Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DEL LABORATORIO DE COMUNICACIONES 3, IMPLEMENTANDO EL *SOFTWARE* LIBRE "GNU RADIO", PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Elizabeth Micaela Apolonia Lux Guarcas
Asesorado por el Ing. Carlos Eduardo Guzmán Salazar

Guatemala, septiembre de 2023

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DEL LABORATORIO DE COMUNICACIONES 3, IMPLEMENTANDO EL *SOFTWARE* LIBRE "GNU RADIO", PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

ELIZABETH MICAELA APOLONIA LUX GUARCAS
ASESORADO POR EL ING. CARLOS EDUARDO GUZMÁN SALAZAR

AL CONFERÍRSELE EL TÍTULO DE

INGENIERA ELECTRÓNICA

GUATEMALA, SEPTIEMBRE DE 2023

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. José Francisco Gómez Rivera (a.i.)
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Ing. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz Gonzáles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADOR	Ing. Marvin Marino Hernández Fernández
EXAMINADOR	Ing. José Anibal Silva de los Ángeles
EXAMINADOR	Ing. Helmunt Federico Chicol Cabrera
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DEL LABORATORIO DE COMUNICACIONES 3, IMPLEMENTANDO EL *SOFTWARE* LIBRE "GNU RADIO", PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 23 de enero de 2020.

Elizabeth Micaela Apolonia Lux Guarcas

Guatemala, 1 de marzo de 2023

Ingeniero
Julio César Solares Peñate
Coordinador Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado ingeniero Solares:

Hago de su conocimiento que he concluido la revisión del trabajo de graduación de la estudiante **Elizabeth Micaela Apolonia Lux Guarcas**, titulado:

DISEÑO DEL LABORATORIO DE COMUNICACIONES 3, IMPLEMENTANDO EL SOFTWARE LIBRE "GNU RADIO", PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Considero que la estudiante Lux Guarcas ha elaborado su trabajo de acuerdo con el propósito que propuso en el protocolo que presento en su momento a la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, para la aprobación de su trabajo de graduación. Por lo que, **APRUEBO** el trabajo referido.

Quedo en la mejor disposición de ampliar cualquier párrafo precedente.
Un cordial saludo.

Carlos Guzmán Salazar
ASESOR

CARLOS GUZMAN SALAZAR
Ingeniero Electricista
Col. No. 2762

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 8 de marzo de 2023

Señor director
Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Estimado Señor director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **DISEÑO DEL LABORATORIO DE COMUNICACIONES 3, IMPLEMENTANDO EL SOFTWARE LIBRE "GNU RADIO", PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** desarrollado por la estudiante **Elizabeth Micaela Apolonia Lux Guarcas**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

Ing. Julio César Solares Peñate
Coordinador de Electrónica

ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

REF. EIME 42.2023.

El director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del asesor, con el Visto Bueno del coordinador de área, al trabajo de Graduación de la estudiante: Elizabeth Micaela Apolonia Lux Guarcas "**DISEÑO DEL LABORATORIO DE COMUNICACIONES 3, IMPLEMENTANDO EL SOFTWARE LIBRE "GNU RADIO", PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA"**, procede a la autorización correspondiente.

Ing. Armando Alonso Rivera Carrillo

Guatemala, 29 de agosto de 2023.

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.424.2023

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **DISEÑO DEL LABORATORIO DE COMUNICACIONES 3, IMPLEMENTANDO EL SOFTWARE LIBRE "GNU RADIO", PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** presentado por: **Elizabeth Micaela Apolonia Lux Guarcas,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO a.i.
Facultad de Ingeniería

Ing. José Francisco Gómez Rivera

Decano a.i.

Guatemala, septiembre de 2023

JFGR/gaoc

ACTO QUE DEDICO A:

Dios	Porque si no fuera por él no lo hubiera logrado, a él le debo todo.
Mis padres	Juan Francisco Lux y Margarita Guarcas, por ser mi apoyo a lo largo de mi vida, por permanecer a mi lado en todo tiempo y así poder culminar esta meta.
Pastor	Fernando Locon, por sus oraciones, sus consejos y guía que son de fortaleza para seguir perseverando en todas las áreas de mi vida y así poder culminar esta etapa.
Mis hermanos	Israel, Eduardo y Erick Lux, por estar siempre apoyándonos en todo lo que se pueda.

AGRADECIMIENTOS A:

La Universidad de San Carlos de Guatemala	Por ser una casa de estudio donde se brindan herramientas para el crecimiento profesional.
Facultad de Ingeniería	Por brindarme conocimiento para seguir creciendo, mejorando y ser así una excelente profesional.
Dios	Por siempre estar a mi lado cuidando de mí, por su perfecto amor y su misericordia que me fortalecen a seguir adelante para alcanzar todo lo que él quiere para mí.
Mis padres	Juan Francisco Lux y Margarita Guarcas, por ser un ejemplo de perseverancia y esfuerzo. Por enseñarme amar y fortalecerme en Dios. Por su amor incondicional que fortalece y guían mi vida a dar lo mejor de mí y así poder alcanzar todas mis metas.
Pastor y hermanos de la iglesia	Fernando Locon, por su apoyo incondicional que me fortalecen a seguir adelante.

Compañeros y amigos de la universidad	Abigail de la Cruz, Edna Coloch, María José García y con todos los que compartí en algún proyecto, clase o laboratorio. Por los desvelos y el aprender juntos.
Mi asesor	Ingeniero Carlos Guzmán, por sus conocimientos compartidos y su valioso tiempo al asesorarme para finalizar de este trabajo de graduación.

ÍNDICE GENERAL

INDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
r	Cantidad de muestras de retardo
g	Coeficiente
Hz	Hertz
KHz	Kilohertz
MHz	Megahertz
µseg	Microsegundos

GLOSARIO

AM	Amplitud modulada, técnica de transmisión que funciona mediante la variación de la amplitud de la señal transmitida en relación con la información que se envía.
BW	Ancho de banda, se mide como la cantidad de datos que se pueden transferir entre dos puntos de una red en un tiempo específico.
Demodulador	Separa la señal de modulación con la frecuencia portadora.
DAB	Siglas en inglés para Digital Audio Broadcasting, que trata de radiodifusión de audio digital.
DVB-T	Siglas en inglés para Digital Video Broadcasting – Terrestrial, es el estándar para la transmisión digital terrestre.
EEPROM	Siglas en inglés para Electrically Erasable Programmable Read Only Memory, es un tipo de memoria que puede ser programada, borrada y reprogramada.

FM	Frecuencia Modulada, permite transmitir información a través de una onda portadora variando su frecuencia.
Hardware	Componentes materiales y físicos de un dispositivo.
Software	Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas.
Ubuntu	Distribución de código abierto basado en Debian.
USB	Siglas en inglés para universal serial bus, estándar que define los cables, conectores y protocolos usados en un bus serial para conectar dos dispositivos.
UHF	Siglas en inglés para Ultra High Frequency, es una banda del espectro electromagnético que ocupa el rango de frecuencias de 300 megahercios a 3 gigahercios.
VHF	Siglas en inglés para Very High Frequency, comprende a la banda del espectro electromagnético que está entre 30 y 300 megahercios.

RESUMEN

Este trabajo de graduación desarrolla el material teórico y práctico para trabajar con el *software* libre GNU Radio y el *hardware* receptor DVB-T en el laboratorio de Comunicaciones 3. Se presenta información de cómo implementar bloques que se encuentran por *default* en este *software*, así como información para crear bloques personalizados que realizaran la acción que se les programe.

En el primer capítulo se proporciona información general sobre la comunicación digital.

En el segundo capítulo se desarrolla la base teórica de lo que es el *software* libre GNU Radio y se especifican las características de bloques que lo componen.

En el tercer capítulo se da una guía específica de lo que se debe realizar para crear bloques personalizados y que estos puedan ser utilizados en la interfaz de GNU Radio.

En el cuarto capítulo se proporciona información de las características del funcionamiento general de la radio FM y la televisión, así como la frecuencia portadora de audio y señal de video de canales de televisión, información que será útil al momento de realizar las practicas.

En el quinto capítulo se proporciona información de las características y costo del *hardware* receptor DVB-T que es utilizado juntamente con el *software* GNU Radio.

En el sexto capítulo se proponen las actividades a realizar en las distintas practicas del laboratorio, esto con el fin de que el estudiante ponga en práctica el material que se le proporcionó a lo largo de este trabajo de graduación.

OBJETIVOS

General

Describir las características de los bloques del *software* libre GNU Radio y creación de bloques personalizados, para ser aplicados a la comunicación digital del laboratorio de Comunicaciones 3.

Específicos

1. Brindar la descripción de las características de bloques del *software* libre GNU Radio, para que el estudiante pueda observarlos y aplicarlos en prácticas y proyecto de laboratorio.

2. Brindar la guía para la creación de bloques personalizados para el *software* libre GNU Radio, para que el estudiante pueda aplicarlos en prácticas y proyecto de laboratorio.

3. Proponer las actividades a realizar en las prácticas de laboratorio.

4. Presentar el costo y características del *hardware* que se puede implementar con el *software* libre GNU radio.

INTRODUCCIÓN

Hoy en día la comunicación digital para la transmisión y recepción de datos está avanzando día tras día. Es fundamental el conocimiento del avance en *hardware* como en *software* de la comunicación digital, para poder implementarlo correctamente en la vida real.

En la carrera de ingeniería electrónica se tiene el laboratorio de Comunicaciones 3. En este laboratorio se implementa el *software* libre GNU radio, el cual es utilizado para poner en práctica las distintas formas de transmisión y recepción de las distintas señales de información digital. Se pretende que el estudiante solucione problemas que puedan ser aplicados a necesidades reales tal como la transmisión de radio y televisión digital con un alto rendimiento mediante el uso simple y rápido de este *software.*

Por lo anterior se da al estudiante toda la información necesaria con respecto al *software* libre GNU radio, para facilitar y agilizar su aprendizaje. Se contempla desde la instalación del *software*, la descripción de las características de bloques utilizables dentro de la interfaz, creación de bloques personalizados y practicas guiadas. Esto le permitirá desenvolverse correctamente dentro del laboratorio, pero así mismo egresar con un conocimiento solido que podrá aplicar en la comunicación digital a lo largo de su vida profesional.

1. COMUNICACIÓN DIGITAL

El término comunicaciones digitales abarca un área extensa de técnicas de comunicaciones, como transmisión y recepción de radios digitales. Los sistemas de transmisión digital requieren de un elemento físico, entre transmisor y receptor, como un par de cables ya sea metálico, coaxial o un cable de fibra óptica, estas señales no cambian continuamente, sino que es trasmitida en paquetes discretos. La información no es inmediatamente interpretada, sino que primero debe ser decodificada por el receptor. En lo que respecta a la ingeniería de procesos, no existe limitación en cuanto al contenido de la señal.

1.1. Radio digital

Radio digital es la transmisión de portadoras analógicas moduladas, en forma digital, entre dos o más puntos de un sistema de comunicación digital. En los sistemas de radio digital, el medio de transmisión es el espacio libre o la atmósfera de la tierra.

La diferencia que existe entre un sistema de radio digital de un sistema de radio convencional AM, FM, es que, en un sistema de radio digital, las señales de modulación y demodulación son pulsos digitales, en lugar de formas de ondas analógicas, pero la radio digital si utiliza portadoras analógicas. Existen tres técnicas de modulación digital, las cuales son: modulación por desplazamiento de frecuencia, modulación por desplazamiento de fase y modulación de amplitud en cuadratura. (Electrónica Fácil, 2004)

1.2. Modulación

La modulación es la operación mediante la cual algunas características (amplitud, frecuencia, fase) de la onda llamada portadora, se modifican en función de otra llamada moduladora, la que contiene la información, para que esta última pueda ser transmitida. Con la modulación se modifica una señal con la finalidad de posibilitar el transporte de información a través de un canal de comunicación y recuperar la señal en su forma original.

1.2.1. Modulación por desplazamiento de frecuencia (FSK)

El FSK binario es una forma de modulación angular de amplitud constante, muy parecido a la modulación en frecuencia convencional, excepto que la señal modulante es un flujo de pulsos binarios que varían, en lugar de una forma de onda analógica que tiene varios niveles de voltaje.

1.2.2. Modulación por desplazamiento de fase (PSK)

El PSK es otra forma de modulación angular, modulación digital de amplitud constante. Es similar a la modulación de fase convencional, excepto que con PSK la señal de entrada es una señal digital binaria y son posibles un número limitado de fases de salida.

> Consiste un procedimiento de la onda portadora en función de un bit de datos (0, 1). El ángulo está asociado con un dato al ser transmitido y con una técnica de codificación usada para representar un *bit*. Se caracteriza porque la fase de la señal portadora representa cada símbolo de información de la señal moduladora, con un valor angular que el modulador elige entre un conjunto discreto de n valores posibles. Un

modulador PSK representa directamente la información mediante el valor absoluto de la fase de la señal modulada, valor que el demodulador obtiene al comparar la fase de esta con la fase de la portadora sin modular. (Modulación Digital, 2010, párr. 9)

1.2.3. Modulación de amplitud en cuadratura (QAM)

La modulación QAM, es una forma de modulación en donde la información digital está contenida, en la amplitud y en la fase de la portadora transmitida. El QAM de ocho (8-QAM), es una técnica de codificación M-ario, en donde M = 8. Lo que varía del 8-PSK, es la señal de salida de un modulador que es de 8-QAM no es una señal de amplitud constante.

1.3. Filtros digitales

Los filtros digitales son un sistema que, dependiendo de las variaciones de las señales de entrada en sus características de tiempo y amplitud, se realiza un procesamiento matemático, generalmente usando la FFT.

El filtrado digital es parte del procesado de trabajar con señal digital. Se le da la denominación de digital más por su funcionamiento interno no por el tiempo de señal entrante que se va a filtrar, así podríamos llamar filtro digital tanto a un filtro que realiza el procesado de la entrada de señales digitales como a otro que lo haga de la entrada de señales analógicas.

El procesamiento interno y la entrada del filtro serán de forma digital por lo que en ocasiones es necesaria dispositivos de conversión analógica-digital o digital-analógica para uso de filtros digitales si la señal es analógica. (Wikipedia, 2022)

1.3.1. Filtro pasa bajos

Este filtro transmite componentes de señales de baja frecuencia y bloquea las componentes de las señales de alta frecuencia. Cualquier señal con frecuencia mayor a la frecuencia de corte del filtro es bloqueada mientras que las señales con frecuencias menores a la frecuencia de corte son transmitidas.

1.3.2. Filtro pasa altos

Este filtro trasmite componentes de señales de alta frecuencia y bloquea las componentes de las señales de baja frecuencia. Cualquier señal con frecuencia menor a la frecuencia de corte del filtro es bloqueada mientras que las señales con frecuencias mayores a la frecuencia de corte son transmitidas.

1.4. Radio definido por *software*

Hasta mediados de los años 90 la flexibilidad de los equipos de comunicaciones era casi nula, pero con el desarrollo de los microprocesadores y la digitalización de las señales se comenzó a realizar el procesamiento de la señal digital mediante *software* y ya no por medio de *hardware*, sustituyendo así a los moduladores, demoduladores, filtros, amplificadores y otros dispositivos electrónicos. Es una tecnología cuyo principal objetivo es que el procesado de la señal se realice mediante *software* y que, además, dicho procesado se acerque lo máximo posible a la antena.

1.4.1. Ventajas del radio definido por *software* (SDR)

- Múltiples modos de operación: SDR puede ser usado para varias aplicaciones, las cuales podrían ser: AM, FM, OFDM, entre otros.
- Reconfiguración: SDR puede estar trabajando en una aplicación específica de operación, sin embargo, se puede configurar para que trabaje en otra aplicación lo cual dependerá de la configuración del *software*.
- Actualización de *over-the-air:* cuando la aplicación está en funcionamiento algunos parámetros pueden variar, estos pueden irse calibrando.
- Menor costo de desarrollo: con SDR se reducen muchos costos de adquisición de *hardware* para una aplicación específica y se puede realizar varias aplicaciones con el mismo *software.* (Lincango, 2018)

Algunos programas de radio definido por *software* son: GNU Radio, RTL-SDR y Dump 1090, cada uno con sus distintas aplicaciones y características específicas.

2. GNU RADIO

GNU Radio es un *software* que mediante un conjunto de bloques de procesamiento de señales digitales se puede desarrollar radio definido por *software*, el cual es libre y de código abierto. GNU radio puede ser utilizado en conjunto con *hardware* externo, como puede ser un DVB-T+DAB+FM, lo que da la posibilidad de realizar varias aplicaciones de *software* radio, si se utiliza sin el *hardware*, GNU radio puede ser utilizado como un entorno de simulación.

Para que GNU radio pueda realizar las conexiones entre bloques de procesamiento de señales utiliza el lenguaje de programación Python y utiliza c++ para el procesamiento de las señales. Esto permite que GNU Radio tenga aplicaciones de alto rendimiento, que son en tiempo real, con interfaces fáciles de usar. (Wikipedia, 2020)

La interfaz gráfica que simplifica la complejidad para utilizar GNU Radio es GNU Radio Companion, en este no es necesario escribir código, si no que se utilizan los bloques existentes ya dentro del *software*, aunque, si se quiere realizar aplicaciones y los bloques ya existentes no satisfacen las características requeridas, se puede realizar nuevos bloques escribiendo código en Python y c++. Esta opción de GNU Radio permite crear más aplicaciones con sus bloques específicos, utilizándolos en conjunto con los ya existentes.

La compatibilidad entre sistemas operáticos para el GNU Radio es amplia, este puede trabajar en el sistema operativo de Windows, Linux entre otros, ya que GNU Radio tiene pocos requerimientos. GNU Radio tiene compatibilidad

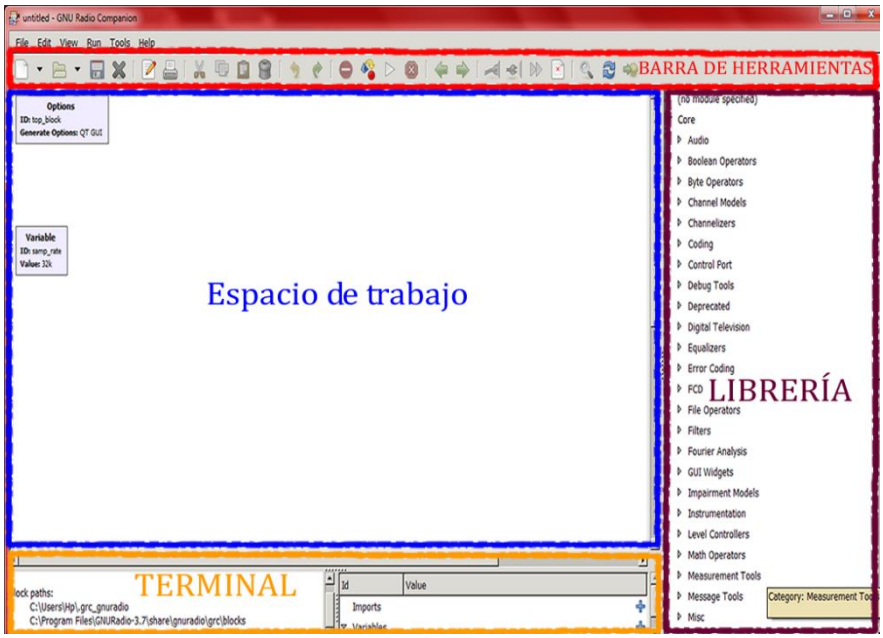
entre archivos contenedores que se generan para que puedan usarse en cualquier sistema.

