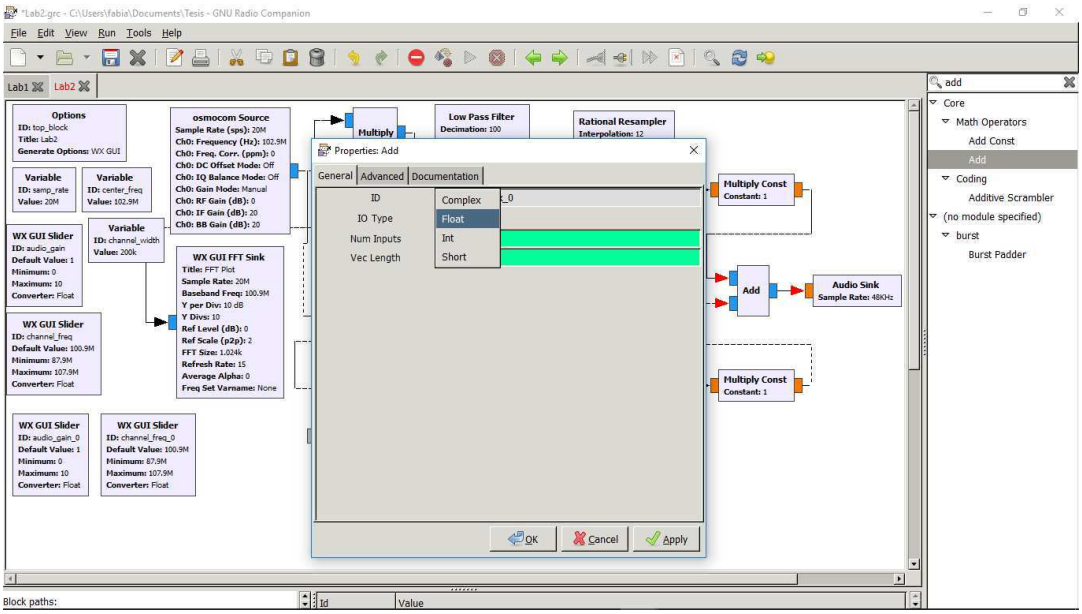

Figura 109. 2lab Paso 6a.

Paso 7. Si se observan los dos diagramas se evidencia que los nombres en la copia son iguales a los originales excepto que se le añade "_0" por ejemplo "center_freq_0", ahora bien, lo que se debe hacer es empezar a cambiar estos nombres en los dos diagramas para no tener problemas más adelante, en el diagrama uno se añadirá un 1 a los nombres como se muestra en las figuras 110, 111, 112, 113 y 114 y en el diagrama copiado se borrara el _0 y se pondrá un 2 como se muestran en las figuras 115, 116, 117 y 118, se debe empezar cambiando los nombres en los Slider, allí se empezaran a poner en rojo los títulos en los diferentes bloques y en estos son lo que también se deben realizar las modificaciones en el nombre del parámetro necesario hasta que todos regresen a su estado normal sin presentar errores.

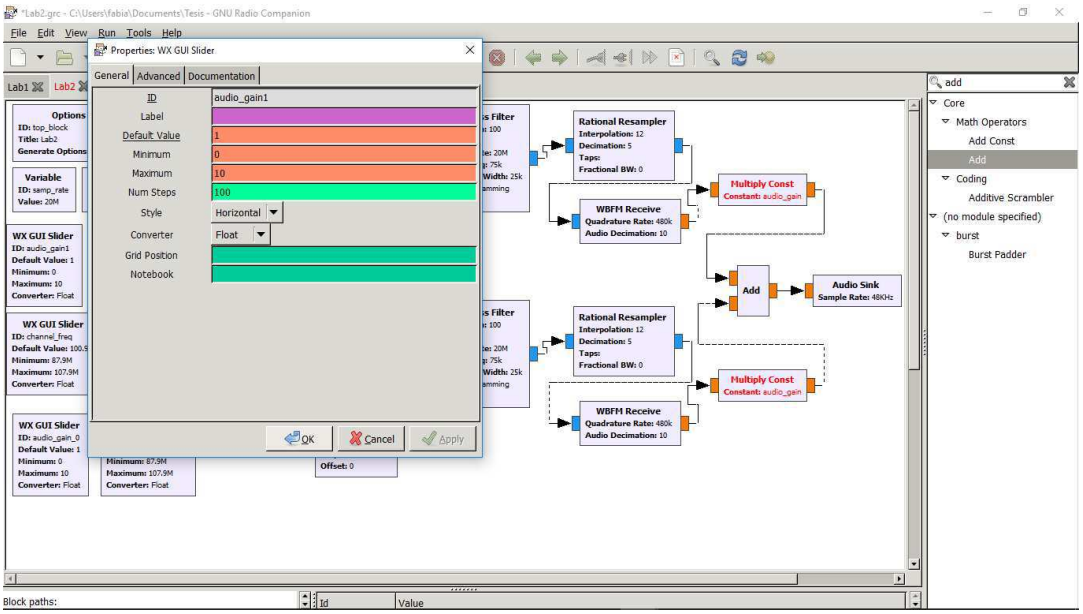

Figura 110. 2lab Paso 7a.

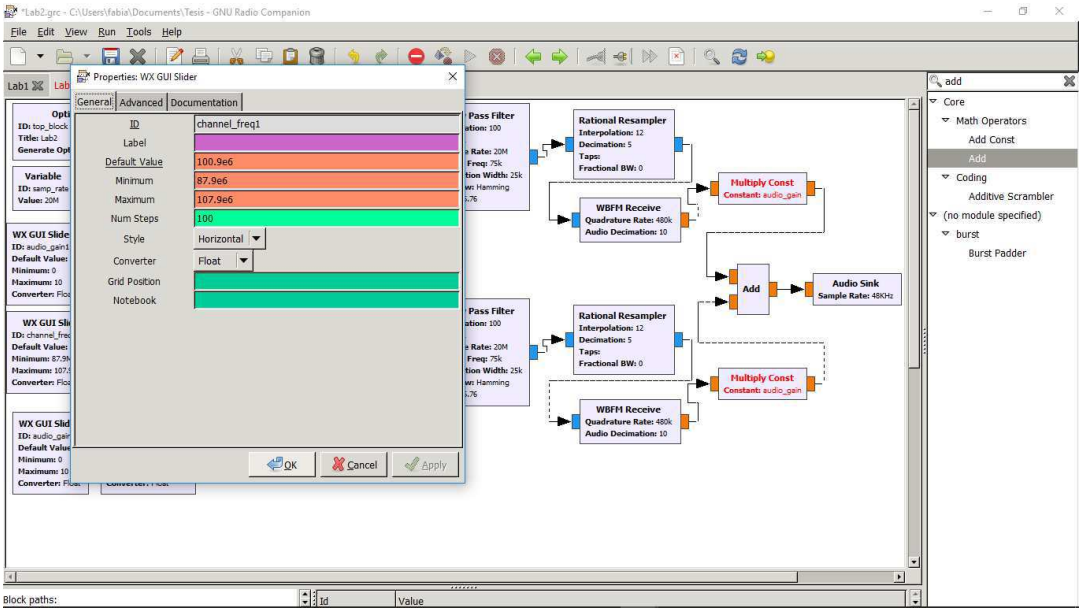

Figura 111. 2lab Paso 7b.

