for index, gamma in enumerate(gammas):
            if gamma != 0:
                pwrs[index] = np.round(1/gamma_zero - 1/gamma, 3)
        return pwrs

    def WattsTodBm(self,power):
        return 10*np.log10(power)+30

    def dBmToWatts(self,power):
        return 10**((power-30)/10)
```

```
54
55      def msg_handler(self, msg):
56          pmt_msg = pmt.to_python(msg)
57          chans = []
58          for chan_msg in pmt_msg:
59              Hm = np.array([[abs(chan_msg[0]), abs(chan_msg[1])],
60                             [abs(chan_msg[2]), abs(chan_msg[3])]])
61              chans.append(Hm)
62          chans = np.asarray(chans)  # matrices H en magnitud
63          # Aplicando water-filling a matrices de canal
64          pwrs_vec = []
65          for Hm in chans:
66              rows, cols = Hm.shape
67              U, D, V = np.linalg.svd(Hm)
68              gammas = [(sigma**2)*self.rho for sigma in D]
69              # Gamma cero
70              gamma_zero = self.gammaZero(gammas)
71              # Verificando incoherencia
72              gammas, new_gamma_flag = self.checkGamma(gamma_zero,
    gammas)
73              # nuevo valor de gamma cero
74              if new_gamma_flag:
75                  gamma_zero = self.gammaZero(gammas)
76              # Umbral
77              mu = 1/gamma_zero
78              # factor de asignacion de potencia
79              factors = self.pwrFactors(gammas, gamma_zero, rows)
80              pwrs_vec.append([mu,factors[0],factors[1]])
81          pwrs_vec = np.asarray(pwrs_vec)
82
83          mu = 0
84          avr0 = 0
85          avr1 = 0
86          c = 0
87          for factors in pwrs_vec:
88              if factors[1] <= self.pwrlimit:
89                  mu = mu + factors[0]
90                  avr0 = avr0 + factors[1]
91                  avr1 = avr1 + factors[2]
92                  c = c + 1
93              else:
94                  mu = mu + self.pwrlimit
95                  avr0 = avr0 + self.pwrlimit
96                  avr1 = avr1 + (1 - self.pwrlimit)
97                  c = c + 1
98          mu = mu/c
99          avr0 = avr0/c
100         avr1 = avr1/c
101
102         # Generacion de mensaje USRP
103         Pt_max_dBm = 20 # [dBm] potencia maxima en cada canal
104         Gain_ref = self.Gain_ref # [dB] Referencia de 50% de potencia
     max = 28 dB para 50 mW
105
106         gain_n210 = [i for i in range(0,31+1)] # Rango de ganancias
    USRP 2920
107         Pt_dBm_0dB = Pt_max_dBm - gain_n210[len(gain_n210)-1] # dBm a
     0 dB de ganancia
```

```
108
109         Power_ref = self.dBmToWatts(Pt_dBm_0dB + Gain_ref) # Watts
110         Pt_max = 2*Power_ref  # [mW] Restriccion de potencia para
    ambos USRP
111
112         pwr_factor = avr0     # Factor de potencia (>0.5)
113         if pwr_factor == 1:   # En caso se asigne toda la potencia al
     primer canal
114             P1 = Pt_max
115             P2_dBm = Pt_dBm_0dB
116         else:
117             P1 = pwr_factor*Pt_max
118             P2 = (1-pwr_factor)*Pt_max
119             P2_dBm = self.WattsTodBm(P2)
120         P1_dBm = self.WattsTodBm(P1)
121
122         gain1 = np.round(P1_dBm - Pt_dBm_0dB,0)
123         gain2 = np.round(P2_dBm - Pt_dBm_0dB,0)
124
125         # puerto "probe" factores/ganancias/umbrales
126         pmt_probe = pmt.to_pmt([mu,avr0,avr1])
127         #pmt_probe = pmt.to_pmt([gain1,gain2])
128         self.message_port_pub(pmt.intern("probe"),pmt_probe)
129
130         # Comandos de configuracion de ganancia para los USRP
131         ant_msg = "TX/RX"
132         pmt_g1 = pmt.to_pmt({'antenna': ant_msg, 'gain': gain1, 'chan
    ': 0})
133         pmt_g2 = pmt.to_pmt({'antenna': ant_msg, 'gain': gain2, 'chan
    ': 0})
134         self.message_port_pub(pmt.intern("g1"),pmt_g1)
135         self.message_port_pub(pmt.intern("g2"),pmt_g2)
```

Segmento de código 24: Código del bloque SVD Waterfilling

Anexo G

Código Octave para gráficas de water-filling

```
clf; clearvars;
hold on;
grc_out = [0.786864 0.781804 0.218196]

threshold =grc_out(1);
bar1 = threshold-grc_out(2)
bar2 = threshold-grc_out(3)

% Umbral
plot (0:3, [1 1 1 1]*threshold, 'k--',"linewidth", 1);

% Nivel de agua
y = [1,1]*threshold;
h = bar(y, "w", 1);
set (h, "facecolor", "c");

% Barra de ruido
y = [bar1 bar2];
h = bar (y, "w", 1);
set (h, "facecolor", "g");

% Leyenda
legend("Umbral","Nivel de agua (Potencia)","Barra (Ruido)","location"
    ,"northeast");

% Config axis
axis ([0 3 0 threshold+0.45])

% Titulos
title ({"Water-filling",'Simulacion en canal complejo'},"fontsize",
    24);
ylabel("Factor de potencia","fontsize", 20);

% Nombres eje X
labels = [' ';' ';'Canal 1';' '; 'Canal 2';' ';' '];
set(gca, 'XTickLabel', labels);

% Tamano letra de los ejes
set(gca, 'FontSize', 20)

% Valores de ganancia
text(1-0.15,(threshold+bar1)/2,mat2str(grc_out(2),3),'Color', 'k',
    ...         % Red text
      'FontWeight', 'bold','FontSize', 20)
text(2-0.15,(threshold+bar2)/2,mat2str(grc_out(3),3),'Color', 'k',
    ...         % Red text
      'FontWeight', 'bold','FontSize', 20)
hold off;
print("waterbars.png", "-F:14")
```

Segmento de código 25: Gráficas de water filling