2.1. Interfaz de GNU Radio

La interfaz de GNU Radio está conformada por:

- Barra de herramientas: en esta área se encuentran funciones que permiten evaluar, corregir y ejecutar el diagrama de bloques desarrollado para una aplicación específica y también se tiene control sobre el área de trabajo.
- Librería: en el área se encuentran disponibles todos los bloques que trae en si GNU Radio, así como los programados por el usuario. Estos bloques se dividen en diferentes categorías, dentro de la librería, los bloques diseñados por el usuario aparecen en una categoría sin identificar.
- Terminal: en esta área se imprimen los mensajes por parte de los bloques de GNU Radio.
- Espacio de trabajo: en el área se colocan los distintos bloques que se utilizaran para desarrollar una aplicación. Esta área está dimensionada para reducir la complejidad del algoritmo de nuestra aplicación. (Jaimes & Lazcano, 2019)

Figura 1. **Interfaz de GNU Radio**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

2.2. Generalidades de los bloques de GNU Radio

Se verá a continuación generalidades que componen a GNU Radio, entre estas están: herramientas, tipo de bloques y datos.

2.2.1. Herramienta de análisis

- Herramientas WX GUI: están basadas en el lenguaje C++ y no son compatible con herramientas QT GUI. Estas herramientas consumen más recursos para el procesamiento.
- Herramientas QT GUI: están basadas en Python, consumen menos recursos y permite desarrollar más herramientas visuales.

2.2.2. Clasificación general de los tipos de bloques

- Fuentes: pueden ser micrófonos, *hardware* y *ficheros*.
- Sumideros: pueden ser altavoces, visualizadores gráficos para ver la onda de la señal y la FFT.
- Bloques de procesado de señal: pueden ser filtros, amplificadores, operadores matemáticos o lógicos, moduladores o demoduladores.

2.2.3. Tipos de datos que manejan los distintos bloques

- *Byte*: 1 *byte* sería ocho bits
- *Short*: 2 *bytes* de datos
- *Int*: 4 *bytes* de datos
- *Float*: 4 *bytes* de datos, para números en punto flotante
- *Complex*: 8 *bytes* de datos, un float para cada componente

Estos bloques procesan los datos de forma continua desde su entrada hacia su salida y desempeñan únicamente una función, lo que hace a GNU Radio más flexible y que se adapte a las necesidades al tener la opción de crear bloques personalizados.

2.3. Bloques de GNU Radio

En seguida se describen detalladamente los parámetros a configurar de cada bloque a utilizar en las prácticas de laboratorio. Para observar los parámetros de cada bloque debe dar doble clic en el bloque a utilizar. (wiki.gnuradio, 2019)

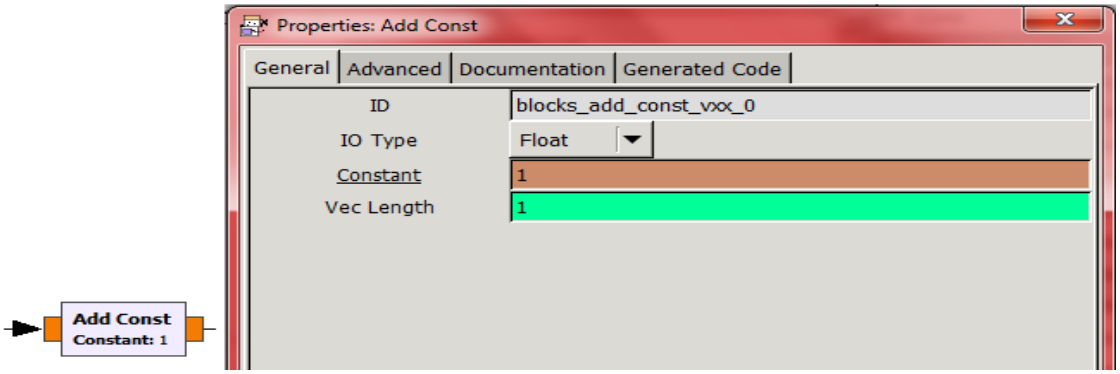
2.3.1. Add Const

Este bloque agrega un valor constante a cada elemento que pasa por él. Salida[m] = entrada[m] + valor constante

Parámetros del bloque Add Const, ver figura 2.

Constant: el valor a introducir debe ser del mismo tipo de datos que los puertos a su entrada y a su salida, puede ser entero, flotante, entre otras.

Figura 2. **Add Const**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

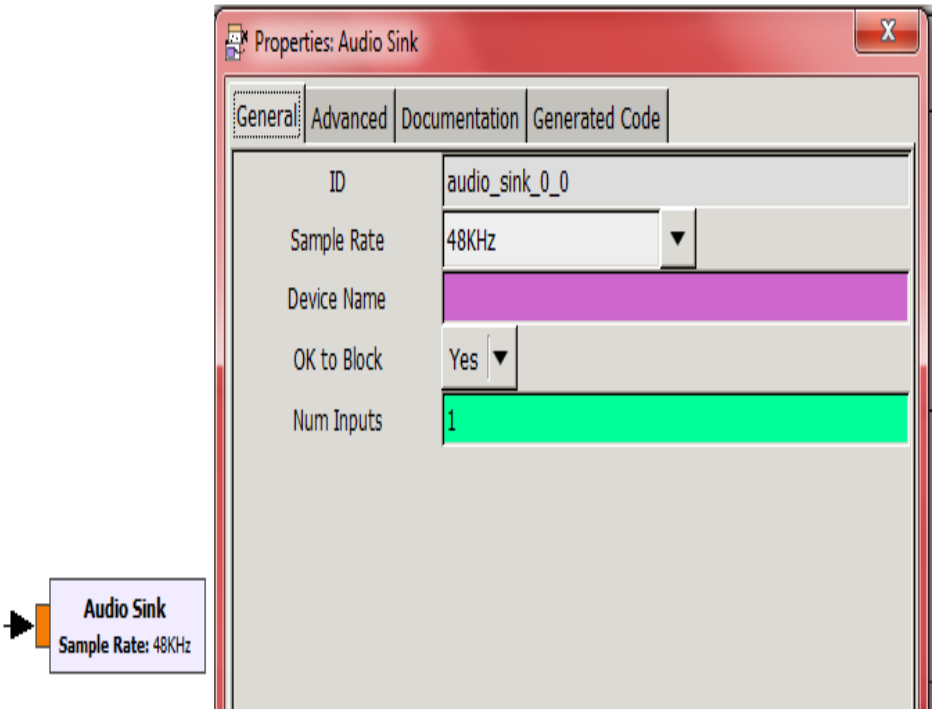
2.3.2. Audio Sink

Este bloque permite reproducir una señal a través de un dispositivo de audio.

Parámetros del bloque Audio Sink, ver figura 3.

- *Sample rate:* se debe escribir la frecuencia de muestreo que sea compatible con el *hardware* a utilizar, en aplicaciones típicas se establece a 48 KHz.
- *Device name*: si se utilizará un dispositivo de audio predeterminado, el nombre del dispositivo debe quedar en blanco, de lo contrario escriba el nombre específico de su dispositivo.
- *OK to block*: se debe tener activado de forma predeterminada cuando Audio Sink no está regulado por ningún otro bloque.
- *Num inputs*: si el receptor de audio tiene múltiples entradas, dependiendo del *hardware*, se puede configurar la entrada 2 para estéreo y 1 para mono.

Figura 3. **Audio Sink**

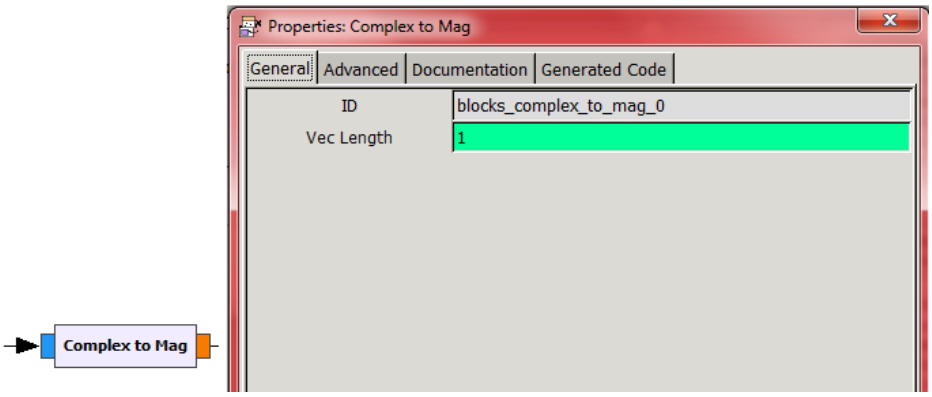


Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

2.3.3. Complex to Mag

Este bloque calcula la magnitud de las muestras complejas, también trabaja con vectores. No tiene parámetros, ver figura 4.

Figura 4. **Complex to Mag**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

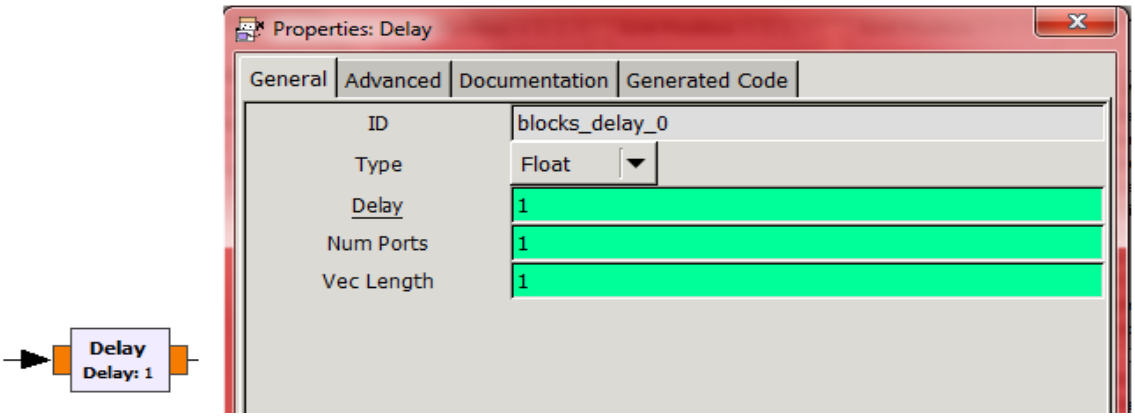
2.3.4. Delay

Este bloque retrasa la entrada por un cierto número de muestras. Si estos retrasos son positivos insertan cero elementos al comienzo de la secuencia, si son negativos descartan elementos de la transmisión. No se puede inicializar el bloque con un retraso negativo, ya que se genera un problema de causalidad con los *buffers*, es mejor colocar el retraso en la otra ruta.

Parámetros del bloque Delay, ver figura 5.

- Delay: número de muestras que se quiere retrasar.

Figura 5. **Delay**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

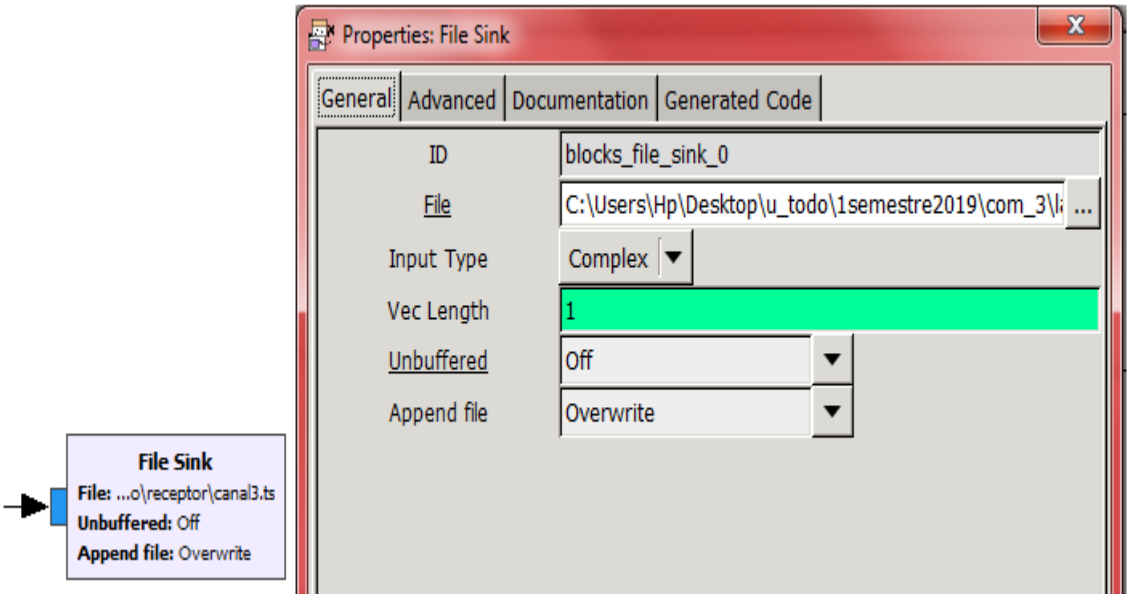
2.3.5. File Sink

Este bloque se utiliza para escribir una secuencia en un archivo binario. Cualquier programa que pueda leer archivos binarios, como Python, C o Matlab, pueden leer este archivo. Utilizando un origen de archivo se podría reproducir el archivo en GRC.

Parámetros del bloque File Sink, ver figura 6.

- *File*: se escribe el nombre del archivo para abrirlo y escribir la salida.
- *Unbuffered*: especifica si la salida está almacenada en la memoria intermedia. Cuando la salida no tiene *búfer*, los datos son transferidos al archivo cada vez que se llame a la función de trabajo, lo que hace que nuestra aplicación funcione lenta.
- *Append file*: se indica si se sobre escribe en el mismo archivo o se crea uno nuevo cada vez.

Figura 6. **File Sink**



Fuente: elaboración propia, realizado con GNU radio y Macromedia Fireworks.

2.3.6. Low Pass Filter

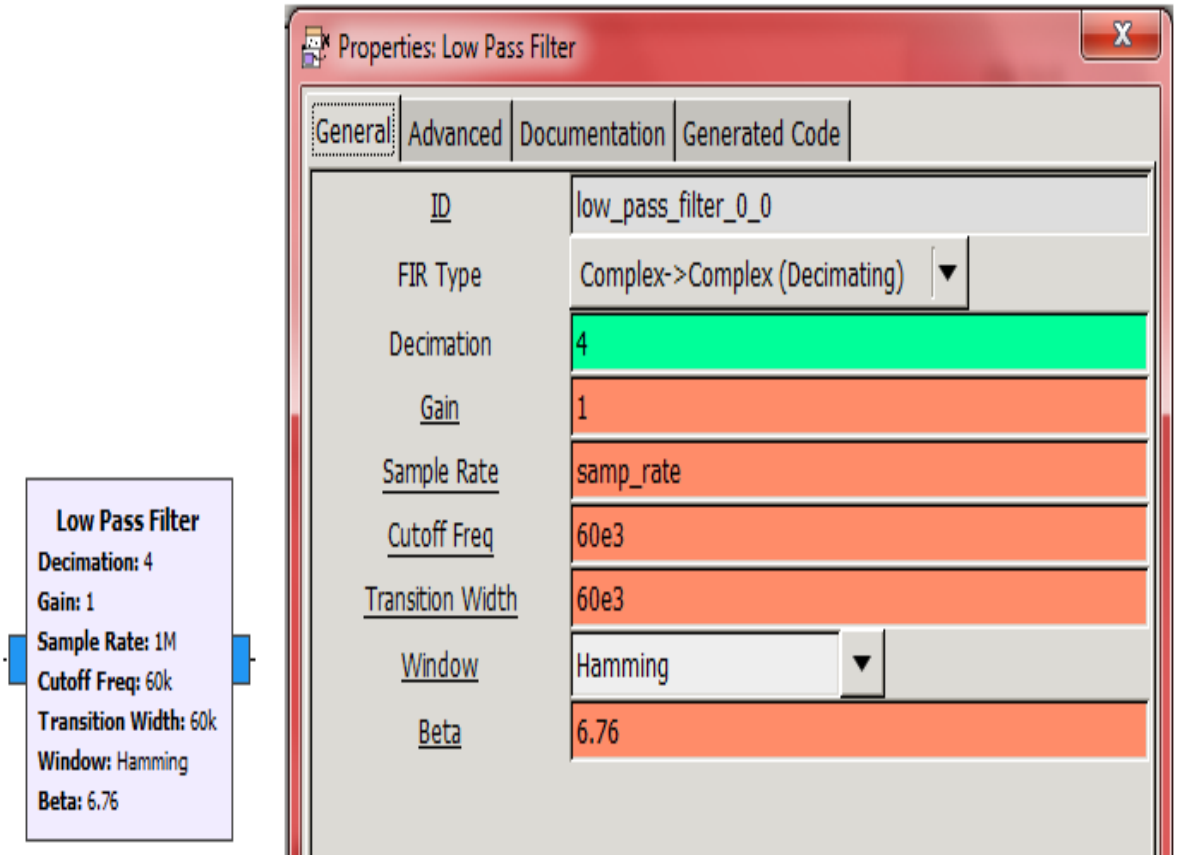
Este bloque dejar pasar solo las frecuencias menores al valor de nuestra frecuencia de corte. Es un filtro FIR diezmado.

Parámetros del bloque Low Pass Filter, ver figura 7.

- FIR *type*: se indica si tanto la entrada como la salida son reales o complejas.
- *Decimation*: es la tasa de diezmado del filtro, este no puede cambiar en tiempo real y debe ser número entero.
- *Gain*: se escribe el valor del factor de escala aplicado a la salida.
- *Sample rate*: se indica las frecuencias de muestro con la que se estará trabajando.

- *Cutoff freq*: es la frecuencia de corte en Hz.
- *Transition width*: es el ancho de banda, entre *stop-band* y *pass-band*, la frecuencia es en Hz.
- *Window*: tipo de ventana a usar.
- *Beta*: este parámetro solo se aplica a la ventana de Kaiser

Figura 7. **Low Pass Filter**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

2.3.7. Multiply

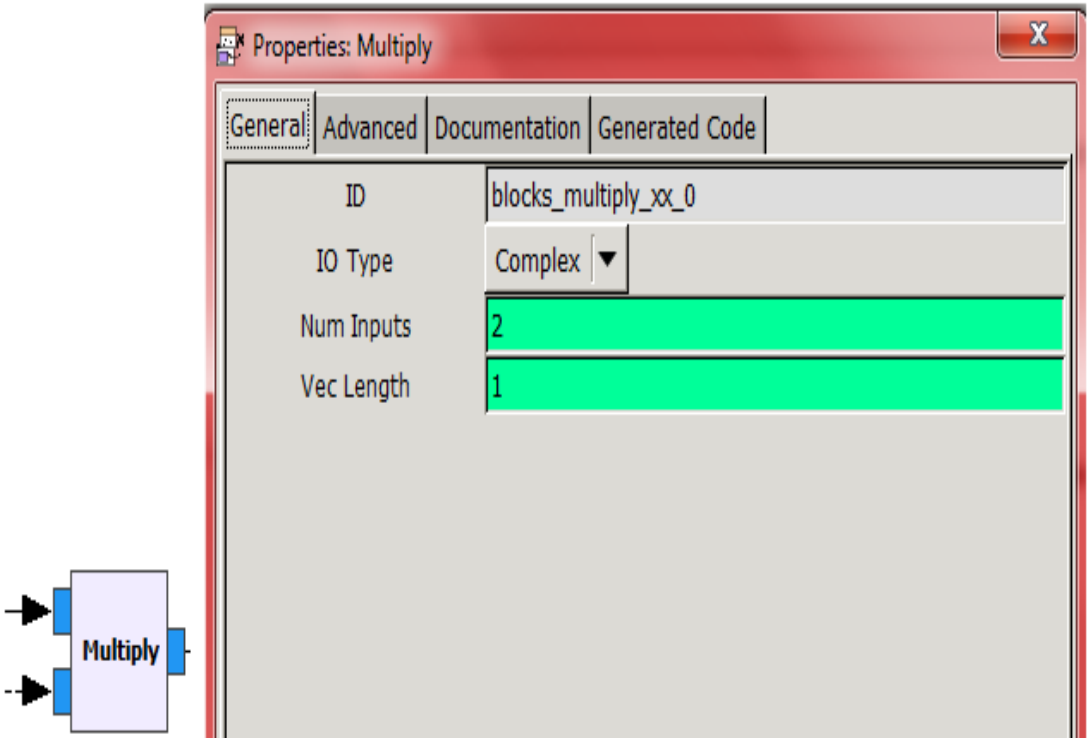
Este bloque multiplica el valor indicado para todas las secuencias de la señal de entrada.