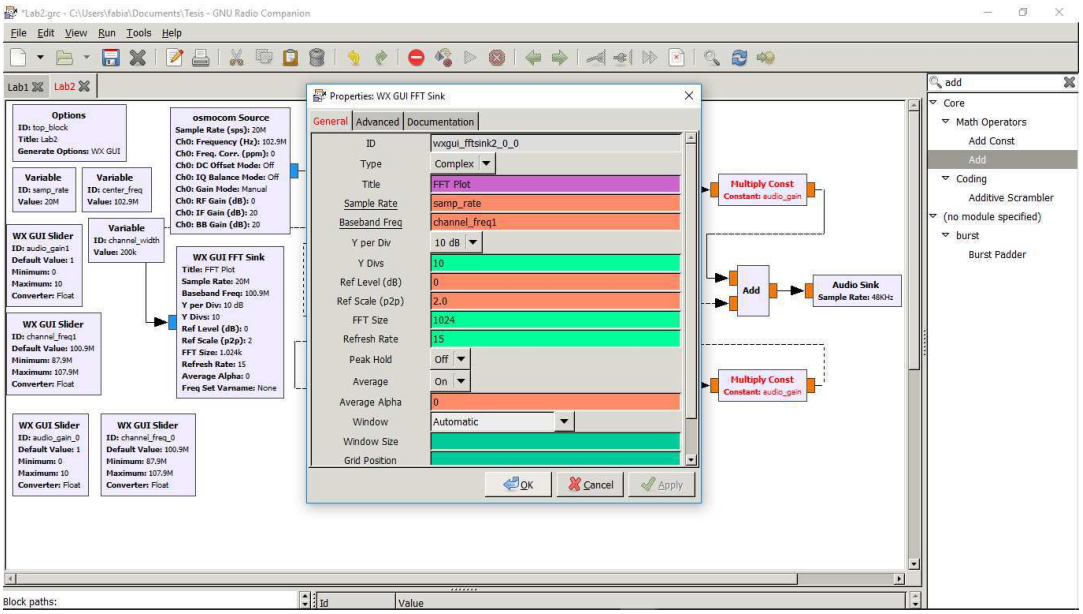

Figura 112. 2lab Paso 7c.

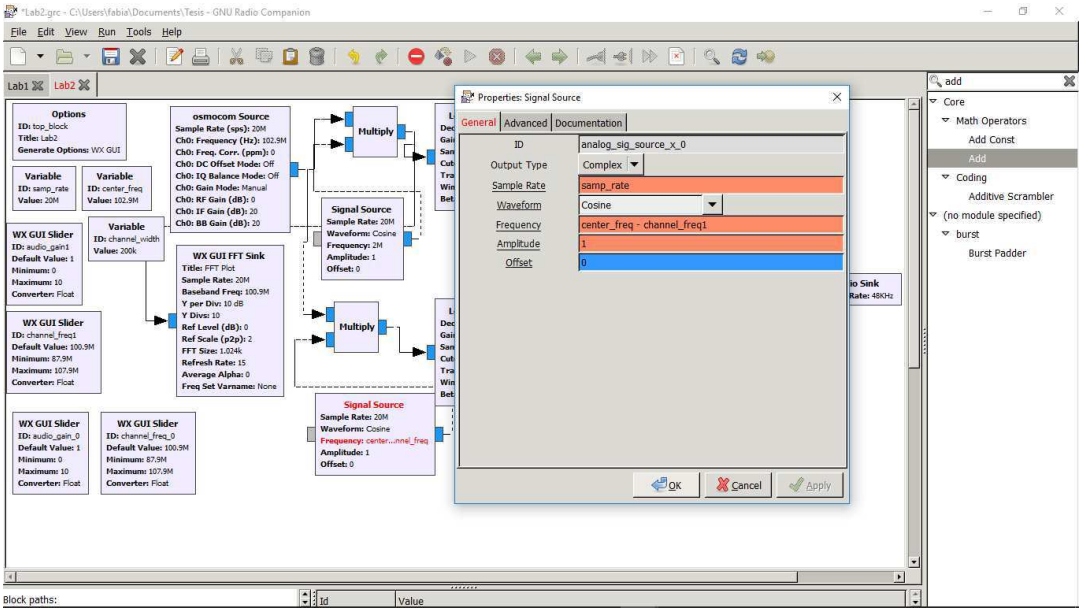

Figura 113. 2lab Paso 7d.

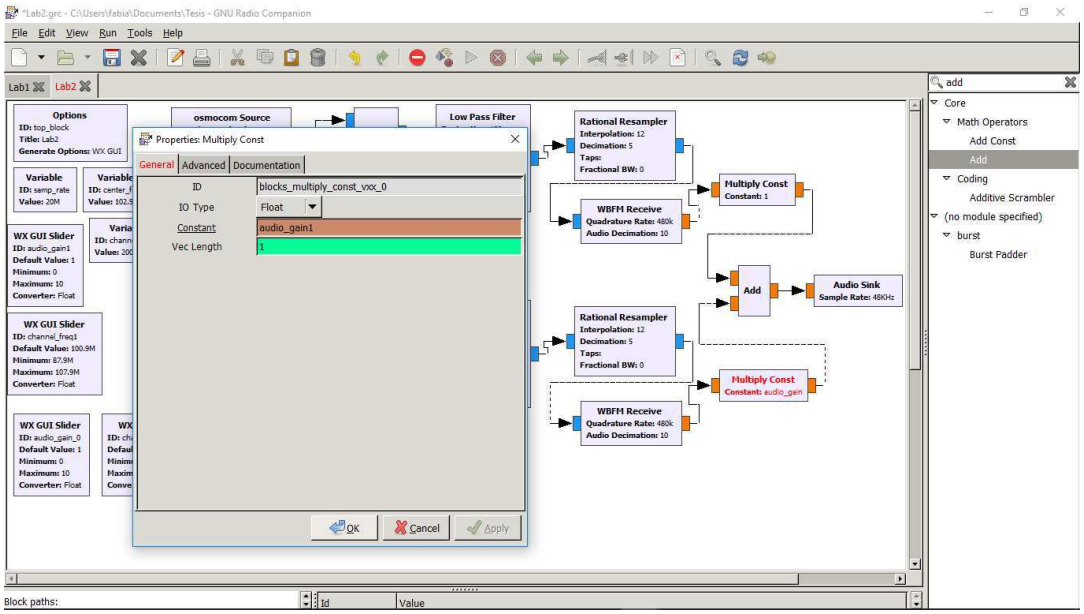

Figura 114. 2lab Paso 7e.