Salida = entrada * valor dado.

Parámetros del bloque Multiply, ver figura 8.

- *Num inputs*: valor dado para multiplicar con la señal de entrada.

Figura 8. **Mutiply**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

2.3.8. Multiply Const

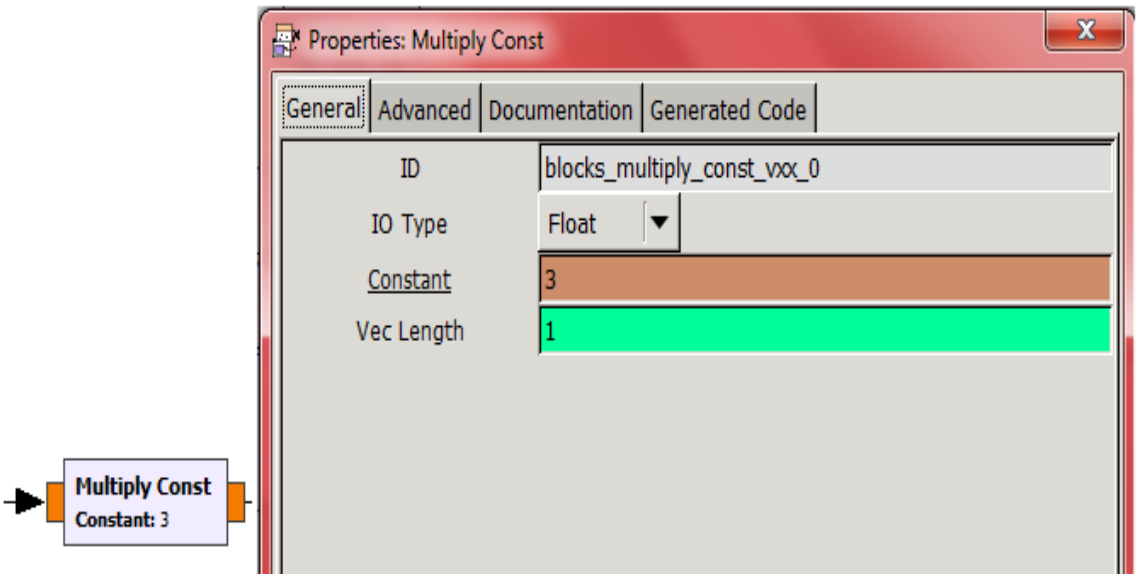
Este bloque multiplica la entrada por una constante, ya sea escalar o vectorial. Si no se está trabajando con un vector, una versión rápida se invoca automáticamente. Con este bloque se puede controlar el volumen de la señal de audio.

Salida = Entrada * Escalar o vectorial

Parámetros del bloque Multiply Const, ver figura 9.

- *Constant*: se da un valor vector, si la entrada es un vector, si no se escribe un escalar.

Figura 9. **Multiply Const**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

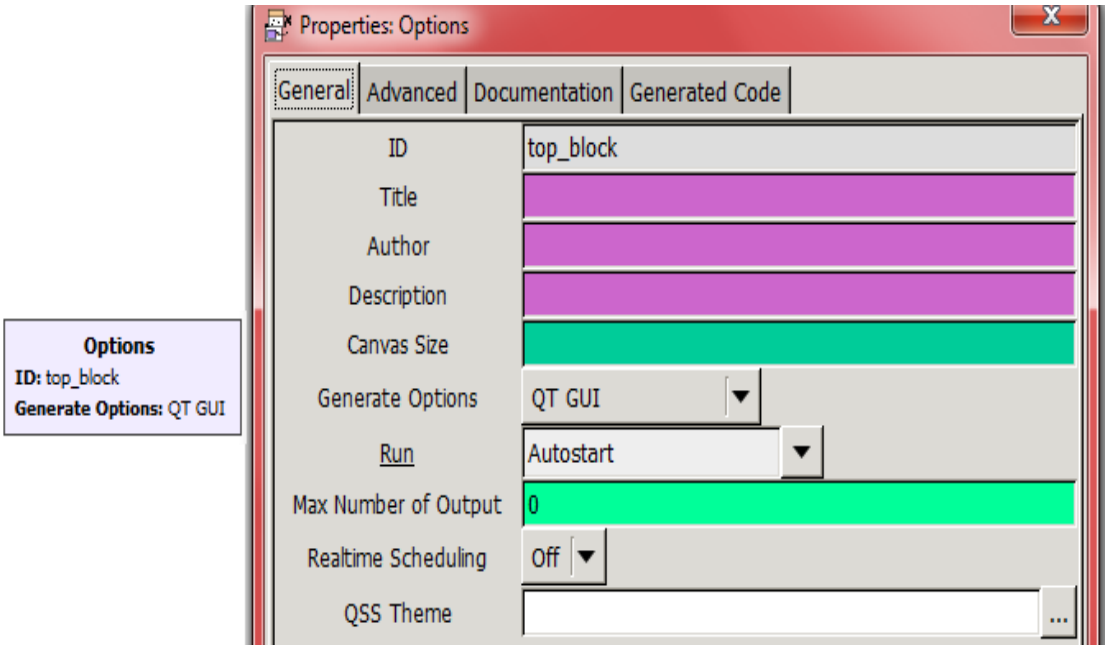
2.3.9. Options

Este bloque por defecto se encuentra en nuestra área de trabajo, solo se permite un bloque Options. En este bloque se describen ciertas características generales de la aplicación a trabajar.

Parámetros del bloque Options, ver figura 10.

- *ID*: determina el nombre del archivo generado y el nombre de la clase.
- *Canvas size*: controla las dimensiones del editor de gráficos de flujo. Las dimensiones deben de estar entre 300 para el ancho y 4096 para el alto.
- *Generate options*: controla el tipo de bloques o código con el que se trabajará, puede ser QT GUI o WX GUI.
- *Run*: este puede controlarse con una variable para dar inicio o detener la aplicación en tiempo de ejecución.
- *Max number of output*: es el número máximo de elementos de salida permitidos para cualquier bloque en nuestra aplicación, se usa para ajustar la latencia máxima en la aplicación. Para deshabilitarlo se establece a un valor 0.

Figura 10. **Options**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

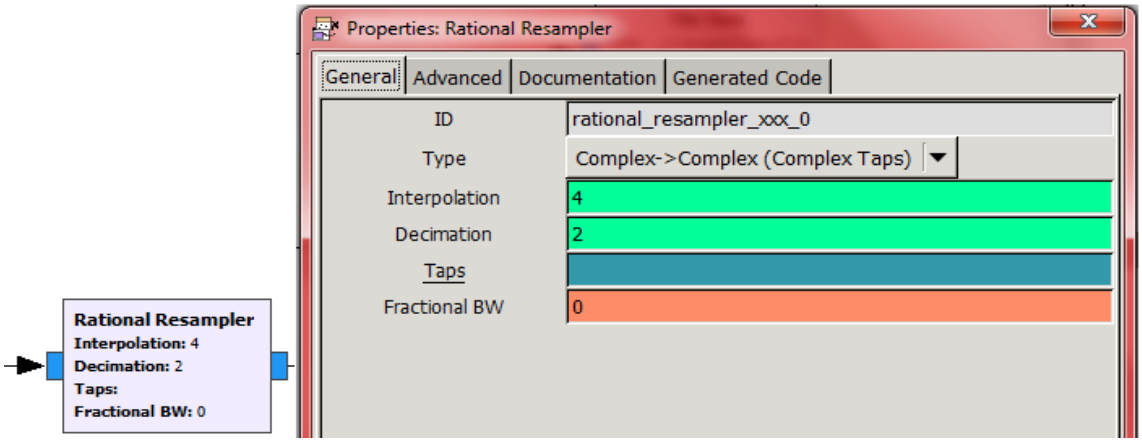
2.3.10. Rational Resampler

Este bloque tiene la característica de ser un Filtro FIR polifásico de remuestreo racional.

Parámetros del bloque Rational Resampler, ver figura 11.

- *Interpolation*: factor de interpolación, entero > 0.
- *Decimation*: factor de diezmado, entero > 0.
- *Taps*: son los coeficientes de filtros opcionales.
- *Fractional BW*: es el ancho de banda fraccional, en 0 - 0.5.

Figura 11. **Rational Resampler**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

2.3.11. RTL-SDR Source

Este bloque es quien recibe nuestra señal de entrada, en este se debe configurar parámetros que dependen de la señal que se quiere recibir.

Parámetros del bloque RTL-SDR Source, ver figura 12.

- *Sample rate*: se indica la frecuencia de muestro para la señal.
- Ch0 *frequency*: es la frecuencia de la señal que se quiere recibir.
- Ch0 *freq*. *Corr*: es la frecuencia de corte con la que trabajará.
- Ch0: *gain mode*: es la ganancia que se le dará a la señal recibida.
- Ch0 *RF gain*: regula la intensidad de la señal de radio frecuencia.
- Ch0 *IF gain*: regula la intensidad de la señal de frecuencia intermedia.
- Ch0 *BB gain*: regula la intensidad de la señal de banda base.
- Ch0 *bandwidth*: es el ancho de banda con el que se estará trabajando.

Figura 12. **RTL-SDR Source**

Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

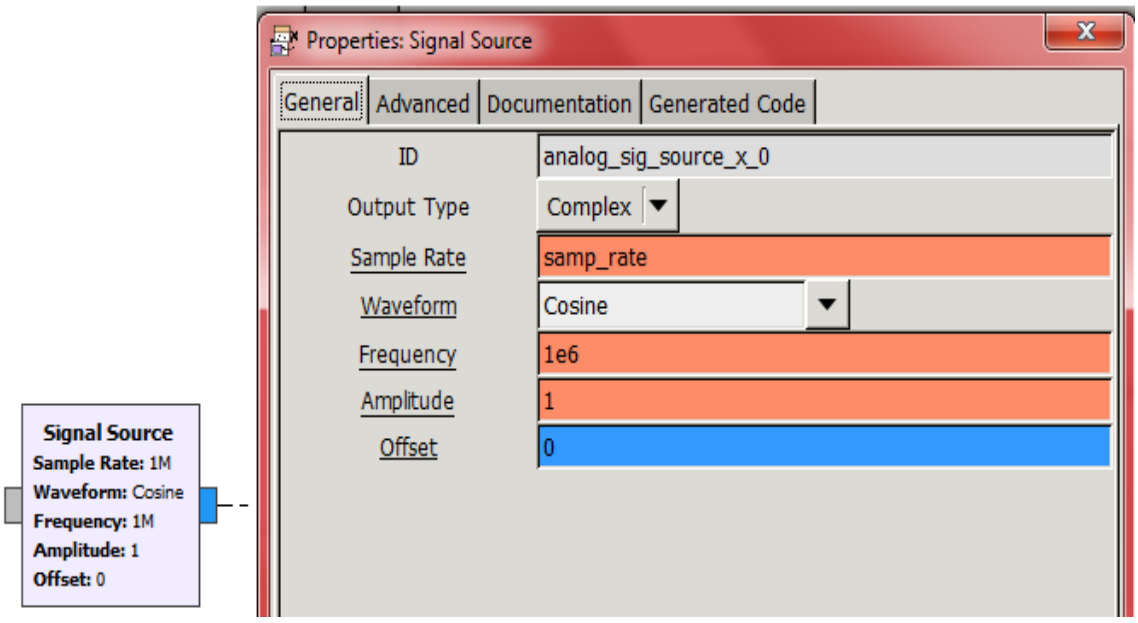
2.3.12. Signal Source

Este bloque es una fuente de señal, el tipo de señal que se genera depende de los parámetros.

Parámetros del bloque Signal Source, ver figura 13.

- *Sample rate*: es la frecuencia de muestreo con la que se trabajará.
- *Waveform*: se indica el tipo de la forma de la señal que se quiere generar.
- *Frequency*: es la frecuencia de nuestra de la forma de onda generada.
- *Amplitude*: será la amplitud que tendrá nuestra señal generada.
- *Offset*: es el desplazamiento que queremos que tenga nuestra señal.

Figura 13. **Signal Source**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

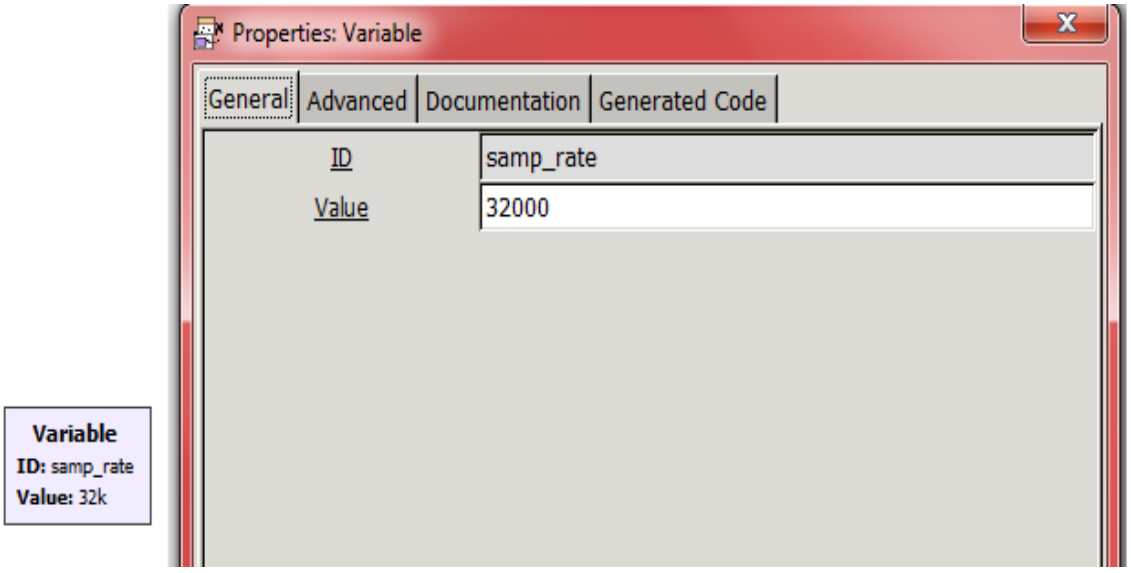
2.3.13. Variable

Este bloque es utilizado como una variable en muchos bloques dentro de nuestra aplicación. Es muy útil ya que, al cambiar el valor de este bloque, ese cambio se verá reflejado en todos los bloques que estén utilizando el nombre de nuestro bloque variable y no se tendrá la necesidad de cambiar el valor manualmente en cada bloque que lo necesite.

Parámetros del bloque Variable, ver figura 14.

- *ID*: es el nombre al que se hace referencia a la variable en los valores de otros bloques.
- *Value*: valor numérico que puede ser cambiado en tiempo real.

Figura 14. **Variable**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

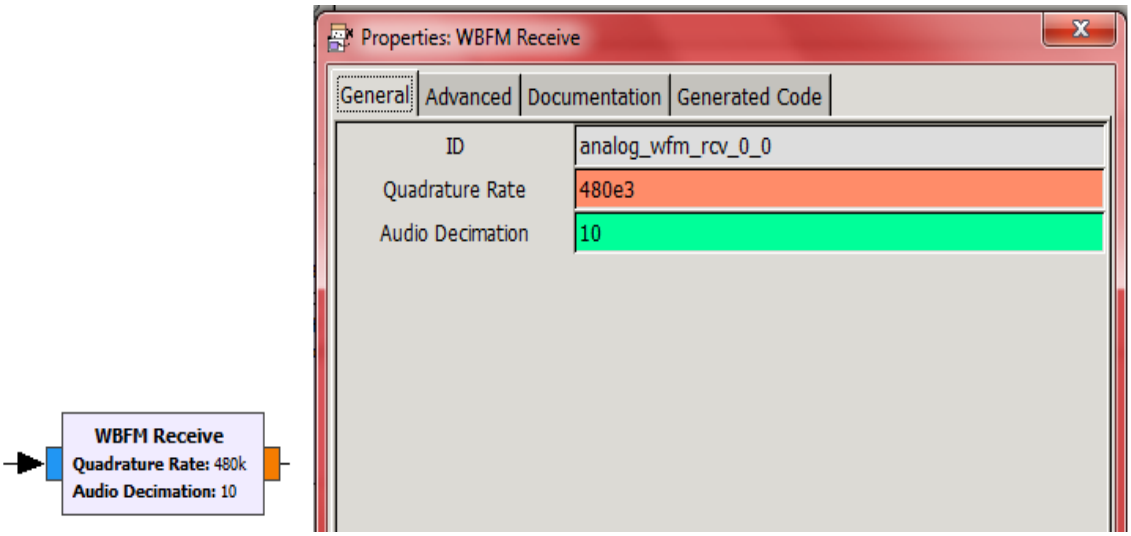
2.3.14. WBFM Receive

Este bloque es un demodulador de una señal de transmisión FM. Entra una señal de banda base compleja convertida hacia abajo y sale un audio demodulador de tipo flotante.

Parámetros del bloque WBFM Receive, ver figura 15.

- *Quadrature rate*: es el valor de ancho de banda recibido.
- *Audio decimation*: cuánto diezmar la velocidad del canal para llegar al audio que se necesita, en el caso de unas tarjetas de audio, se necesita llegar a 48kHz por lo que se da a *audio decimation* el valor de 10.

Figura 15. **WBFM Receive**



Fuente: elaboración propia, realizado con GNU Radio y Macromedia Fireworks.

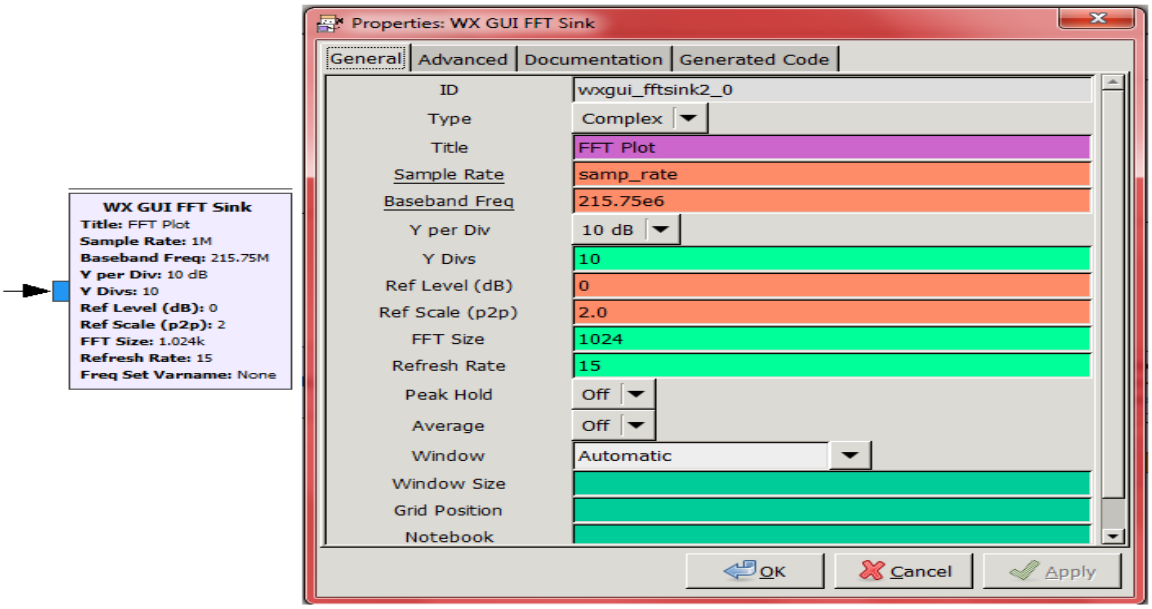
2.3.15. WX GUI FFT Sink

Este bloque se utiliza como un analizador de espectro de la señal de entrada, por lo que se tiene la gráfica de la transformada rápida de Fourier de esta señal.

Parámetros de WX GUI FFT Sink, ver figura 16.