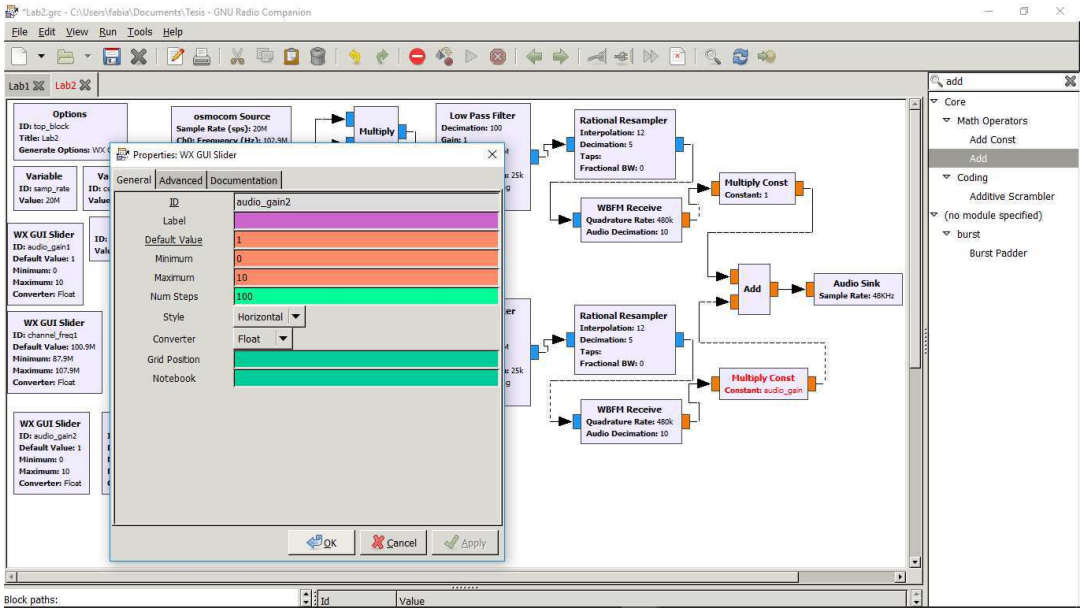

Figura 115. 2lab Paso 7f.

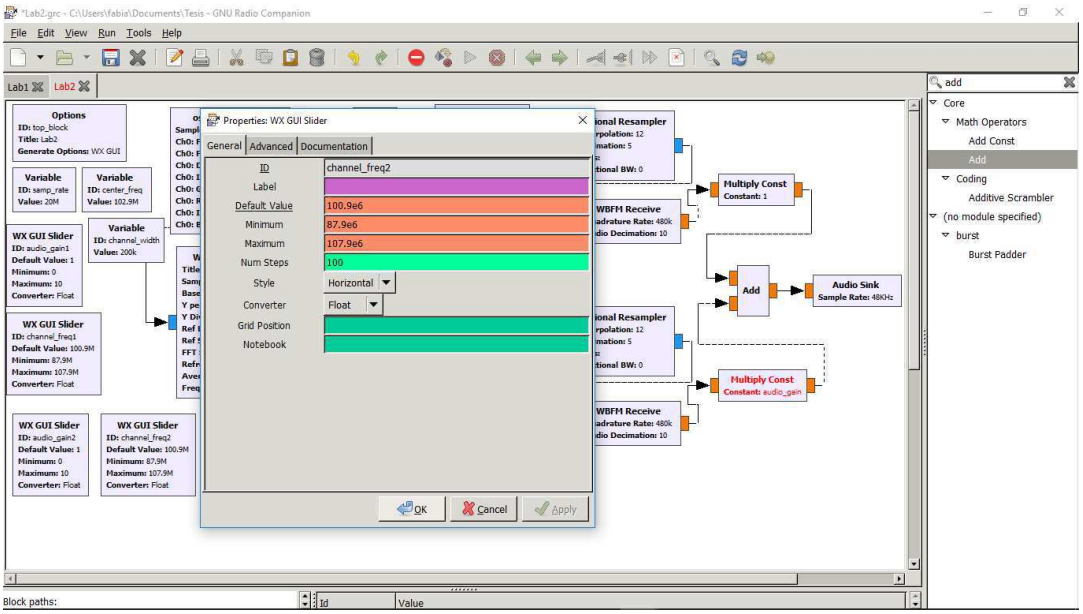

Figura 116. 2lab Paso 7g.

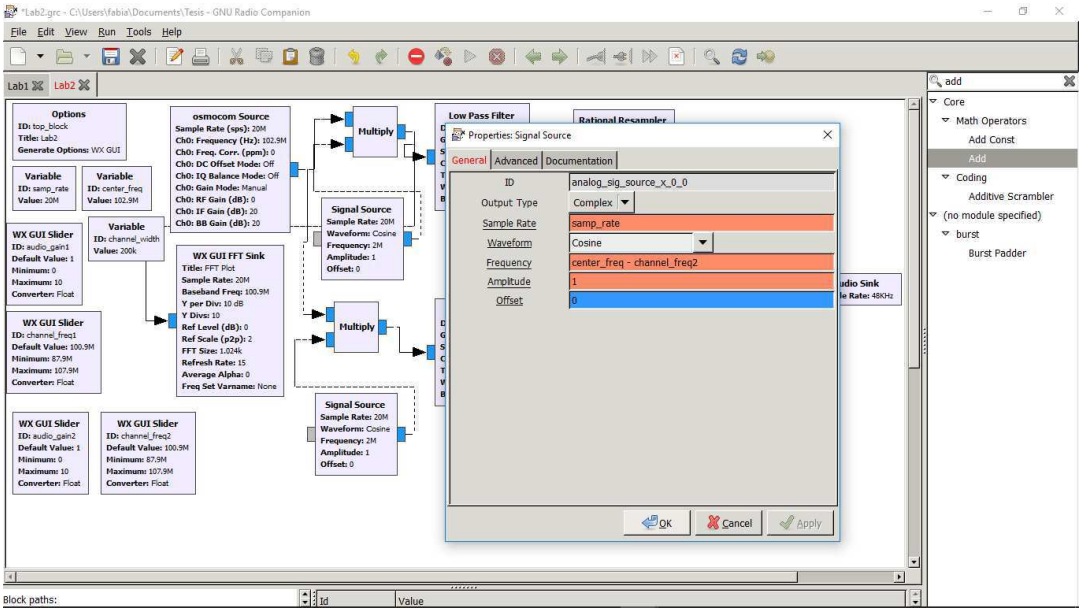

Figura 117. 2lab Paso 7h.

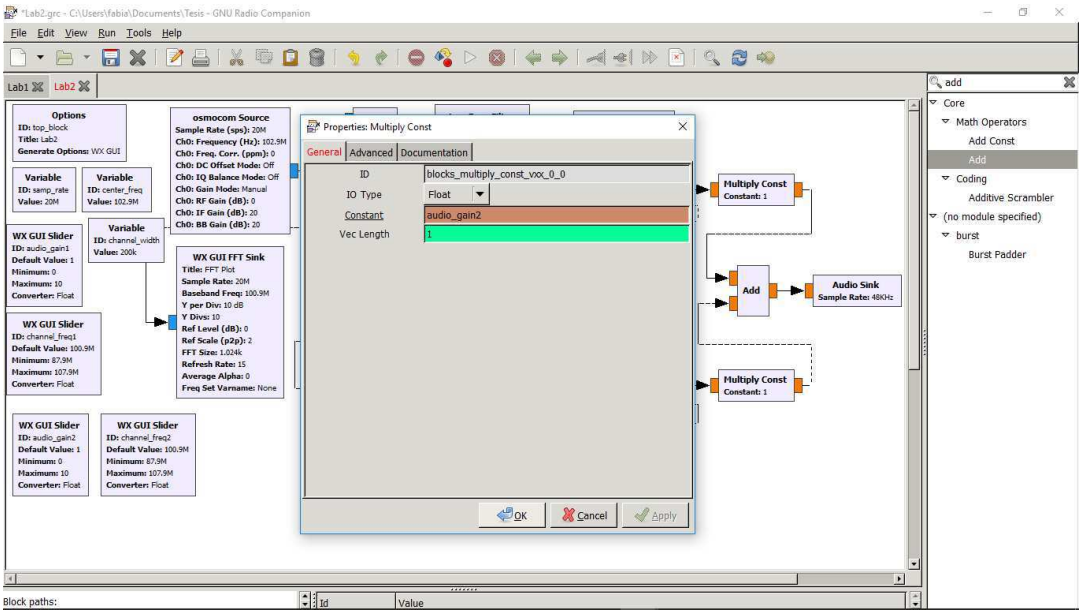

Figura 118. 2lab Paso 7i.

Paso 8. Ahora se va a realizar otro cambio en los bloques Slider del segundo diagrama para que estos que queden con los mismos valores del primero y poder notar una diferencia, en el Slider channel_freq pondremos la frecuencia 103.9e6 y en el Slider audio_gain se establecerá el valor en 0, en las figuras 119 y 120 se muestran estas configuraciones.

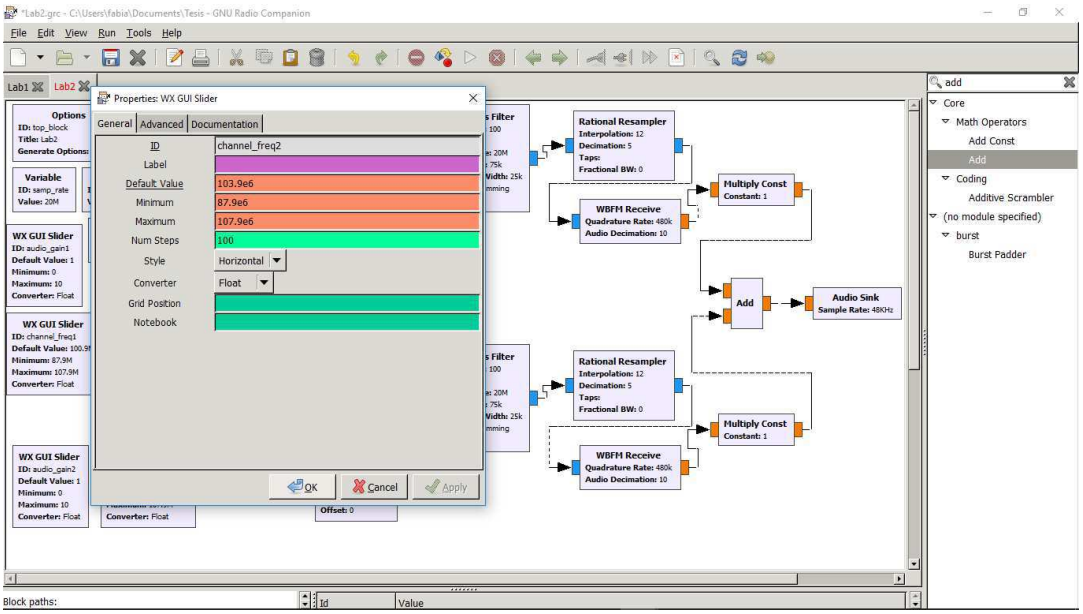

Figura 119. 2lab Paso 8.

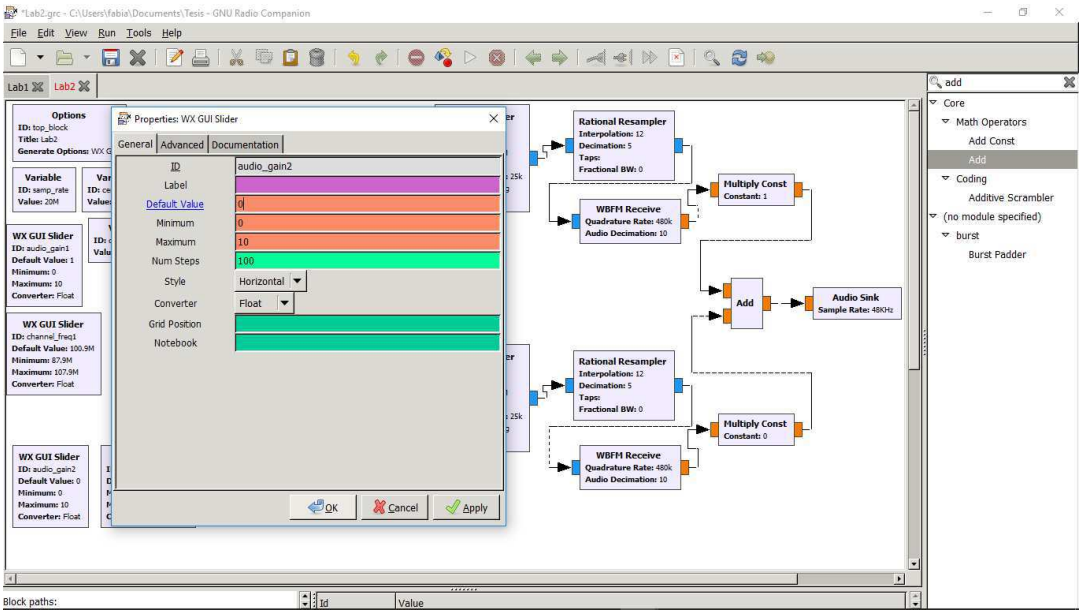

Figura 120. 2lab Paso 8a.

Al poner la ganancia en 0 hará que solo se escuche una emisora al momento de iniciar el grafico FFT Plot y luego con el deslizador se podrá ir graduando el volumen para escuchar la otra emisora también que por defecto será 103.9Mhz, recordemos que la emisora por defecto que saldrá en el primer diagrama será 100.9Mhz con una ganancia de 1 que quiere decir que esta emisora será la que se empezara a escuchar al ejecutar la ventana FFT Plot, luego ya se podrá variar el volumen y las frecuencias de los dos diagramas.

Paso 9. Lo siguiente será realizar la prueba para saber si las configuraciones están funcionando correctamente, para esto ejecutaremos la ventana FFT Plot. Para este caso y como se mencionó en el paso 0 efectivamente al ejecutar analizador este no funcionó correctamente con la tasa de muestreo establecida en 10 millones, esto se debe a la cantidad de recursos que consume el diagrama y que a dicha tasa el computador no soporta, por esta razón se le redujo el valor a la tasa de muestreo, consiguiendo que el más alto posible para que se ejecutara correctamente fuera en 7 millones dando como resultado lo que se aprecia en la Figura 121. Si a

ustedes les sucede lo mismo y el flujo no se ejecuta correctamente a la tasa que tienen establecida previamente lo único que debe es ajustarla al valor máximo que les permita que todo fluya y se ejecute correctamente, esto lo único que va a alterar es el ancho de banda que se verá en la ventana, como se sabe a 20 millones se observa todo el ancho de banda del espectro asignado para la radiodifusión en FM el cual es de 20Mhz, si funciona en 10 millones se observara la mitad del ancho de banda, es decir 10Mhz, si funciona en 7 millones como tasa de muestreo solo se podrá observar el 35% del ancho de banda total, es decir 7Mhz y así sucesivamente, sin embargo esto no quiere decir que la HackRF no sintonice las estaciones de radio fuera del espectro que se puede observar, precisamente, el no poder observar es la única afectación, pero si se elige con el deslizador una frecuencia que no logramos observar en la ventana no importa, igual la escucharemos sin problemas al igual que las frecuencias que si estan en nuestro rango de observación.