- *Sample rate:* frecuencia de muestreo con la que se estará trabajando.
- *Baseband freq*: frecuencia que se utilizará para la recepción de la señal.
- FFT *size*: es el tamaño que se usará para calcular y mostrar la FFT.
- *Refresh rate*: es la frecuencia con la que se estará actualizando la FFT.

Figura 16. **WX GUI FFT Sink**



Fuente: elaboración propia, empleando GNU Radio y Macromedia Fireworks.

3. CREACIÓN DE MÓDULOS Y BLOQUES PERSONALIZADOS, UTILIZANDO GR-MODTOOL

EL módulo OOT ya está incluido en el programa de GNU Radio, es utilizado para crear los módulos personalizados y dentro de estos módulos personalizados agregar nuevos bloques.

Encontrará que en las imágenes de ejemplo está incluido el nombre Elizabeth, fue utilizado con el fin de demostrar que las características de cada bloque deben ser definidas por el usuario, empezando con el nombre, al igual que las distintas acciones que pueden definirse para los bloques personalizados.

Se pueden crear dos tipos de bloques, estos pueden ser de Python o de C++, dependiendo de la acción que tendrá el bloque. Los bloques de Python pueden ser más utilizados para funciones gráficas y los bloques de C++ cuando se necesite un alto rendimiento.

- Se utilizará gr-modtool para crear un módulo nuevo fuera del árbol.
- Dentro del módulo nuevo con gr-modtool se realizará un bloque nuevo de Python y uno de C++.
- Se modificarán códigos de Python y C++ para darles las características personalizadas del funcionamiento de cada bloque.
- Se modificará el archivo YAML y .CC.
- Se instalará el bloque y se utilizará dentro de un ejemplo para demostrar que el bloque nuevo funciona correctamente.

Se utilizará GNU Radio V3.10.1.1 y el sistema operativo Ubuntu V21.10 o versiones más recientes.

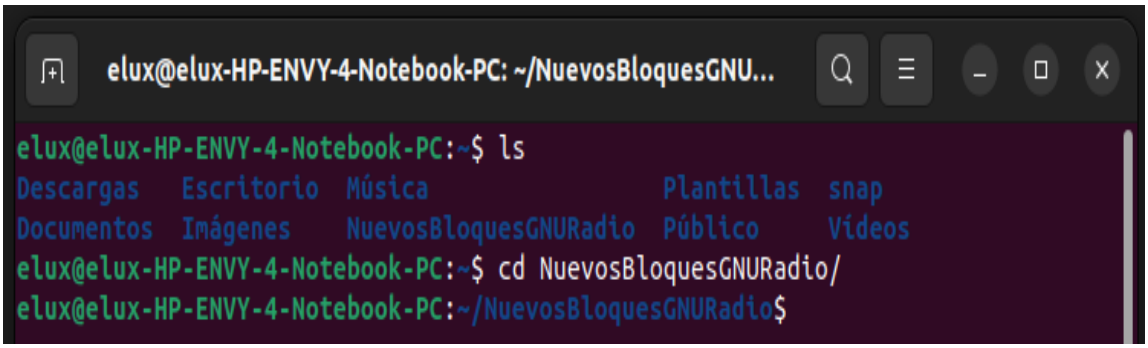
3.1. Creación de nuevo módulo OOT y bloque personalizado de Python

A continuación, se indicarán los pasos para crear el módulo OOT y bloques de Python que van dentro de este módulo OOT y así obtener nuevos bloques en la interfaz de GNU Radio. (wiki.gnuradio, 2023)

3.1.1. Pasos para la realización de módulo OOT en GNU Radio

En la línea de comandos del sistema operativo Ubuntu, ingrese a una carpeta con directorio apropiado para realizar el módulo y bloque.

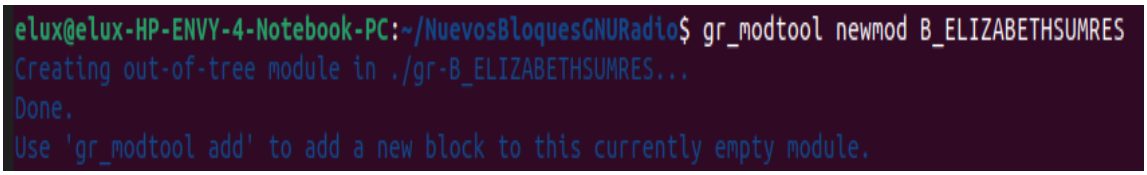
Figura 17. **Ingreso a carpeta para iniciar a crear módulo y bloques personalizados**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Cree un módulo personalizado con el siguiente comando, indicando el nombre que se le quiere dar, en este caso es B_ELIZABETHSUMRES.

Figura 18. **Creación de módulo**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se creó el módulo, el cual contiene el código esqueleto. Ingrese a la carpeta de este módulo creado, encontrará los siguientes archivos y directorios:

Figura 19. **Archivos y directorios del módulo personalizado**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

3.1.2. Pasos para la creación de un bloque personalizado OOT de Python para GNU Radio

Se creará un bloque personalizado que dependiendo de si el usuario seleccione *true* o *false*, el bloque puede sumar o restar las dos entradas y dar el resultado en una sola salida.

Cree un nuevo bloque escribiendo el siguiente comando dentro de la carpeta del módulo creado anteriormente, indicando el nombre que quiere darle al nuevo bloque, en este caso es SRElizabeth.

Figura 20. **Comando para crear un bloque personalizado**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se iniciará un cuestionario sobre cómo se definirán las características del bloque personalizado, esto creará los nuevos archivos con el nombre indicado.

- Seleccione en este caso el bloque de sincronización: *sync.*
- Como lenguaje ingrese: Python.
- Ingrese el nombre de los derechos de autor.
- Ingrese la lista de argumentos. Esto dependerá de la acción que tendrá su bloque. En este caso el argumento inicial será *true*.
- Ingrese n ya que no se quiere el código de garantía de calidad.

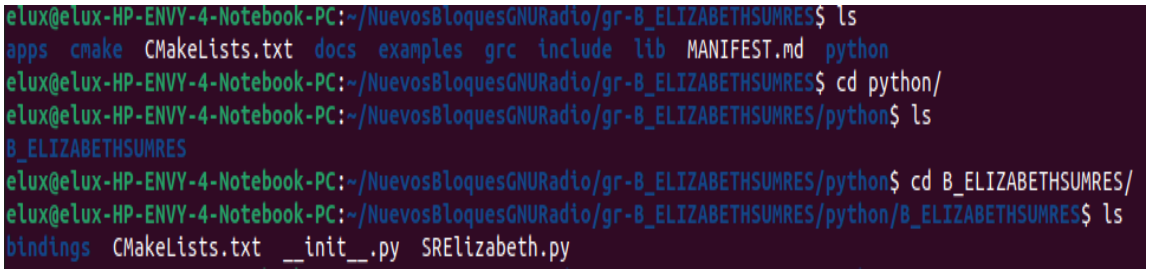
Figura 21. **Características del bloque personalizado**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

3.1.3. Modificación del archivo Python para especificar las funciones que tendrá el bloque personalizado

Ingrese a la carpeta de Python, encontrará otra carpeta con el nombre del nuevo módulo creado, ingrese a esa carpeta, en este encontrará el archivo con el nombre que indico para el bloque personalizado, este debe ser modificado para especificar la función del nuevo bloque.

Figura 22. **Ingreso a la carpeta del bloque personalizado**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Abra el archivo Python del bloque personalizado para editarlo.

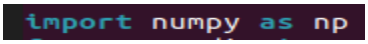
Figura 23. **Archivo Python**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Modifique de acuerdo con sus acciones requeridas, para nuestro caso nuestro programa quedará de la siguiente manera:

Se modifica la instrucción de importación.

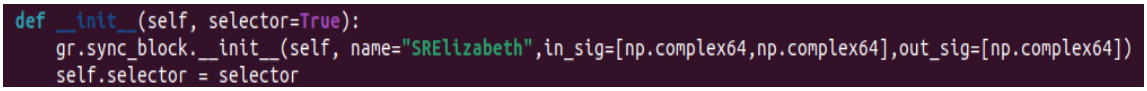
Figura 24. **Importación en Python**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se debe modificar las funciones de *init* y *work*. Nuestro bloque aceptará dos entradas complejas y tendrá una salida compleja, por ellos se cambian en la función *init* los parámetros *in_sig* y *out_sig*. Se agrega el parámetro *selector* con su argumento inicial *true*, este será una variable miembro.

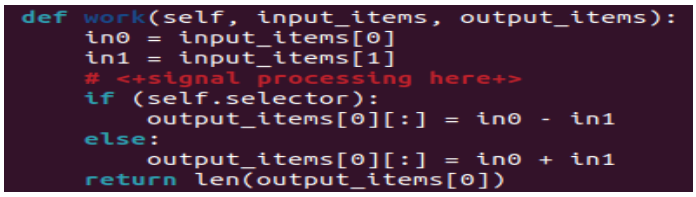
Figura 25. **Función *init***



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Nuestro bloque sumará y restará dependiendo el valor de nuestro parámetro, por ello se modifica la función *work* para tener el resultado esperado. En nuestro programa especificamos dos entradas, las cuales se sumarán o restarán dando como resultado una salida.

Figura 26. **Función *work***



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

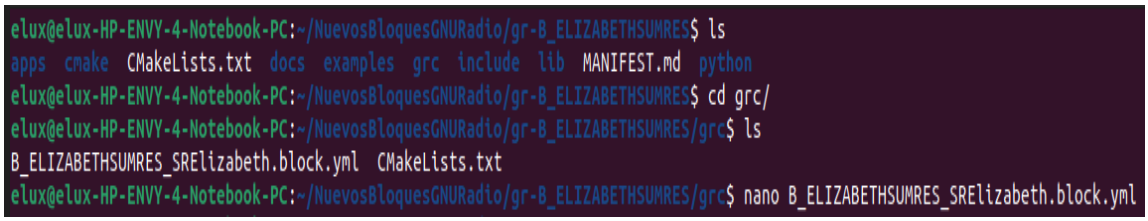
Guarde los cambios realizados en el archivo con CTRL + S.

3.1.4. Modificación del archivo YAML para especificar la interfaz del nuevo bloque

Este archivo YAML debe coincidir con el archivo de Python, es decir, debe tener especificado los parámetros, las entradas y salidas que se utilizaron en Python.

Abra el archivo .yml.

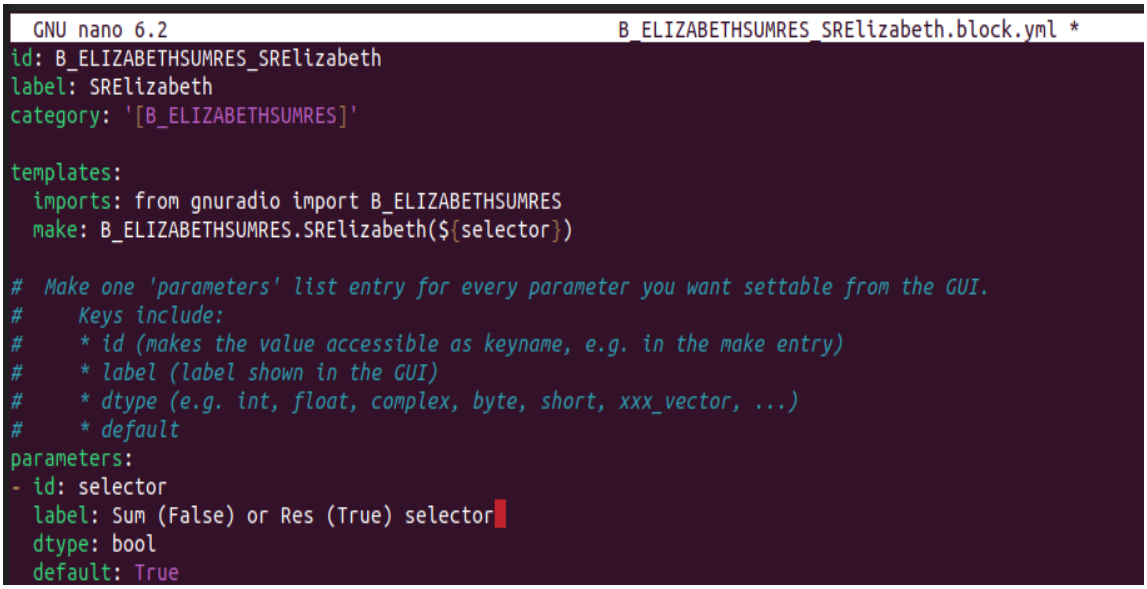
Figura 27. **Archivo .yml**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se define el parámetro utilizado en la parte de *parameters*:

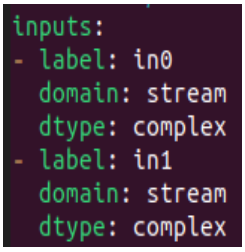
Figura 28. **Parámetros en archivo .yml**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se definen las dos entradas.

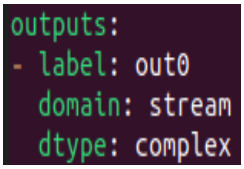
Figura 29. **Definición de entradas en archivo .yml**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se define la salida.

Figura 30. **Definición de salida en archivo .yml**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Guarde los cambios realizados en el archivo con CTRL + S

3.1.5. Compilación e instalación del bloque

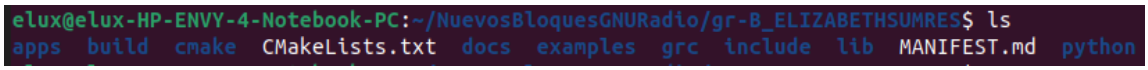
En el directorio de gr_(Nombre dado a su bloque), se debe realizar un directorio de compilación con el nombre *build* utilizando el comando *mkdir*.

Figura 31. **Creación de carpeta *build***

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Al crear el directorio *build*, el directorio gr debe tener estos archivos y directorios.

Figura 32. **Contenido del directorio gr del nuevo bloque**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Ingrese al directorio de compilación *build*

Figura 33. **Ingreso a directorio *build***

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

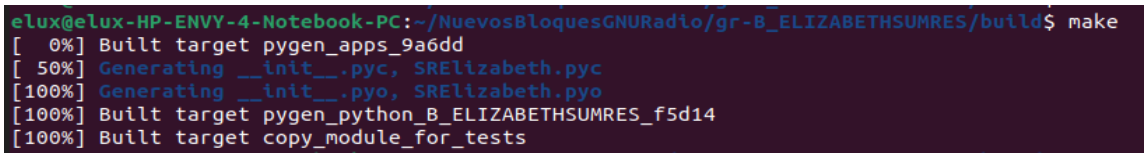
Ejecute *cmake* .. para preparar los archivos *make*.

Figura 34. **Ejecución de *cmake* ..**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Compile el módulo.

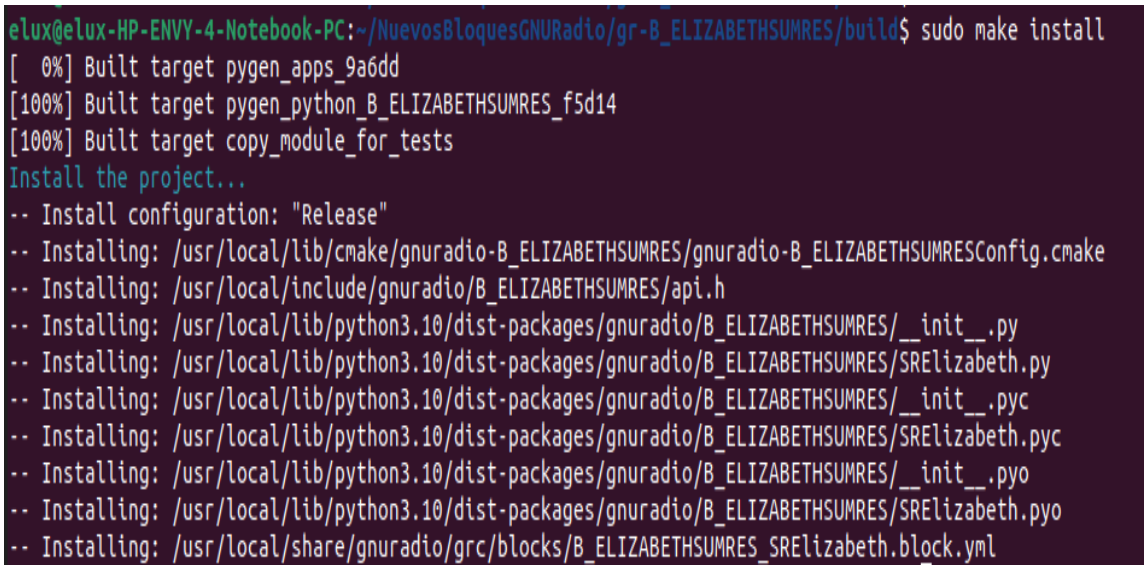
Figura 35. **Compilación de módulo personalizado**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Instale el módulo, con ellos se instalarán varios archivos.

Figura 36. **Instalación de módulo personalizado**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Actualice la vinculación de la biblioteca del nuevo módulo con:

Figura 37. **Vinculación de biblioteca**

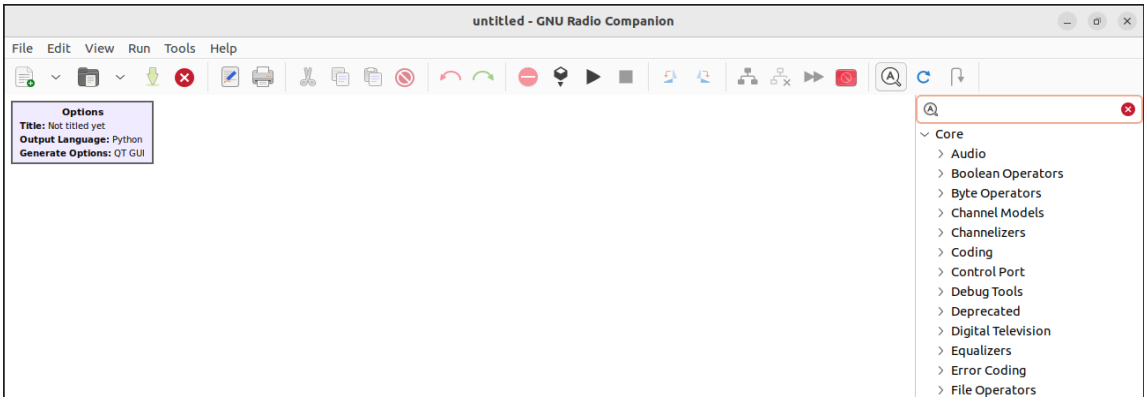
Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Con todos los pasos realizados anteriormente el nuevo bloque ya puede ser utilizado en GNU Radio.

3.1.6. Uso del nuevo bloque de Python en GNU Radio

Abra el programa de GNU Radio, en el lado derecho haga clic en la imagen de lupa, esto nos permitirá buscar nuestro modulo y bloque personalizado.

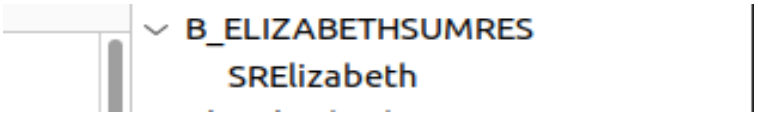
Figura 38. **GNU Radio – búsqueda**



Fuente: elaboración propia, realizado con GNU Radio.

Escriba en el espacio el nombre del módulo creado.