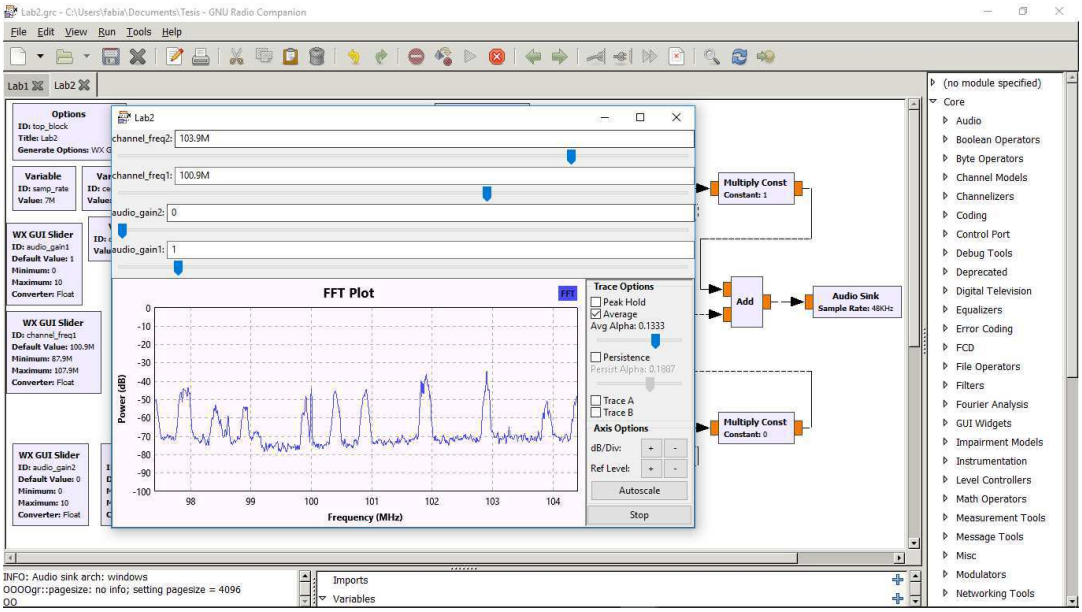

Figura 121. . 2lab Paso 9.

Este es el fin del segundo laboratorio basado al igual que el primero en uno de los videos del curso de SDR realizados por el Sr. Osman, M., (2015) donde se evidencia que gracias a la HackRF en conjunto con el software GRC es posible la recepción de todo el ancho de banda del espectro asignado a la radio FM, es decir, que este hardware logra la recepción hasta un ancho de banda de 20Mhz y que permite escuchar más de una emisora a la vez gracias al entorno GRC sin necesidad de manipular el hardware si no solo con configuraciones de software.

Hasta este punto como pueden evidenciar se ha trabajado netamente con señales analógicas las cuales son recibidas por la Hack RF y digitalizadas para su tratamiento en el software GRC, para recordar las señales analógicas son continuas, son funciones que varían a lo largo del tiempo, las señales digitales son similares solo que están hechas de valores discretos a lo largo del tiempo, es decir que cuando tomamos una señal analógica y la digitalizamos lo que se hace es tomar muestras en diferentes puntos de la señal análoga, esta muestras se convierten en valores discretos, lo que significa que entre más número de muestras se tomen más parecida será la señal digital a la señal análoga ya que los valores discretos estarán más cerca unos de los otros, lo que permitirá una mayor exactitud y calidad al momento de reconstruir la información (demodular).

Para recalcar y como se dijo anteriormente el objetivo de este trabajo y las practicas realizadas fue el brindarle a los estudiantes e interesados las primeras bases necesarias en el manejo del entorno GRC en conjunto con la Hack RF, para que puedan realizar laboratorios más complejos, practicas mucho más avanzadas que demanden un flujo aún más complejo que el hecho en este segundo laboratorio, pero todo partiendo de la explicación de los componentes tanto del hardware y el software GRC presentada en este trabajo junto con las prácticas de laboratorio.

Esta práctica de laboratorio al igual que la primera también fue documentada en video a modo de guía, este video fue subido al sitio web YouTube con el siguiente link de enlace….

9. CONCLUSIONES

- Se ha logrado documentar satisfactoriamente la historia del desarrollo de la tecnología SDR, para un mejor entendimiento en cuanto al uso del mismo.
- A pesar de la falta de información en español acerca de SDR, puesto que en la gran mayoría de los diferentes documentos existentes actualmente se encuentran en idioma ingles; lo que representaba una barrera para la realización del proyecto y más aún para la realización de futuros proyectos, se ha conseguido consolidar una amplia documentación en español que servirá de apoyo para el diseño de futuras soluciones con SDR.
- Se llevaron a cabo dos laboratorios prácticos cuya finalidad fue dar introducción a las aplicaciones de SDR, estos laboratorios permitieron enteder de una manera mas profunda el manejo de los recursos de SDR y poder visualizar lo mucho que estos sistemas permiten realizar a nivel académico y profesional.
- Se realizó un documento en forma de guía paso a paso por cada uno de los laboratorios realizados. Estas guías les brindaran a los interesados las bases principales que les permitirán la realización de futuros proyectos.
- Los dos laboratorios también fueron documentados en forma de videos que junto con las guías y la documentación de SDR les proporcionaran a los interesados unas bases teórico-prácticas importantes para la reproducción e implementación de estas dos prácticas, así como de nuevas aplicaciones SDR más elaboradas y complejas.

10. REFERENCIAS

Alfaro, G. R. (16 de Octubre de 2011). *Telecomunicaciones electronicas*. Obtenido de Modulacion-Demodulacion: https://sites.google.com/site/telecomunicacionesmegatec/home/modulacion-demodulacion

AMSAT-UK. (2011). *Sitio Web Oficial de FUNcube Dongle,*.

ANE. (18 de Febrero de 2016). *ane.gov*. Obtenido de https://www.ane.gov.co/index.php/2015-12-08-19-09-44/13-preguntas-y-respuestas-frecuentes/130-clasificacion-tematica-acceso-fijo-inalambrico

Azurdia, J. R. (2016). Obtenido de http://slideplayer.es/slide/10939671/

AIRSPY. (2016). *AIRSPY*. Obtenido de airspy.com

BILIĆ, D. G. (31 de Octubre de 2014). *welivesecurity*. Obtenido de https://www.welivesecurity.com/la-es/author/dgiusto/

Blossom., E. (2011). *GNU Radio: the open-source software defined Radio,*. Obtenido de http://dev.emcelettronica.com/gnu-radio-open-source-software-defined-radio http://gnuradio.org/redmine/wiki/gnuradio.

Bob Stewart, K. B. (2015). *Software Defined Radio Using MATLAB & Simulink and the RTL-SDR.* Glasgow, Scotland, UK: Deparment of Electronic and Electrical Engineering.

Comin1415cs. (s. f.). *ondas de radio de baja frecuencia*. Obtenido de https://sites.google.com/site/comin1415cs/medios-de-trasmisi/ondas-de-radio-de-baja-frecuencia

19 de Junio de 2012). Obtenido de https://es.wikipedia.org/wiki/Receptor_superheterodino

Congreso de la Republica . (23 de Enero de 2009). LEY 1286 DE 2009 . Bogota.

Convertidores Digitales DAC y analogico a Digital ADC. (2017). Obtenido de http://www.el.uma.es/marin/Tema7_practica4.pdf

DocBook. (08 de Marzo de 2013). *GNU Radio Compainion: ece.uvic.ca*. Obtenido de http://www.ece.uvic.ca/~elec350/grc_doc/index.html

Drake M, J. (2005). *Ruidos e Interferencias: Técnicas de reducción.* Cantabria.

E&Q Engineering. (2010). *E&Q Engineering Solutionsand Innovation*. Obtenido de Radio Definida por Software.

Electrontools. (s.f.). *COMPUERTAS LÓGICAS BÁSICAS Y SUS TABLAS DE VERDAD: ETOOLS*. Obtenido de http://www.electrontools.com/Home/WP/2016/05/27/compuertas-logicas-basicas-y-sus-tablas-de-verdad/

Escuela Universitaria de Ingenieria . (2017). *Acomee*. Obtenido de https://www.acomee.com.mx/CONVERTIDORES.pdf

FLOYD, T. (2000). *Fundamentos Digitales.* Ed. Prentice Hall .

Forum, W. I. (2017). *Welcome to the Wireless Innovation Forum*. Obtenido de http://www.wirelessinnovation.org/

Garcia, C. M. (2011). *Diagrama en bloques de un Sistema Digital deComunicaciones.* (U. C. Villas, Ed.) Santa Clara.

GG, L. (22 de Noviembre de 2015). *El Cajon del Electronico*. Obtenido de http://elcajondelelectronico.com/tag/frecuencia-intermedia/

GNU Radio Manual and C++ API Reference: gnuradio.org. (18 de Agosto de 2016). Obtenido de gnuradio.org: https://gnuradio.org/doc/doxygen/group__block.html

gnuradio.org. (Julio de 2017). Obtenido de https://wiki.gnuradio.org/index.php/GNURadioCompanion

gnuradio.org. (s.f). Obtenido de https://gnuradio.org/doc/doxygen/top__block_8h.html

gnuradio.org. (s.f.). Obtenido de https://gnuradio.org/doc/doxygen/block_8h.html

Gomez, M. C. (16 de Agosto de 2012). *Blogspot.* Obtenido de MODULACION Y DEMODULACION: http://carlosgomez098.blogspot.com.co/2012/08/46-modulacion-y-demodulacion.html

Hengles Almeida, J. J., Batista Lopes, P., & Akamine, C. (Junio de 2017). A Proposal for the Next Generation of ISDB-TB using FBMC in a SDR Implementation on GNU Radio Environment. *IEEE Latin America Transactions*. doi:1548-0992

Hernández P, R., & Hernández H, G. (s.f.). *Universidad Autónoma del Estado de Hidalgo*. Obtenido de https://www.uaeh.edu.mx/scige/boletin/huejutla/n9/r1.html

Hernandez, I. (2004). *Procesamiento Digital de Señales.*

Iiehara, K., Shiba, H., Shono, T., Sairato, Y., Yoshioka, H., Nakatsugawa, M., . . . Umeaira, M. (Diciembre de 2002). DESIGN AND PERFOWCE EVALUATION OF SOFTWARE DEFINED RADIO PROTOTYPE FOR PHS AND lEEE802.11 WIRELESS LAN. *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*.

Kadman, W. (2013, MARZO). *R820T RAFAEL MICRO- HIGH PERFORMANCE LOW POWER ADVANCED DIGITAL TV SILICON TUNER*. Retrieved from http://radioaficion.com/cms/r820t-rafael-micro/

Kanga. (2011). *Sitio Web Oficial de Kanga Products,*. Obtenido de http://www.kangaproducts.co.uk/index.php/products/finningley-80m-sdr.

LLC., S. R. (s.f.). *SRL QuickSilver QS1R Receiver,*. Obtenido de http://www.srl-llc.com.

Mario. (05 de Abril de 2012). *Neoteo Electronica*. Obtenido de http://www.neoteo.com/sdr-radio-definida-por-software/

MATLAB, T. D. (2011). *Universidad Central "Marta Abreu" de Las Villas Facultad de Ingeniería Eléctrica.* Obtenido de file:///Users/iPhoenix/Downloads/Carlos%20Manuel%20García%20Algora.pdf

Merino, M. (1989). *Cicuitos Electricos: Digitales I.* Madrid: Servicios Publicaciones de la E.T.S.I Telecomunicaciones . Obtenido de http://www.el.uma.es/marin/Tema7_practica4.pdf

Ministerio de Tecnologías de la Información y las Comunicaciones. (29 de Julio de 2009). *mintic.gov*. Obtenido de http://www.mintic.gov.co/portal/604/w3-article-3707.html

MINTIC. (2016). *Ministerio de Tecnologías de la Información y las Comunicaciones*. Obtenido de http://www.mintic.gov.co/portal/604/w3-article-2350.html

Muslimin, J., Asnawi, A. L., Ismail, A. F., & Jusoh, A. Z. (Julio de 2016). SDR-Based Transceiver of Digital Communication System Using USRP and GNU Radio. *Computer and Communication Engineering (ICCCE), 2016 International Conference on*. doi:978-1-5090-2427-8

Nave, M. O. (Abril de 2014). *Transformada Rápida de Fourier*. Obtenido de Transformada Rápida de Fourier: http://hyperphysics.phy-astr.gsu.edu/hbasees/Math/fft.html

NooElec Inc. (s.f.). *NooElec NESDR XTR Tiny SDR & DVB-T USB Stick (RTL2832U + E4000) w/ Antenna and Remote Control*. Obtenido de http://www.nooelec.com/store/sdr/sdr-receivers/nesdr-xtr-rtl2832u-e4000.html

Osman, M. (Mayo de 2015). Video: Software Defined Radio with HackRF, Lesson 1: Welcome. Obtenido de https://greatscottgadgets.com

Pernter., M. (s.f.). *Sitio Web Oficial de PM-SDR,*. Obtenido de http://www.iw3aut.altervista.org/.

Phil Lapsley, J. B. (1996). *"DSP Processor Fundamentals: Architectures.* Berkeley, California: Design Technology, Inc. Retrieved 10 25, 2017

Pinar, I., & Murillo, J. (2011). *Laboratorio de Comunicaciones Digitales Radio Definida por Software.* Sevilla: Universidad de Sevilla.

Prieto, R., & Rafael, R. (2017). IMPLEMENTACIÓN DE UN SISTEMA DE COMUNICACIÓN. Bogotá, Colombia.