Figura 39. **Búsqueda de modulo personalizado**

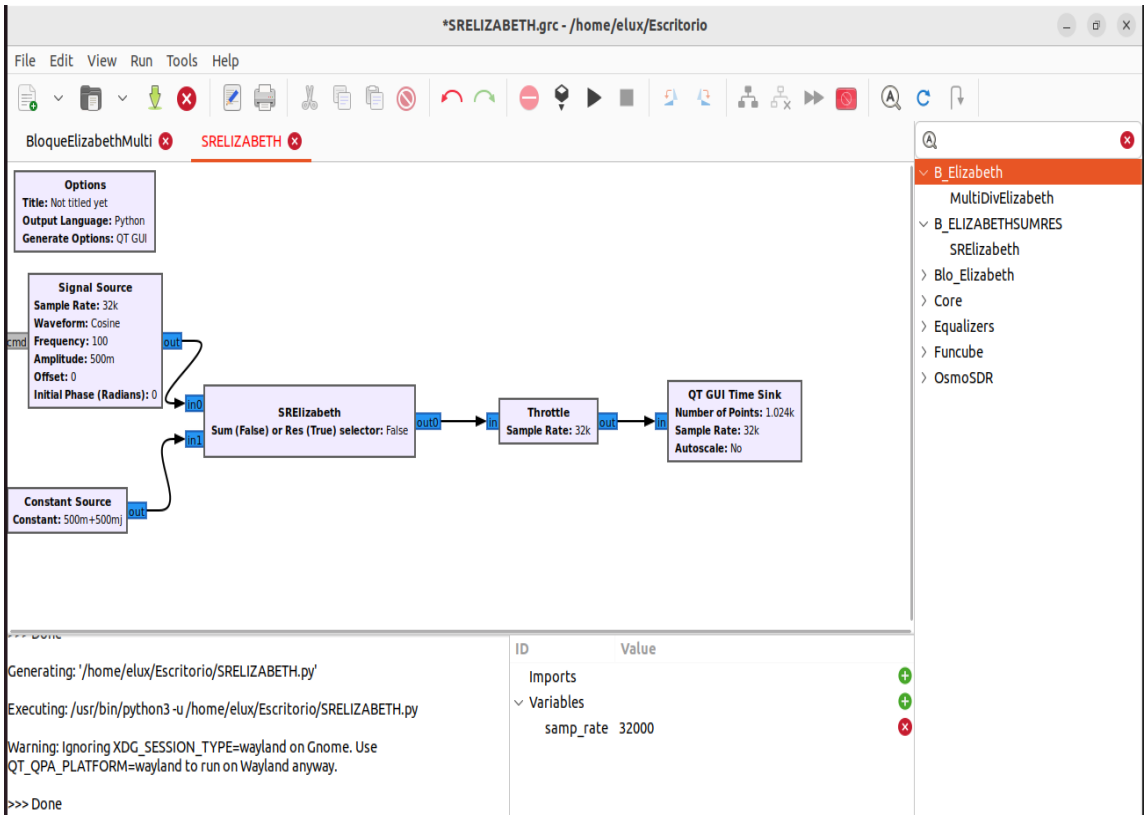


Fuente: elaboración propia, realizado con GNU Radio.

Seleccione el nuevo bloque y agréguelo al espacio de trabajo de GNU Radio. Se puede observar las dos entradas y la única salida que se definieron en los archivos.

Agregue los demás bloques de fuentes de señal y constantes que serán las entradas del nuevo bloque, especifique el parámetro a utilizar en el bloque personalizado, ya sea *true* o *false*, agregue los bloques de Throttle y QT GUI Time Sink a la salida. Puede configurar los bloques de acuerdo con las características de la siguiente imagen.

Figura 40. **Uso de bloque personalizado de Python en GNU Radio**

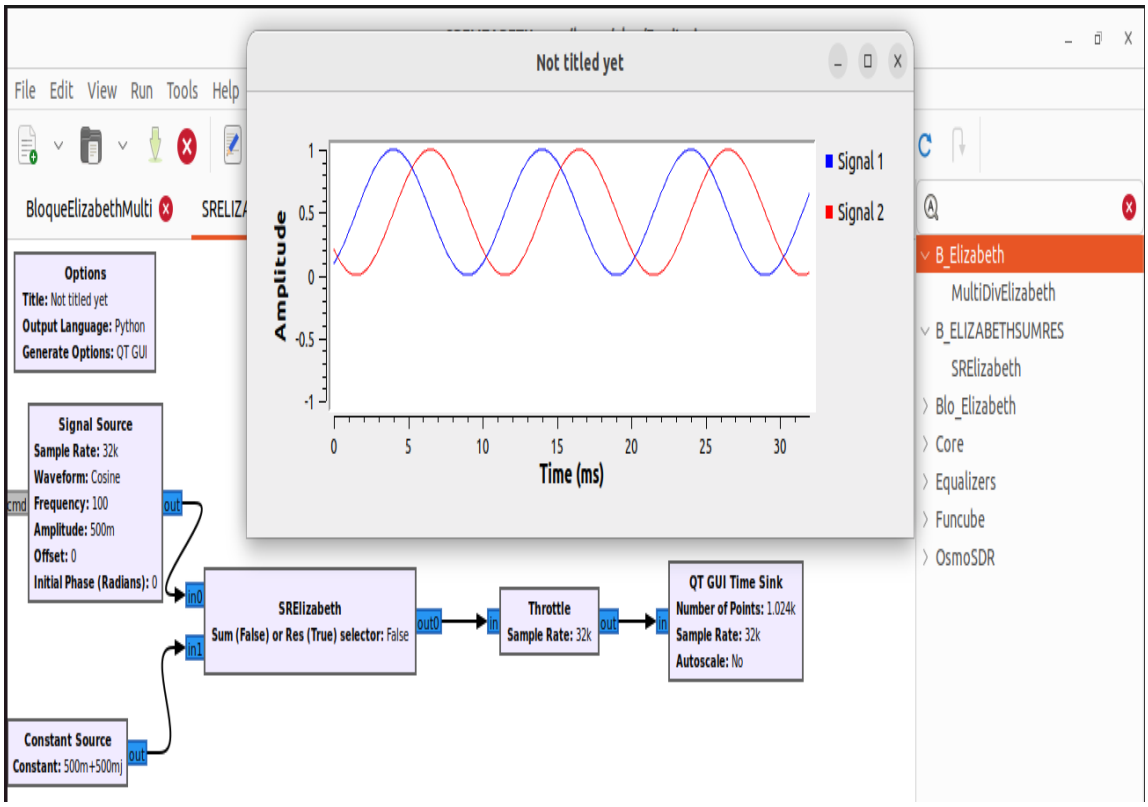


Fuente: elaboración propia, realizado con GNU Radio.

Puede ir variando las características de los bloques de fuente de señal a la entrada para ver los cambios a la salida del bloque personalizado. También puede modificar el parámetro de *true* o *false* en el bloque personalizado.

Ejemplo con parámetro *false.*

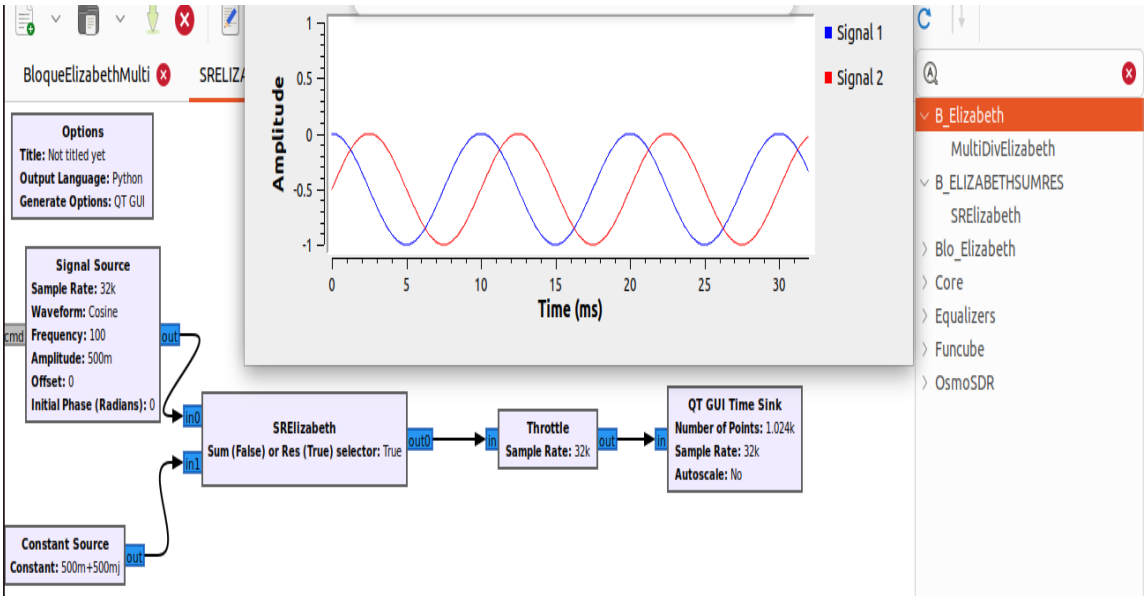
Figura 41. **Ejemplo con bloque personalizado utilizando el parametro *false***



Fuente: elaboración propia, realizado con GNU Radio.

Ejemplo con parámetro *true.*

Figura 42. **Ejemplo con bloque personalizado utilizando el parámetro *true***



Fuente: elaboración propia, realizado con GNU Radio.

Si es necesario realizar cambios en los archivos que se usaron para la configuración del nuevo bloque, debe eliminar la carpeta *build* y realizar de nuevo los pasos para crear e instalar el bloque personalizado.

Utilice el siguiente comando si es necesario eliminar la carpeta *build*:

Figura 43. **Comando para eliminar directorio *build***

```
elux@elux-HP-ENVY-4-Notebook-PC:~/NuevosBloquesGNURadio/gr-Blo_Elizabeth$ sudo rm -rf build/
```

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

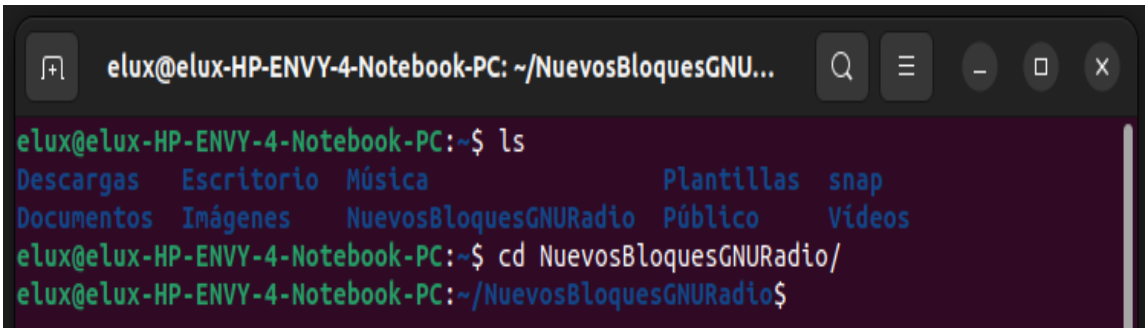
3.2. Creación de bloque personalizado de C++ dentro del módulo OOT

A continuación, se indicarán los pasos para crear el módulo OOT y bloques de C++ que van dentro de este módulo OOT y así obtener nuevos bloques en la interfaz de GNU Radio. (wiki.gnuradio, 2023)

3.2.1. Pasos para la realización de módulo OOT en GNU Radio

En una línea de comandos del sistema operativo Ubuntu, ingrese a una carpeta con directorio apropiado para realizar el módulo y bloque.

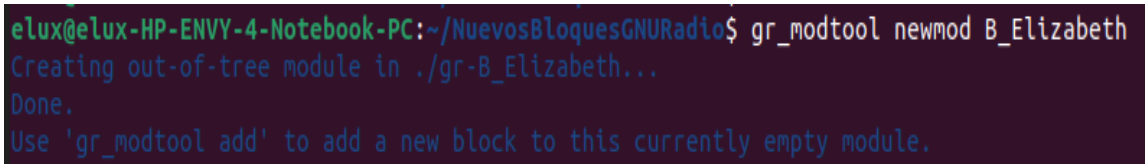
Figura 44. **Ingreso a carpeta para iniciar a crear módulo y bloques personalizados para C++**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Cree un módulo personalizado con el siguiente comando, indicando el nombre que se le quiere dar, en este caso es B_Elizabeth.

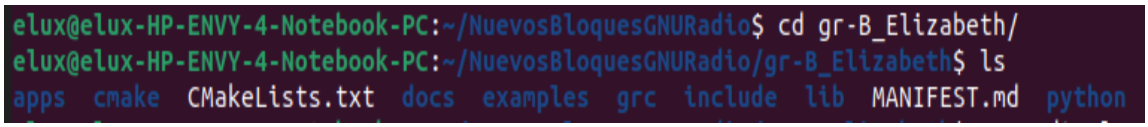
Figura 45. **Creación de módulo B_Elizabeth**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se creó el módulo, el cual contiene el código esqueleto. Ingrese a la carpeta de este módulo creado, encontrará los siguientes archivos y directorios:

Figura 46. **Archivos y directorios del módulo personalizado B_Elizabeth**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

3.2.2. Pasos para la creación de un bloque personalizado OOT de C++ para GNU Radio

Se creará un bloque personalizado que dependiendo de si el usuario seleccione *true* o *false*, el bloque pueda multiplicar tres entradas o multiplicar dos entradas y dividirlas entre la tercera entrada y así dar el resultado en una sola salida.

Cree un nuevo bloque escribiendo el siguiente comando dentro de la carpeta del módulo creado anteriormente, indicando el nombre que quiere darle al nuevo bloque, en este caso es MultiDivElizabeth.

Figura 47. **Comando para crear un bloque personalizado de C++**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se iniciará un cuestionario sobre cómo se definirá las características del bloque personalizado, esto creará los nuevos archivos con el nombre indicado.

- Seleccione en este caso el bloque de sincronización: *sync.*
- Como lenguaje ingrese: *cpp*.
- Ingrese el nombre de los derechos de autor.
- Ingrese la lista de argumentos. Esto dependerá de la acción que tendrá su bloque. En este caso el argumento inicial será *true*.
- Ingrese n ya que no se quiere el código de garantía de calidad.

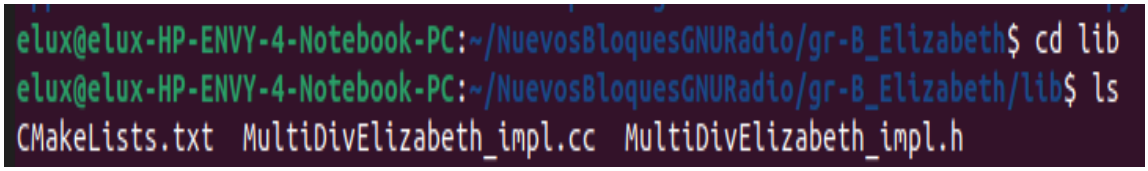
Figura 48. **Características del bloque personalizado**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

3.2.3. Modificación del archivo .h

Ingrese a la carpeta lib, en esta carpeta encontrará el archivo con el nombre que indico para el bloque personalizado MultiDivElizabeth_impl.h, este debe ser modificado para definir el funcionamiento del bloque.

Figura 49. **Ingreso a la carpeta lib**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Abra el archivo .h del bloque personalizado para editarlo.

Figura 50. **Archivo .h**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Modifique de acuerdo con sus acciones requeridas, para nuestro caso nuestro programa quedará de la siguiente manera:

En *private* se especifica que el *selector* será booleano.

Figura 51. **Especificación del *selector* en archivo .h**

```
#ifndef INCLUDED_B_ELIZABETH_MULTIDIVELIZABETH_IMPL_H
#define INCLUDED_B_ELIZABETH_MULTIDIVELIZABETH_IMPL_H

#include <gnuradio/B_Elizabeth/MultiDivElizabeth.h>

namespace gr {
namespace B_Elizabeth {

class MultiDivElizabeth_impl : public MultiDivElizabeth
{
private:
    bool _selector;

public:
    MultiDivElizabeth_impl(bool selector);
    ~MultiDivElizabeth_impl();

    // Where all the action really happens
    int work(int noutput_items,
             gr_vector_const_void_star& input_items,
             gr_vector_void_star& output_items);
};

} // namespace B_Elizabeth
} // namespace gr

#endif /* INCLUDED_B_ELIZABETH_MULTIDIVELIZABETH_IMPL_H */
```

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Guarde los cambios realizados en el archivo con CTRL + S.

3.2.4. Modificación del archivo .cc

Ingrese a la carpeta lib, en esta carpeta encontrará el archivo con el nombre que indico para el bloque personalizado MultiDivElizabeth_impl.cc, este debe ser modificado para definir el funcionamiento del bloque.

Abra el archivo .cc del bloque personalizado para editarlo.

Figura 52. **Archivo .cc**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Modifique de acuerdo con sus acciones requeridas, para nuestro caso nuestro programa quedará de la siguiente manera:

Nuestro bloque aceptará entradas complejas y tendrá una salida compleja, debe configurarse de esta manera.

Figura 53. **Entradas y salidas complejas**

```
#include "MultiDivElizabeth_impl.h"
#include <gnuradio/io_signature.h>

namespace gr {
namespace B_Elizabeth {

#pragma message("set the following appropriately and remove this warning")
using input_type = gr_complex;
#pragma message("set the following appropriately and remove this warning")
using output_type = gr_complex;
```

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Defina tres entradas y una sola salida. Debe almacenar el valor del parámetro *selector*, se debe utilizar *_selector* ya que así es como quedó definido en el documento .h

Figura 54. **Configuración de entradas, salida y parámetro *selector***

```
MultiDivElizabeth_impl::MultiDivElizabeth_impl(bool selector)
    : gr::sync_block("MultiDivElizabeth",
                     gr::io_signature::make(
                         3 /* min inputs */, 3 /* max inputs */, sizeof(input_type)),
                     gr::io_signature::make(
                         1 /* min outputs */, 1 /*max outputs */, sizeof(output_type)))
{

        _selector=selector;
```

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Debe modificar la función *work* para definir las variables in0, in1, in2 correspondiente a las tres entradas y *out* que corresponde a nuestra salida. Nuestro bloque multiplicará las tres entradas si *_selector* es *true*, si *_selector* es *false* multiplicará las primeras dos entradas y dividirá en la tercera entrada.

Figura 55. **Función *work* ()**

```
MultiDivElizabeth_impl::~MultiDivElizabeth_impl() {}

int MultiDivElizabeth_impl::work(int noutput_items,
                                 gr_vector_const_void_star& input_items,
                                 gr_vector_void_star& output_items)
{
    auto in0 = static_cast<const input_type*>(input_items[0]);
    auto in1 = static_cast<const input_type*>(input_items[1]);
    auto in2 = static_cast<const input_type*>(input_items[2]);
    auto out = static_cast<output_type*>(output_items[0]);

    for (int index=0; index<noutput_items; index++)
        {
                if (_selector)
                        {
                                out[index] = in0[index]*in1[index]*in2[index];
                        }
                else
                        {
                                out[index] = (in0[index]*in1[index])/in2[index];
                        }
        }
```

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Guarde los cambios realizados en el archivo con CTRL + S.

3.2.5. Modificación del archivo YAML para especificar la interfaz del nuevo bloque

Este archivo YAML debe tener especificado los parámetros, las entradas y salidas que se definieron en los documentos .h y .cc.

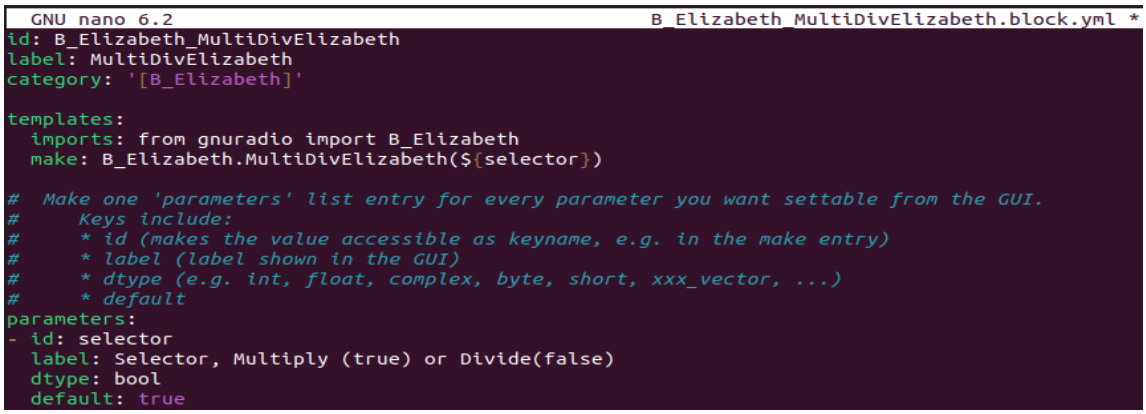
Abra el archivo .yml

Figura 56. **Archivo .yml**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se define el parámetro utilizado en la parte de *parameters*:

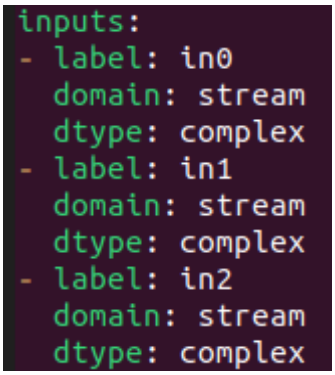
Figura 57. **Parámetros en archivo .yml**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se definen las tres entradas.