Primavera, G., Coladonato, P. A., Fortuna, C. N., Castia, M. S., Alfonso Melian, N., & Villafañe, D. (s.f.). *econ*. Obtenido de TECNOLOGÍA DE LA INFORMACIÓN: http://www.econ.uba.ar/www/departamentos/sistemas/plan97/tecn_informac/briano/seoane/tp/2002_1/vinculoscomunicacion.htm

RODRÍGUEZ, P. (06 de Agosto de 2013). *XATAKAMOVIL*. Obtenido de https://www.xatakamovil.com/conectividad/hackrf-el-santo-grial-de-los-transmisores-de-radiofrecuencia

Rowetel. (s.f.). *Codec2: Rowetel*. Obtenido de Rowetel web site : http://www.rowetel.com/?page_id=452

Rubestein, R. (marzo de 2010-2011). *Technology Trends*. Obtenido de http://data.memberclicks.com/site/sdf/tut-SDR_article.pdf,

Salazar, J. (s.f.). Procesadores digitales de señal. *Universidad Politecnica de Cataluña*, 1-8.

Sethares, W. A., & Johnson Jr, C. R. (2003.). *Telecommunication Breakdown. Concepts of Communication Transmitted via Software-Defined Radio.,.* Upper Saddle River: NJ: Prentice Hall.

Sierra, E. G., & Ramírez, G. A. (Noviembre de 2015). Low Cost SDR Spectrum Analyzer and Analog Radio Receiver Using GNU Radio, Raspberry Pi2 and SDR-RTL Dongle. *Communications (LATINCOM), 2015 7th IEEE Latin-American Conference on*.

Stewart, R. W., Crocket, L., Atkinson, D., Barlee, K., Crawford, D., Chalmers, L., . . . Sozer, E. (Septiembre de 2015). A low-cost desktop software defined radio design environment using MATLAB, simulink, and the RTL-SDR. *IEEE Communications Magazine*.

Suzuki, H., Kawakita, Y., & Ichikawa, H. (Agosto de 2016). Remote implementation of GNU radio-based SDR development environment. *Communications (APCC), 2016 22nd Asia-Pacific Conference on*. doi:978-1-5090-0676-2

turmero, P. (2015). *Monografias*. Obtenido de http://www.monografias.com/trabajos105/convertidores-dac-y-adc/convertidores-dac-y-adc.shtml

UIT, U. I. (2005). *Union Internacional de Telecomunicaciones (UIT)*. Obtenido de http://www.itu.int/es/Pages/default.aspx

Union Internacional de Telecomunicaciones. (14 de Marzo de 2011). *Recomendacion G.711: ITU*. Obtenido de http://www.itu.int/rec/T-REC-G.711/es

Vaquerizo, J. Á., & Moreno, R. (2016). ASTROCUENCIA. *EXPERIMENTANDO CON LAS ONDAS DE RADIO*.

Vega L, C., Arvizu G, D., & García S, A. (2008). ALGORITMOS PARA. Boca del rio, Mexico.

Villegas, F. F. (2008, Octubre). *EA1URO*. Retrieved from http://www.ea1uro.com/sdr1/sdr.htm

Wikipedia. (Junio de 2017). *Aliasing: Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Aliasing

Wikipedia. (2017). *Coma flotante: Wikipedia*. Obtenido de https://es.m.wikipedia.org/wiki/Coma_flotante

Wikipedia. (13 de OCtubre de 2017). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Frecuencia_intermedia

WiNRADiO. (2011). *WiNRADiO... the future of radio. WR-G31DDC 'EXCALIBUR'*. Obtenido de http://www.winradio.com/home/g31ddc.htm.

11. ANEXOS

11.1. ANEXO 1: DOCUMENTACIÓN DE SOFTWARE GNU RADIO COMPANION.

Según gnuradio.org, (2017) GNU Radio Companion (GRC) es una herramienta y un entorno de desarrollo que facilita el desarrollo de aplicaciones para GNU Radio mediante una interfaz gráfica. Esta herramienta facilita el diseño, ya que está basada en bloques y consiste en la creación de ficheros Python, cabe destacar que para el desarrollo de aplicaciones no es necesario saber programar en este lenguaje (Python) o C++, esto es debido a que GRC genera el código necesario para la aplicación que es construida gráficamente, es decir que GRC es básicamente un lenguaje de programación en entorno visual con el uso de código libre y es utilizado para el procesamiento de señales haciendo uso de las librerías o biblioteca de GNU Radio, varias de las grandes ventajas es en cuanto a que este entorno facilita la corrección de errores, la gestión de pruebas y en cuanto a la ejecución de cada una de las aplicaciones construidas con GRC. (Pinar & Murillo, 2011) (gnuradio.org, 2017)

En primera instancia hay que señalar que para la realización de proyectos y creación de módulos utilizando el GRC hay que entender que esta herramienta está basada en interconectar los diferentes bloques usados, de una manera esquematizada y para esto según Pinar & Murillo (2011) los bloques se pueden agrupar de la siguiente forma:

- ***Sources*** (fuentes): Estos bloques detallan cualquier tipo de fuente, tales como: archivos de audio, generadores de señal, fuentes binarias aleatorias, micrófonos, etc.

- ***Bloques de procesado de señal***: En este grupo se encuentran todos aquellos bloques que de alguna manera realizan algún tratamiento a la señal o señales con que se trabajada. Algunos ejemplos podrían ser: amplificadores, filtros, remuestreadores, moduladores, multiplicadores, etc.

- ***Sinks*** (sumideros): Los bloques de este grupo son los que hacen parte del final del diagrama que se realiza en GRC, y son los que aseguran que la señal como destino final llegara a un fichero de cualquier formato o a una tarjeta de sonido. En este conjunto también encontramos los diferentes bloques que permiten visualizar las señales (*Graphical sinks*) como *FFT sink* (para visualizar la FFT de la señal en un punto), *Constelation sink* u *Osciloscope sink* (para representar la señal en tiempo en cualquier lugar del diseño) entre otros.

Básicamente todas las clases que se encuentran en GRC hacen parte de alguno de estos grandes grupos, sin embargo cabe señalar que en GRC se usan una serie de bloques de "variables" los cuales no están incluidos en alguno de estos tres grupos. Estos bloques lo que hacen es definir las diferentes variables pertinentes que se van a utilizar dentro del esquema, estos bloques no se interconectan con los bloques que efectúan todo el proceso del proyecto que se quiere realizar, es decir que no se conectan con ningún bloque pertenecientes a los tres grupos de bloques mencionados. Esto se verá con más detalle y de forma mucho más practica al realizar los laboratorios propuestos más adelante en este trabajo.

Como ya se ha dicho GNU está basado en bloques de procesamiento de señal de radio los cuales están implementados bajo lenguaje de programación C++ y que luego estos son portados a Python para poder crear los diferentes grafos en la interfaz visual de GRC que permiten la realización de diferentes proyectos de una manera mucho más fácil que si se realizara con lenguaje puro de programación.