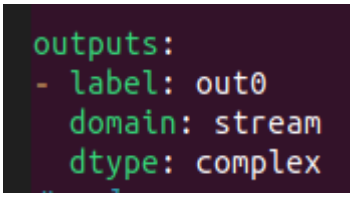
Figura 58. **Definición de entradas en archivo .yml**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Se define la salida.

Figura 59. **Definición de salida en archivo .yml**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Guarde los cambios realizados en el archivo con CTRL + S.

3.2.6. Compilación e instalación del bloque

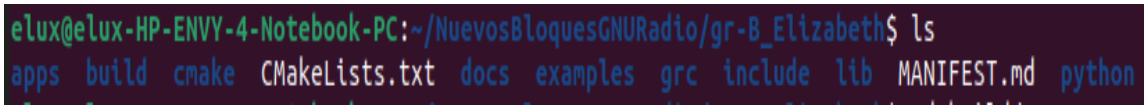
En el directorio de gr_(nombre dado a su bloque), se debe realizar un directorio de compilación con el nombre *build* utilizando el comando *mkdir*.

Figura 60. **Creación de carpeta *build***

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Al crear el directorio *build*, el directorio gr debe tener estos archivos y directorios.

Figura 61. **Contenido del directorio gr del nuevo bloque**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Ingrese al directorio de compilación *build*.

Figura 62. **Ingreso a directorio *build***

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Ejecute *cmake* .. para preparar los archivos *make*.

Figura 63. **Ejecución de *cmake* ..**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Compile el módulo

Figura 64. **Compilación de módulo personalizado**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Instale el módulo, con ellos se instalarán varios archivos.

Figura 65. **Instalación de módulo personalizado**

Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Actualice la vinculación de la biblioteca del nuevo módulo con el comando:

Figura 66. **Vinculación de biblioteca**

```
elux@elux-HP-ENVY-4-Notebook-PC:~/NuevosBloquesGNURadio/gr-B_Elizabeth/build$ sudo ldconfig
```

Fuente: elaboración propia, r.ealizado con línea de comandos del sistema operativo Ubuntu.

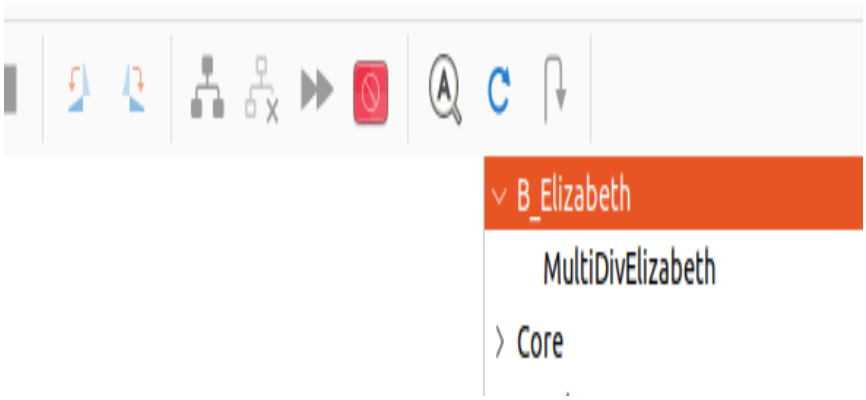
Con todos los pasos realizados anteriormente el nuevo bloque ya puede ser utilizado en GNU Radio.

3.2.7. Uso del nuevo bloque de C++ en GNU Radio

Abra el programa de GNU Radio, en el lado derecho haga clic en la imagen de lupa, esto nos permitirá buscar nuestro módulo y bloque personalizado.

Escriba en el espacio el nombre del módulo creado.

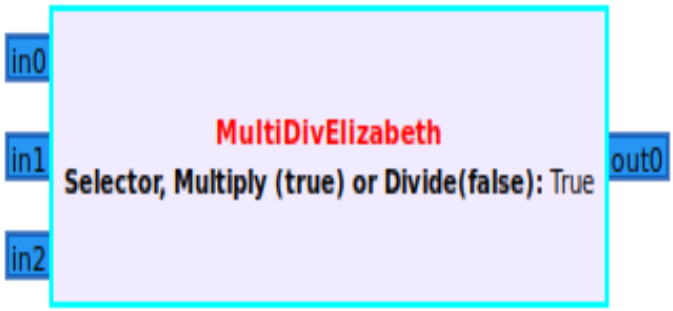
Figura 67. **Búsqueda de módulo y bloque personalizado**



Fuente: elaboración propia, realizado con GNU Radio.

Seleccione el nuevo bloque y agréguelo al espacio de trabajo de GNU Radio. Se puede observar las tres entradas y la única salida que se definieron en los archivos.

Figura 68. **Bloque MultiDivElizabeth en GNU Radio**

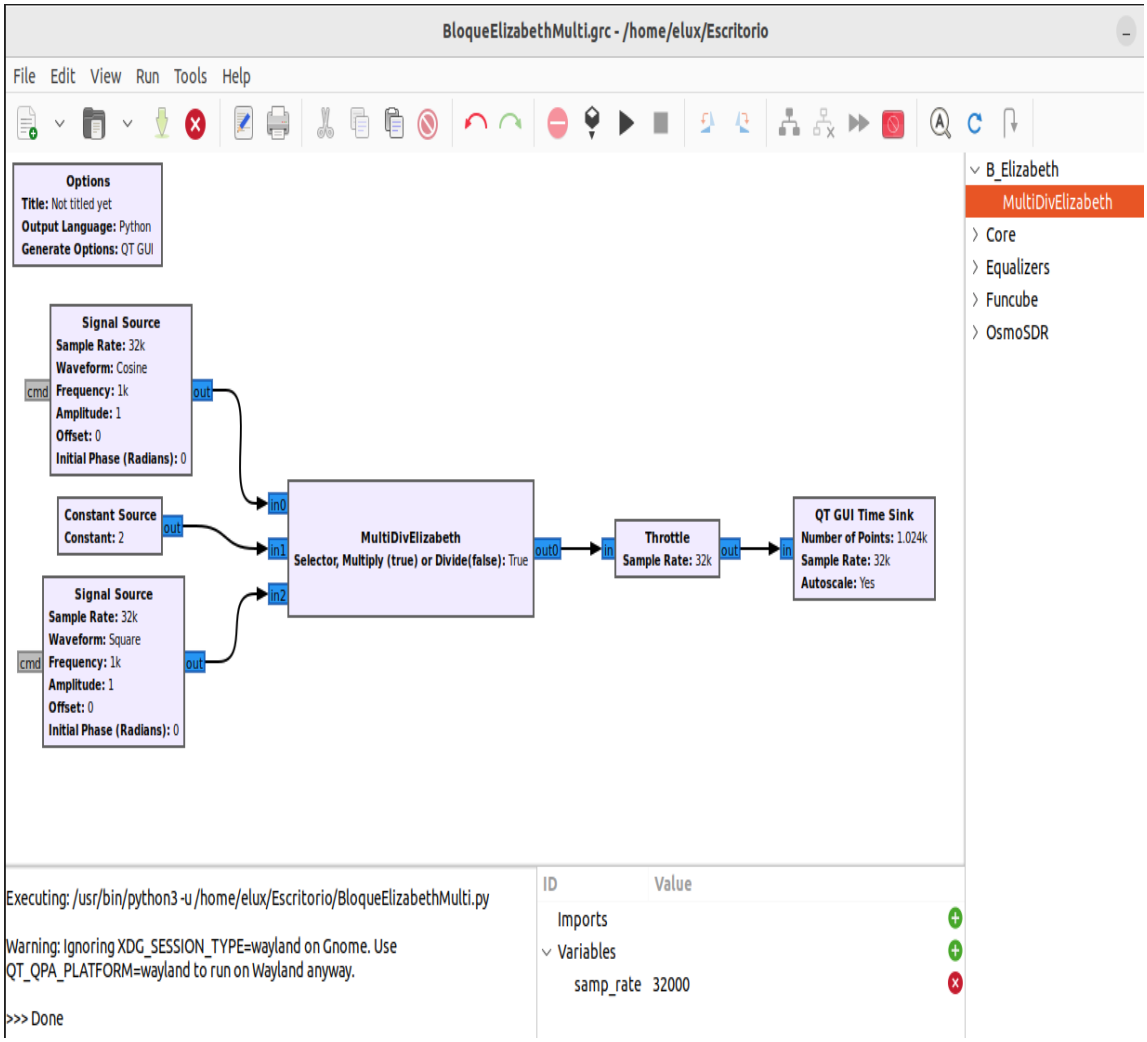


Fuente: elaboración propia, realizado con GNU Radio.

Agregue los demás bloques de fuentes de señal y constantes que serán las entradas del nuevo bloque, especifique el parámetro a utilizar en el bloque personalizado, ya sea *true* o *false*, agregue los bloques de Throttle y QT GUI Time Sing a la salida.

Puede configurar los bloques de acuerdo con las características de la siguiente imagen.

Figura 69. **Uso de bloque personalizado de c++ en GNU Radio**

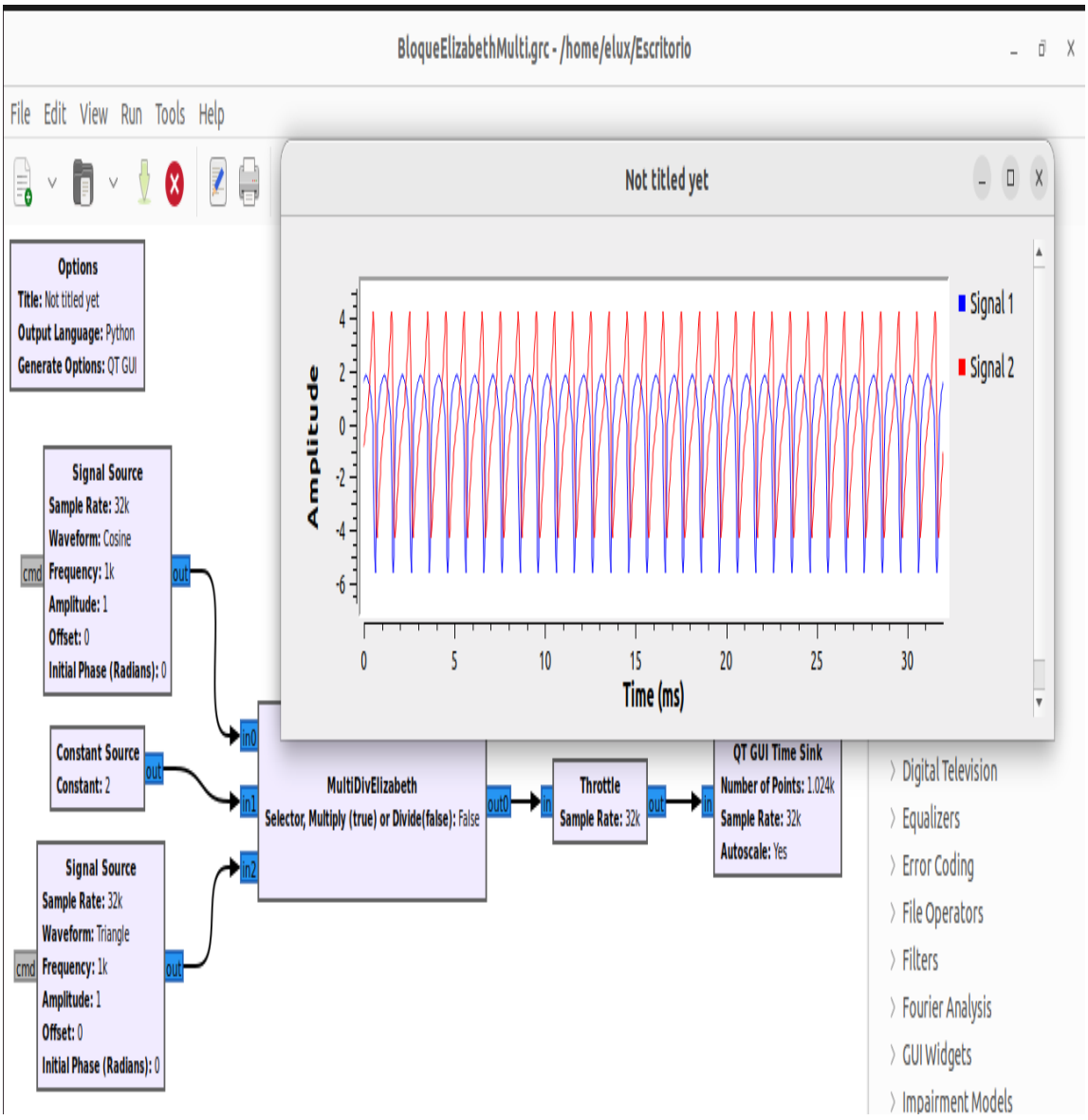


Fuente: elaboración propia, realizado con GNU Radio.

Puede ir variando las características de los bloques de fuentes de señal a la entrada para ver los cambios a la salida del bloque personalizado. También puede modificar el parámetro de *true* o *false* en el bloque personalizado.

Ejemplo con parámetro *false.*

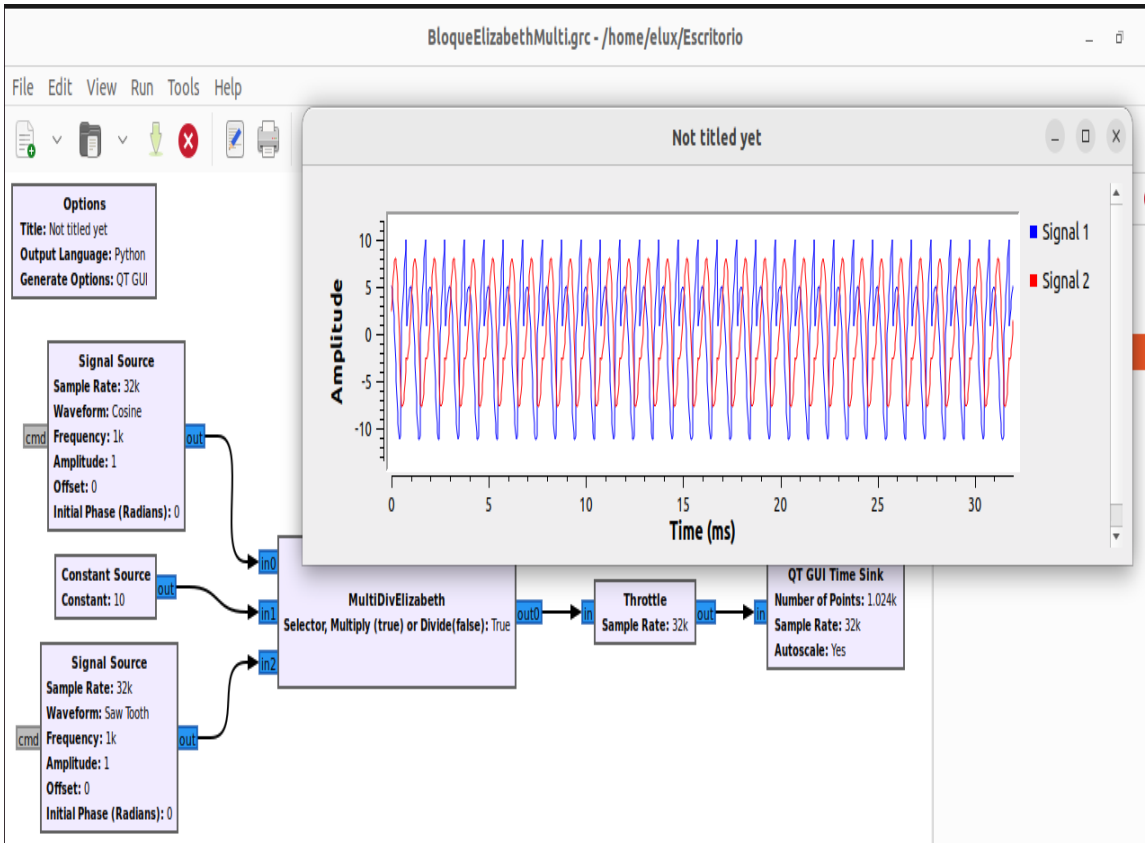
Figura 70. **Ejemplo con bloque personalizado utilizando el parametro *false***



Fuente: elaboración propia, realizado con GNU Radio.

Ejemplo con parámetro *true.*

Figura 71. **Ejemplo con bloque personalizado utilizando el parámetro *true***



Fuente: elaboración propia, realizado con GNU Radio.

Si es necesario realizar cambios en los archivos que se usaron para la configuración del nuevo bloque, debe eliminar la carpeta *build* y realizar de nuevo los pasos para crear e instalar el bloque personalizado.

Utilice el siguiente comando si fuera necesario eliminar la carpeta *build*:

Sudo rm -rf *build*

4. FUNCIONAMIENTO GENERAL DE LA RADIO FM Y LA TELEVISIÓN

Se presentará el funcionamiento general de la radio FM y la televisión, así como información de la norma NTSC y frecuencia portadora de video y audio de algunos canales de televisión.

4.1. Radio FM

La radio modulada en frecuencia es capaz de recibir ondas electromagnéticas de radio para convertirlas en corriente eléctrica y así separar la frecuencia portadora y el ruido de la frecuencia con la información útil que será reproducida para que el oído humano lo capte. El proceso de recepción y reproducción de la información se ve a continuación. (ECURED, s.f.)

4.1.1. Antena

Las señales electromagnéticas son captadas por la antena, la antena capta todas las señales que estén a su alcance.

4.1.2. Circuito de Sintonía

Es el circuito resonante por donde entra la señal deseada, esto se logra haciendo coincidir la frecuencia de resonancia del circuito con la frecuencia de la señal que se recibe, lo que permite impedir el paso de muchas señales no deseadas, en este caso estaciones de radio no deseadas.

4.1.3. Mezclador

En nuestro proceso tenemos un oscilador local, al unir la señal de nuestro oscilador con la señal entrante, se crea un batimiento entre ambas señales que producen la frecuencia intermedia que es igual a 10.7 MHz. A la salida de nuestro mesclador se tiene 4 frecuencias las cuales son: la suma de ambas señales, la resta de ambas señales, la señal de entrada y la señal del oscilador.

4.1.4. Filtro

Este filtro provee al proceso la selectividad y estabilidad ya que está configurado para sintonizar una frecuencia fija.

4.1.5. Demodulador

Este demodulador separa la frecuencia de la señal modulada de la frecuencia de la señal portadora.

4.1.6. Amplificador

La señal al ser demodulada pasa al amplificador de baja frecuencia para que este aumente la amplitud y el altavoz pueda ser estimulado.

4.1.7. Altavoz

Cuando el altavoz es estimulado por la señal amplificada la señal se convierte de eléctrica a acústica. Para que el altavoz pueda reproducir la señal, generalmente esta señal debe tener una frecuencia de 48KHz.

4.2. Televisión

Un sistema de televisión consta de varios equipos que capturan imagen y sonido, mediante ciertos procesos convierten esa información en señales eléctricas análogas o digitales.

Partes que conforman este proceso de TV, ver figura 17. (Monografias.com S.A., s.f)

4.2.1. Sintonizador

Es el bloque receptor de una TV para distintas señales emitidas, estas señales posteriormente serán separadas de señales que no nos interesan.

4.2.2. AGC

El Automatic Gain Control (Control automático de ganancia), es el bloque que equilibra las amplitudes de la salida del amplificador de video en la frecuencia intermedia del canal que se está sintonizando.

4.2.3. Canal FI

La función de este bloque es seleccionar la frecuencia intermedia del video del canal que se quiere sintonizar.

4.2.4. FI de sonido

La función de este boque es separar la imagen del sonido, es decir la frecuencia subportadora del audio.

4.2.5. Salida de audio

Ya que se tiene la señal del audio separada de la señal de video, en este bloque se amplifica la señal del audio y se presenta como pulsos en un dispositivo de salida.

4.2.6. Amplificador de video

Este bloque deberá amplificar desde continua hasta una frecuencia de 4.3MHz, así el video transmitido pueda ser reproducido como modulación de una portadora de RF.

4.2.7. Separador de sincronismo

La función de este bloque es extraer en la señal compuesta de video, la cantidad de impulsos exactos para enclavar la imagen en la pantalla. Este sincronismo se obtiene de unos impulsos de tensión negativa que pueden ser tanto horizontal como vertical y operan a una frecuencia cercana a la del transmisor, este transmisor envía información de cuando inicia una línea y que número de línea es al separador de sincronismos, lo cual evita que flote la imagen de un lado a otro en la pantalla.

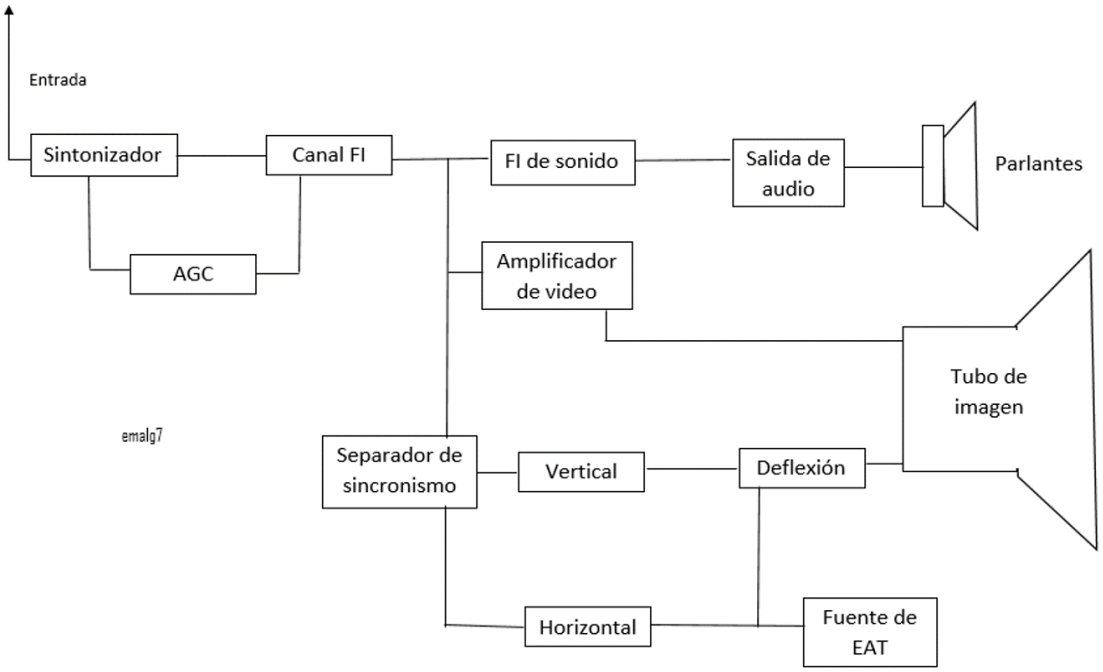
4.2.7.1. Sincronismo horizontal

Este sincronismo dura alrededor de 12 µseg, durante este tiempo el sistema de barrido de la pantalla logra posicionarse a la izquierda de esta misma para pintar una nueva línea. Esta es la información que acompaña a la señal del transmisor, al momento de recibir la señal, el circuito de barrido de la pantalla inicia su recorrido.

4.2.7.2. Sincronismo vertical

Este sincronismo dura alrededor de 1.6 mseg, durante este tiempo el sistema de barrido de la pantalla logra posicionarse en la parte alta de esta misma para iniciar un nuevo campo. Esta es la información que acompaña a la señal del transmisor, al momento de recibir la señal, se indica que debe iniciar un nuevo campo. Estos impulsos repetidos conforman un sincronismo vertical duran lo mismo que varias líneas.

Figura 72. **Diagrama de bloques de televisor**



Fuente: elaboración propia, realizado con Microsoft Word.

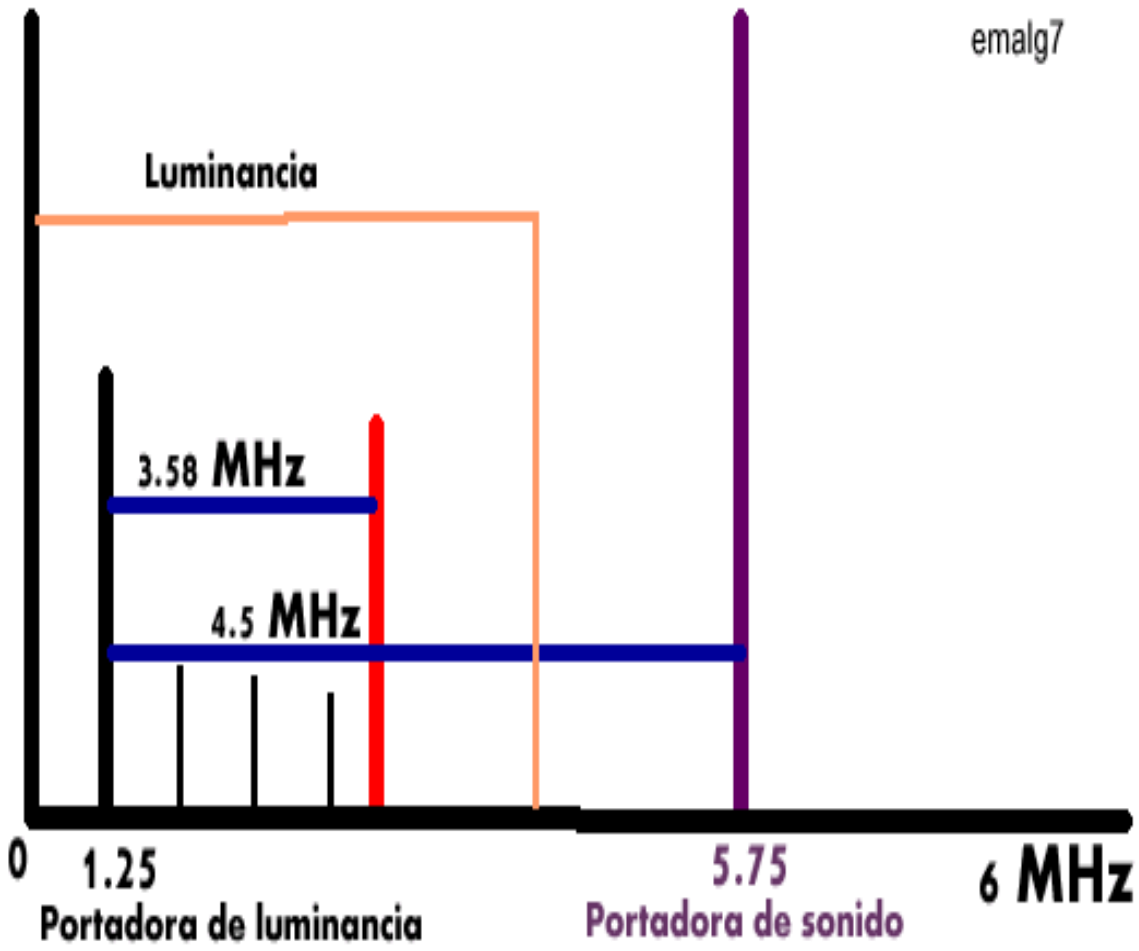
4.3. Norma NTSC

La norma NTSC que en español es Comité Nacional de Sistema de Televisión, es un sistema de televisión analógico, en este la imagen se compone de 525 líneas entrelazadas, mostrándose a una velocidad de 29.97 cuadros por segundo, utilizando una frecuencia de campo de 60 Hz.

Según la norma NTSC, se ha especificado las diferentes señales de información de una TV de la siguiente manera, ver figura 18. (Wikipedia, 2023)

- La banda total de una señal de información de TV debe ser de 6 MHz, en banda base.
- Señal de luminancia que viene a ser la imagen, está modulando en amplitud una portadora de 1.25 MHz, está luminancia genera un ancho de banda de 4.2 MHz.
- Señal de audio, está modulando en frecuencia una portadora que tiene una separación de 4.5 MHz de la portadora de luminancia.
- Señal de crominancia que viene a ser el color, está modulando en cuadratura una subportadora que tiene una separación de 3.57 MHz de la portadora de luminancia.

Figura 73. **Señales de la norma NTSC para una TV**



Fuente: elaboración propia, realizado con Macromedia Fireworks.

4.4. Frecuencias de portadora de video y audio

Frecuencia de la portadora de audio y de video asociadas a algunos canales de Televisión de banda baja que se observa en la tabla I y banda alta que se observa en la tabla II. (Wikipedia, 2023)

Tabla I. **Frecuencias VHF de banda baja**

Frecuencias VHF de banda baja (banda I) en MHz					
Canal	**Borde inferior**	**Portadora de video**	**Piloto ATSC**	**Portadora de audio**	**Borde superior**
2	54	55.25	54.31	59.75	60
3	60	61.25	60.31	65.75	66
4	66	67.25	66.31	71.75	72
(Brecha en el plan de la banda)					
5	76	77.25	76.31	81.75	82
6	82	83.25	82.31	87.75	88

Fuente: elaboración propia, realizado con Microsoft Word.

Tabla II. **Frecuencias VHF de banda alta**

Frecuencias VHF de banda alta (banda III) en MHz					
Canal	**Borde inferior**	**Portadora de video**	**Piloto ATSC**	**Portadora de audio**	**Borde superior**
7	174	175.25	174.31	179.75	180
8	180	181.25	180.31	185.75	186
9	186	187.25	186.31	191.75	192
10	192	193.25	192.31	197.75	198
11	198	199.25	198.31	203.75	204
12	204	205.25	204.31	209.75	210
13	210	211.25	210.31	215.75	216

Fuente: elaboración propia, realizado con Microsoft Word.

5. RECEPTOR DVB-T, SUS CARACTERÍSTICAS Y COSTO

Un Dongle DVBT-T es un sintonizador de televisión digital basado en el chipset RTL2832U, ver figura 74.

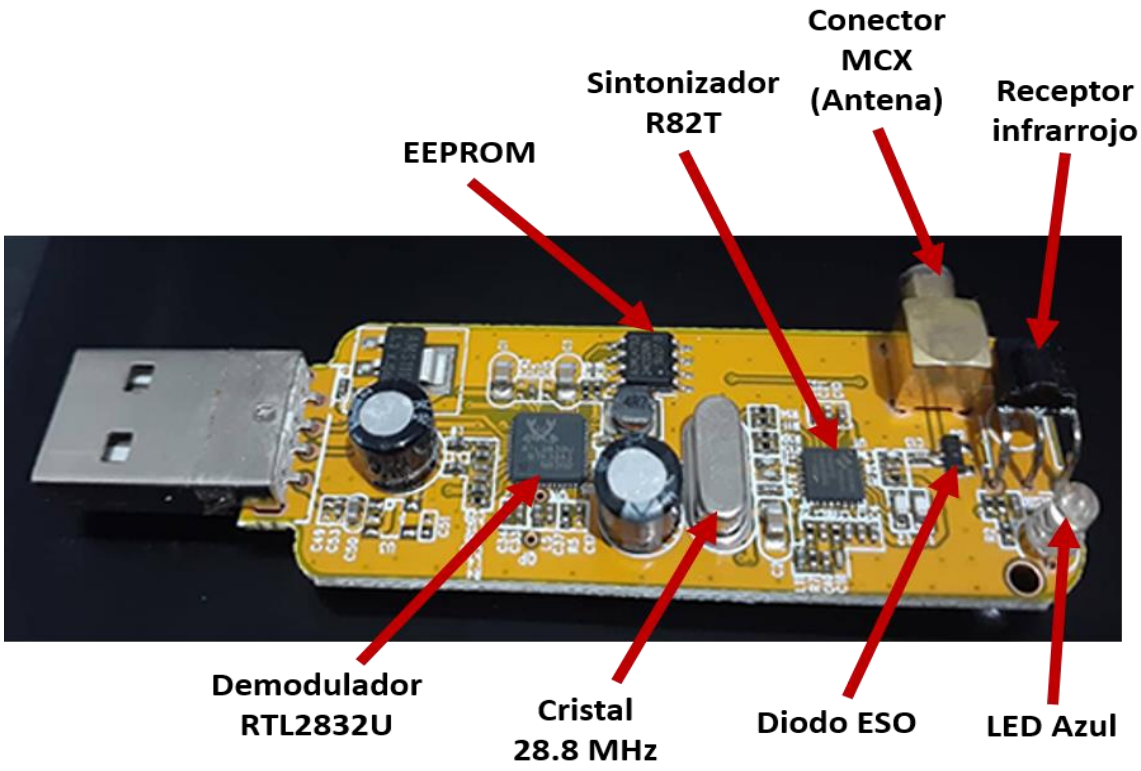
5.1. Características del *hardware* receptor DVB-T

- Interfaz: USB 2.0
- Formato: DVBT-TV DVD, radio FM / DAB-radio WMA.
- Frecuencia de recepción:
  - DVBT 48.25 - 863.25 MHz
  - Radio FM 87.5 - 108 MHz
  - Radio DAB banda L-1.452.960 - 1.490.624 KHz
  - VHF 174.928 – 239200 KHz
- Tasa de muestreo 3.2 MS/s
- Muestras de 8 bits.
- Sistema de televisión: DVB-T y DAB.
- Ancho de banda: 6/7/8 MHz
- Fuente de alimentación: Bus USB. (Tuners, 2013)

5.2. Costo

El DVB-T tiene un precio de aproximadamente $ 13 dólares USD al realizar la compra en línea, esto en varias plataformas, el precio dependerá de la situación de varios factores en el momento en que se solicite el dispositivo.

Figura 74. **Partes del *hardware* receptor DVB-T USB dongle con RTL2832U**



Fuente: elaboración propia, realizado con Microsoft Word.

6. PRÁCTICAS DE LABORATORIO DE COMUNICACIONES 3

A continuación, se proponen las actividades a realizar para las prácticas de laboratorio de Comunicaciones 3, lo cual queda a discreción de quien esté a cargo de este laboratorio.

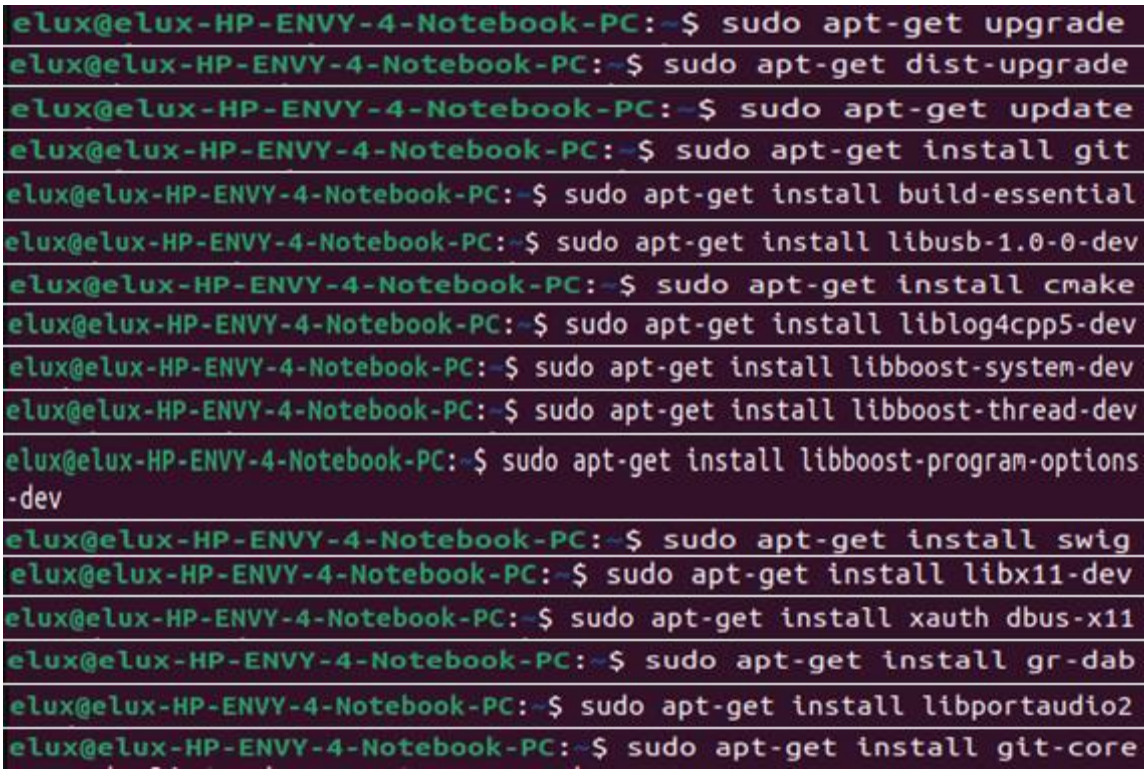
6.1. Primera práctica. Instalación de GNU Radio

Es importante mencionar que se está utilizando el sistema operativo Ubuntu V21.10 o puede usarse las versiones más recientes.

A continuación, abra la línea de comandos del sistema operativo Ubuntu e instale cada paquete.

Debe instalar cada paquete con los comandos que se indican en la siguiente imagen. Con esto se evitará tener errores en lo que se esté realizando a lo largo de las prácticas ya que algunos paquetes no son directamente del programa a utilizar, pero son llamados en algunos procesos y por ello es necesario instalarlos. (Studylib, 2013)

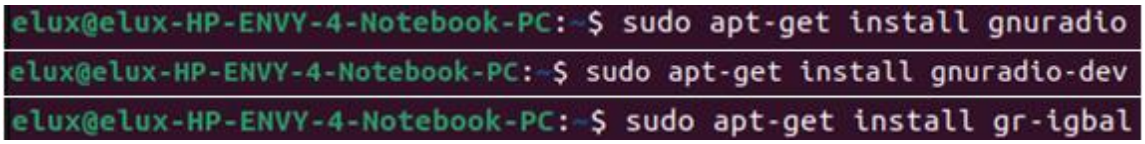
Figura 75. **Instalación de paquetes**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Instalación de GNU Radio: a continuación, debe instalar los paquetes que se indican en la imagen, el primero es la instalación directa de GNU Radio.

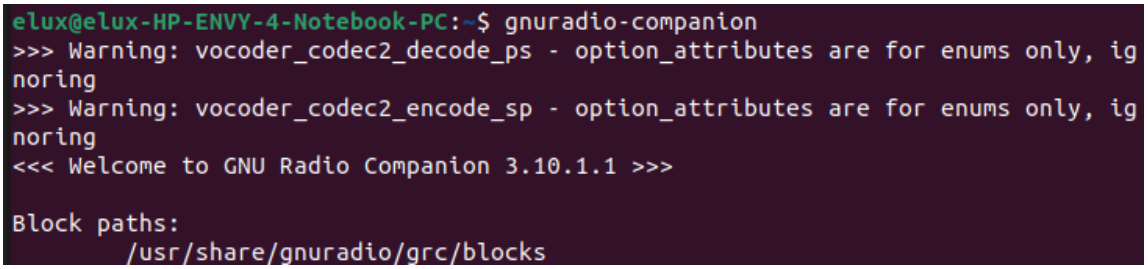
Figura 76. **Instalación de GNU Radio**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

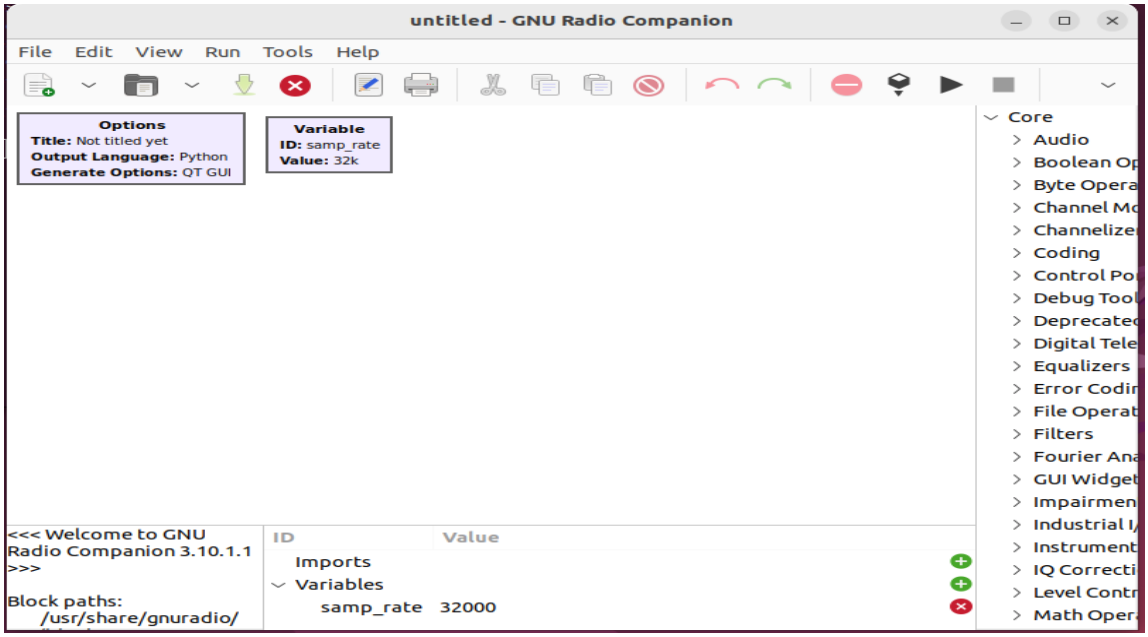
Para empezar a utilizar GNU Radio, escriba el siguiente comando, esto le mostrará la ventana de GNU Radio.

Figura 77. **Comando para utilizar GNU Radio**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Figura 78. **Ventana de GNU Radio**



Fuente: elaboración propia, realizado con GNU Radio.

6.2. Segunda práctica. Uso de bloques en GNU Radio

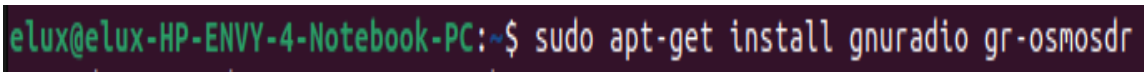
Se le solicita realizar un recepto de radio FM y un receptor de audio de televisión, se le indicarán los bloques que se estarán utilizando en cada receptor.

Para saber las características de cada bloque que se estará utilizando, puede observar la información en el capítulo 2 a partir de la página 10.

Para estos receptores necesitamos un bloque que se encargará de recibir la señal, este no viene directamente en GNU Radio, por lo que hay que instalarlo. A continuación, se muestran los comandos para instalar el bloque receptor.

Instalación de OSMOSDR.

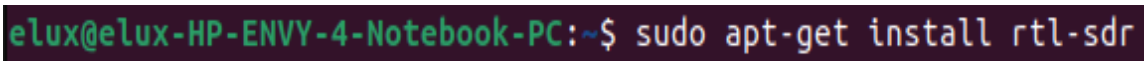
Figura 79. **Instalación de OSMOSDR**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

Instalación de RTL.

Figura 80. **Instalación de RTL**



Fuente: elaboración propia, realizado con línea de comandos del sistema operativo Ubuntu.

6.2.1. Receptor de radio FM

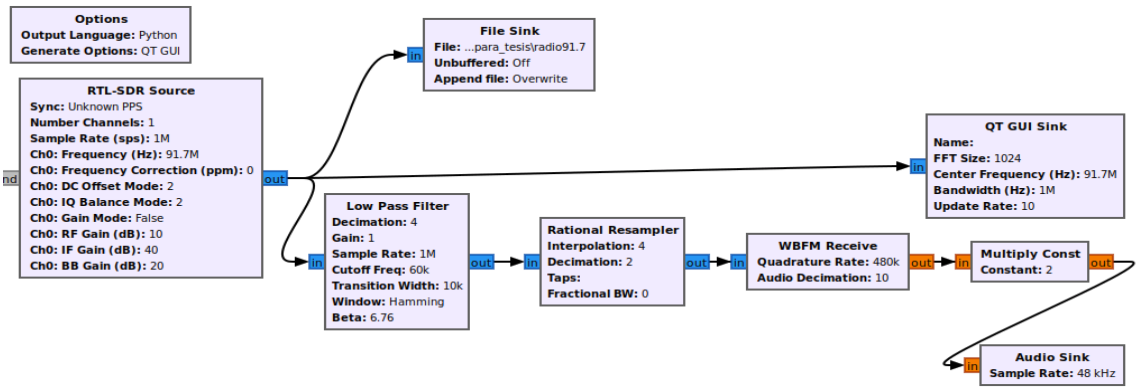
Se le solicita realizar un receptor de radio FM, debe realizar el proceso de recepción y reproducción de la señal, este proceso lo puede encontrar en el capítulo 4 a partir de la página 59. Debe conectar el hardware receptor DVB-T al puerto USB. Se le indican a continuación los bloques a utilizar en GNU Radio.

- Bloque RTL-SDR Source
- Bloque File Sink
- Bloque Low Pass Filter
- Bloque Rational Resamppler
- Bloque WBFM Receive
- Bloque Multiply Const
- Bloque Audio Sink
- Bloque QT GUI Sink

La frecuencia del bloque receptor será de la señal que usted quiere recibir, en este caso se está utilizando la frecuencia 91.7 MHz que tiene relación con la radio Rhema Stereo de la ciudad de Guatemala.

En la siguiente imagen encontrara el ejemplo de cómo debe ir cada bloque en GNU Radio.

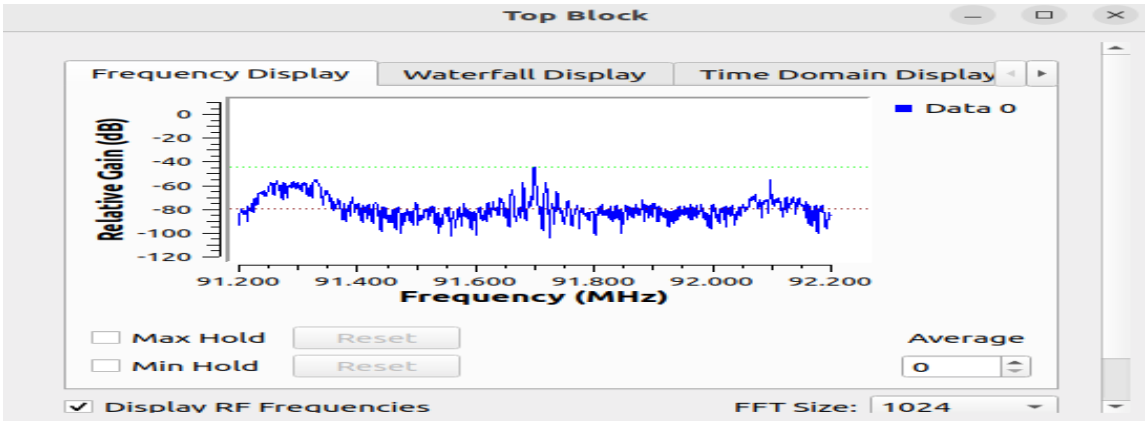
Figura 81. **Receptor FM**



Fuente: elaboración propia, realizado con GNU Radio.

Resultado al ejecutar el programa en GNU Radio: como se tiene un bloque de audio, se debe escuchar lo que está transmitiendo la radio, también se utilizó un bloque QT GUI Sink por lo que se tiene una ventana emergente que muestra información de la señal que se recibe en la frecuencia central especificada.

Figura 82. **Información de la señal recibida**



Fuente: elaboración propia, realizado con GNU Radio.

6.2.2. Receptor de audio de televisión

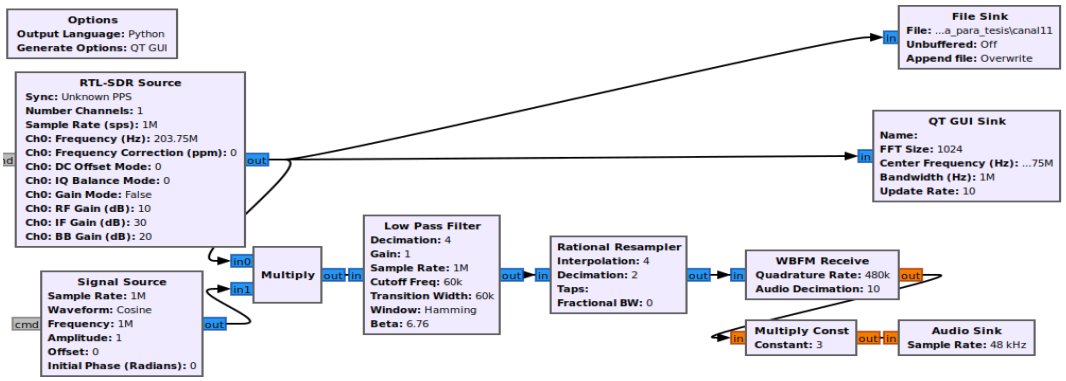
Se le solicita realizar un receptor de audio de televisión, debe realizar el proceso de recepción y reproducción de la señal. Debe conectar el hardware receptor DVB-T al puerto USB. Se le indican a continuación los bloques a utilizar en GNU Radio.

- Bloque RTL-SDR Source
- Signal Source
- Multiply
- Bloque File Sink
- Bloque Low Pass Filter
- Bloque Rational Resamppler
- Bloque WBFM Receive
- Bloque Multiply Const
- Bloque Audio Sink
- Bloque QT GUI Sink

La frecuencia del bloque receptor será de la señal que usted quiere recibir, en este caso se está utilizando la frecuencia 203.75 MHz que tiene relación con el audio del canal 11 de televisión. Las frecuencias de la portadora de audio de los canales de televisión los puede encontrar en las tablas I y II del capítulo 4 en la sección 4.4 en la página 66.

En la siguiente imagen encontrara el ejemplo de cómo debe ir cada bloque en GNU Radio.

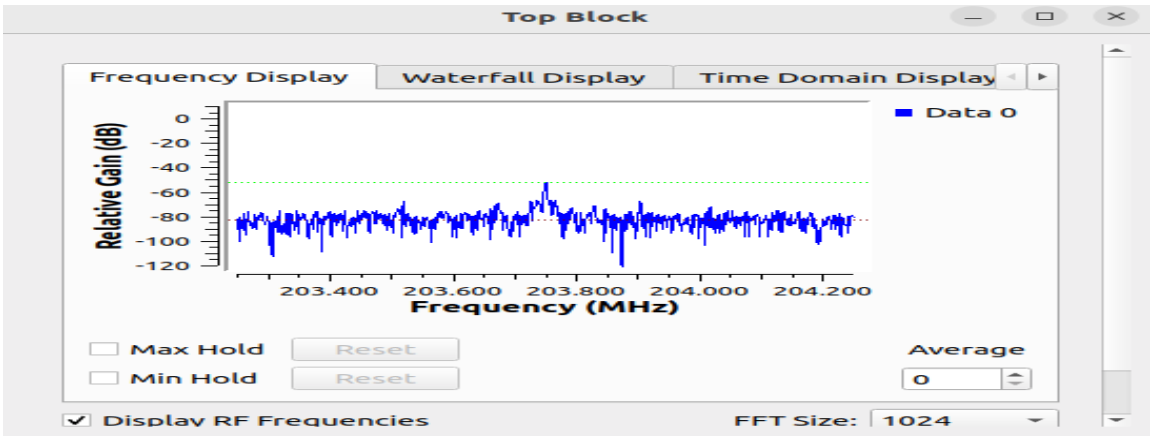
Figura 83. **Receptor de audio de televisión**



Fuente: elaboración propia, realizado con GNU Radio.

Resultado al ejecutar el programa en GNU Radio: como se tiene un bloque de audio, se debe escuchar el audio que está recibiendo, también se utilizó un bloque QT GUI Sink por lo que se tiene una ventana emergente que muestra información de la señal que se recibe en la frecuencia central especificada.

Figura 84. **Información de la señal recibida**



Fuente: elaboración propia, realizado con GNU Radio.

6.3. Tercera práctica. Creación de bloques personalizados para GNU Radio y uso de estos mismos

Se propone realizar bloques personalizados con distintas funciones matemáticas, lo cual queda a discreción de la persona encargada del laboratorio de Comunicaciones 3 ya que pueden solicitarse bloques personalizados que puedan ser útiles para el proyecto final del laboratorio.

6.3.1. Bloque personalizado de suma y resta

Se le solicita crear un bloque personalizado de Python que sume y reste. Todos los pasos para crear el bloque personalizado de suma y resta, desde la creación del nuevo módulo OOT donde se crea el bloque personalizado, los puede ver en el capítulo 3 en su sección 3.1 a partir de la página 28.

En esta sección se le informa y muestra cada uno de los pasos a seguir, los comandos y programación que debe u.tilizar en la línea de comandos del sistema operativo Ubuntu para obtener así el bloque personalizado en la ventana de GNU Radio y así poder utilizarlo.

6.3.2. Bloque personalizado de multiplicación y división

Se le solicita crear un bloque personalizado de C++ con la función de multiplicación y división. Todos los pasos para crear el bloque personalizado de multiplicación y división, desde la creación del nuevo módulo OOT donde se crea el bloque personalizado, los puede ver en el capítulo 3 en su sección 3.2 a partir de la página 42.

En esta sección se le informa y muestra cada uno de los pasos a seguir, los comandos y programación que debe utilizar en la línea de comandos del sistema operativo Ubuntu para obtener así el bloque personalizado en la ventana de GNU Radio y así poder utilizarlo.

6.3.3. Bloque personalizado indicado por el instructor

El instructor les solicitará un bloque personalizado con una función en específico. Puede que se le solicite bloques que se puedan utilizar en el proyecto del semestre.

Se propone realizar bloques personalizados para recibir video de una señal de tv. Se le proporciona información del proceso de una tv en el capítulo 4 en la sección 4.2 a partir de la página 61. Se presenta información de la norma NTSC en el capítulo 4 en su sección 4.3 a partir de la página 64. Se le proporciona la frecuencia portadora de video de tv en el capítulo 4 en su sección 4.4 en las tablas I y II en la página 66.

CONCLUSIONES

1. Se describen las características de los bloques utilizados en GNU Radio, proporcionando la información teórica e imágenes que apoyan visualmente el aprendizaje en el estudiante.

2. Se desarrollan los pasos específicos para crear los bloques personalizados para GNU Radio, en estos se proporciona los comandos a ejecutar, se utilizan imágenes con el fin de que el estudiante tenga todo lo necesario y así pueda crear nuevos bloques de acuerdo con las necesidades que se le presenten.

3. Se entrega una propuesta de actividades a realizar en las diferentes prácticas, se incluye información que el estudiante pudiera necesitar a lo largo de estas mismas.

4. Se proporciona la información de las características y costo del hardware que se implementa junto con el programa GNU Radio.

RECOMENDACIONES

1. Seguir las instrucciones tal y como se proponen al momento de trabajar las practicas ya que está pensando para que el estudiante ponga en práctica todo lo enseñado y se da las páginas donde puede encontrar la información de apoyo, preferiblemente si el estudiante realiza individualmente las prácticas para que cada uno pueda desarrollar sus habilidades.

2. Realizar grupos de tres estudiantes para el proyecto de laboratorio de Comunicaciones 3, para que la carga de trabajo sea equitativa, no mayor a tres para que todos los integrantes puedan poner en práctica el uso de GNU Radio.

3. Tener versiones posteriores a este documento debido a que el software libre GNU Radio es muy amplio y existen bloques que no se especificaron, y así seguir ampliando la información de las características de otros bloques que el estudiante pueda necesitar dentro del *software*.

REFERENCIAS

1. ECURED. (s.f.). *Receptor de radio de FM*. Recuperado de ECURED: https://www.ecured.cu/Receptor_de_radio_de_FM#Diagrama_en_bloques.

2. Electrónica Fácil. (2004). *MODULACIÓN DIGITAL :FSK – PSK - QAM*. Recuperado de Electrónica Fácil: https://www.electronicafacil.net/tutoriales/MODULACION-DIGITAL-FSK-PSK-QAM.html.

3. Jaimes, R., & Lazcano, S. (2 de septiembre de 2019). *SDR y GNU Radio como plataforma para un laboratorio de comunicaciones digitales*. Recuperado de Computación e Informática: https://www.redalyc.org/journal/5122/512261374007/html/.

4. Lincango, F. (2018). *Implementación de un Prototipo de Sistemas de Comunicación Inalámbrico OFDM en SDR.* Quito: Escuela Politécnica Nacional. Recuperado de https://bibdigital.epn.edu.ec/bitstream/15000/19827/1/CD-9231.pdf.

5. Modulación Digital. (9 de Noviembre de 2010). *Resumen Modulación Digital*. Recuperado de Modulación Digital: http://modulaciondigital.blogspot.com/2010/11/resumen-modulacion-digital.html.

6. Monografias.com S.A. (s.f). *Diagrama de bloque de un TV*. Recuperado de Monografias.com S.A.: https://www.monografias.com/trabajos105/diagrama-bloque-tv/diagrama-bloque-tv.

7. Studylib. (2013). *Instalación.* Recuperado de Studylib: https://studylib.es/doc/6446247/1.-instalaci%C3%B3n-este-proceso-de-instalaci%C3%B3n-se-realizara-d....

8. Tuners. (6 de junio de 2013). *New USB 2.0 DVB-T STICK RTL2832U+R820T TV-DVD(MPEG-2)SDR FM radio / DAB radio-WMA -Black*. Recuperado de Tuners: https://toptuners723.blogspot.com/2013/06/new-usb-20-dvb-t-stick-rtl2832ur820t-tv.html.

9. wiki.gnuradio. (16 de septiembre de 2019). *Category: Block Docs*. Recuperado de wiki.gnuradio: https://wiki.gnuradio.org/index.php/Category:Block_Docs.

10. wiki.gnuradio. (26 de enero de 2023). *Creación de OOT de C++ con gr-modtool*. Recuperado de wiki.gnuradio: https://wiki.gnuradio.org/index.php?title=Creating_C%2B%2B_OOT_with_gr-modtool.

11. wiki.gnuradio. (3 de febrero de 2023). *Creación de Python OOT con gr-modtool*. Recuperado de wiki.gnuradio: https://wiki.gnuradio.org/index.php?title=Creating_Python_OOT_with_gr-modtool.

12. Wikipedia. (17 de diciembre de 2020). *GNU Radio*. Recuperado de Wikipedia: https://es.wikipedia.org/wiki/GNU_Radio.

13. Wikipedia. (23 de diciembre de 2022). *Filtro digital*. Recuperado de Wikipedia: https://es.wikipedia.org/wiki/Filtro_digital.

14. Wikipedia. (17 de marzo de 2023). *NTSC*. Recuperado de Wikipedia: https://es.wikipedia.org/wiki/NTSC.

15. Wikipedia. (13 de febrero de 2023). *Pan-American television frequencies*. Recuperado de Wikipedia: https://en.wikipedia.org/wiki/Pan-American_television_frequencies.