Por lo anterior ahora se presenta una explicación detallada basada en DocBook (2013) de algunos de los módulos y sus respectivos bloques, enfocada 100% a GRC (forma gráfica). Esta explicación se centra en tomar algunos de los bloques más usados e importantes en el software

GRC, para así poder entender de forma explícita que es lo que hace y para qué sirve cada uno de ellos. Teniendo claro que en C++ se implementan dichos módulos y sus respectivas clases, y que luego estas son traídas a Python para su desarrollo de forma gráfica en GRC, se hace imprescindible esta explicación que ayudara y permitirá un entendimiento más claro y práctico de lo que se puede hacer en GRC con sus diferentes bloques y así facilitar la realización de los diferentes proyectos propuestos para este software; estos módulos se encuentran en forma de lista en la parte derecha de la pantalla de la hoja de trabajo de GRC, sin embargo cabe resaltar que en este espacio solo se explicaran algunos bloques de dicha lista mas no se entrara detallar el entorno de GRC, no obstante más adelante en el desarrollo de los laboratorios se mostrara en forma de guía los diferentes componentes del entorno GRC.

En primera instancia se presenta el primer bloque con el que se empieza a trabajar en GRC, luego de este si se entrara a explicar los diferentes módulos y los bloques que lo componen:

11.1.1. Options (Bloque de Opciones del Workspace)

Este es bloque el primer bloque con el que se trabaja en GRC y se denomina de opciones, en este se debe especificar el ID, Titulo, el tamaño de la ventana, y quizás lo más importante el tipo de flujograma (bloques de interfaz gráfica) que se empezara a diseñar (Prieto & Rafael, 2017). En la Figura 122 se muestra las propiedades de este bloque.

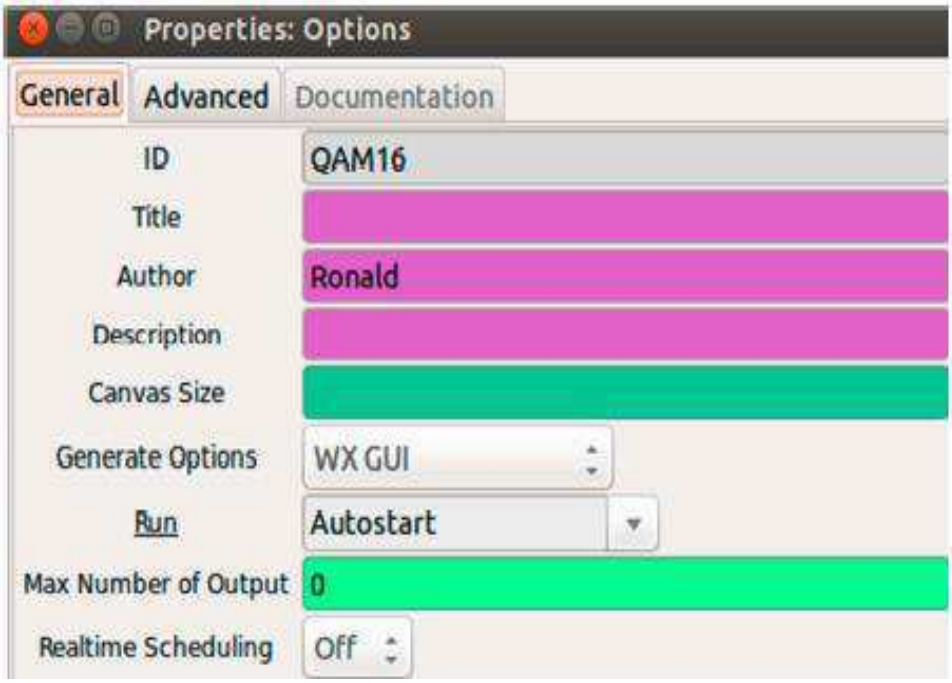

Figura 122. Propiedades bloque Options. (Prieto & Rafael, 2017)

El parámetro *Generate Options* es lo más importante en este bloque ya que en este es en donde establecemos el tipo de flujograma, las opciones que este bloque nos muestra con su explicación son las siguientes:

QT GUI y WX GUI: hacen referencia al tipo de librerías que se utilizaran para la interfaz gráfica, QT consume menos recursos de maquina

NO GUI: es para hacer un flujograma sin interfaz gráfica y así no consumir tantos recursos de máquina

HIER BLOCK: es para crear un bloque jerárquico que será almacenado dentro del programa y podrá ser usado en otros programas. (Prieto & Rafael, 2017)

11.1.2. Level Controllers (Controladores de nivel)

11.1.2.1. AGC

El bloque AGC es un controlador de nivel o ganancia que básicamente lo que hace es que implementa un nivel automático de ganancia utilizando un solo parámetro de velocidad

(Rate) tanto para aumentar como para deteriorar. Esta ganancia es actualizada por la ecuación “Gain = Gain + Rate* (Reference – abs (input)) para cada una de las muestras. En la Figura 123 se muestra el bloque AGC en su forma gráfica y las propiedades de este bloque respectivamente.

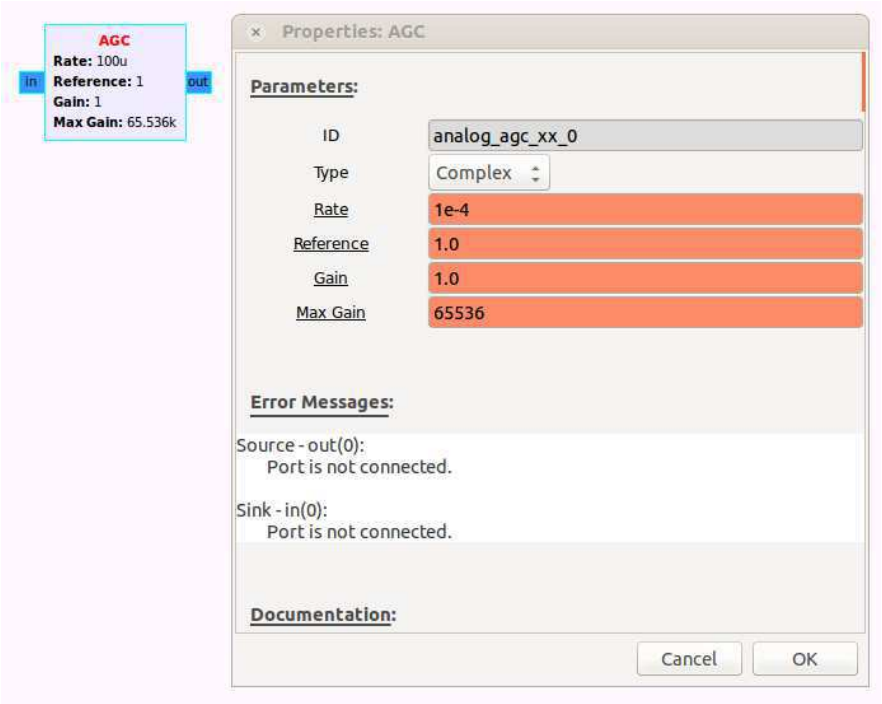

Figura 123. Bloque AGC y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cinco parámetros más en sus propiedades, los cuales son:

Type: especifica el tipo de dato de la entrada y la salida, estos pueden ser *Complex* y *Float.* El primero especifica que la entrada y la salida son valores complejos, es decir un valor real sumado con un valor imaginario y el segundo que la entrada y la salida son valores de punto o coma flotante (reales), es decir que almacena una aproximación muy precisa del valor.

Rate: este parámetro es de tipo real y en él lo que se hace es establecer la velocidad a la cual el AGC podrá ajustar la ganancia. Este ajuste de ganancia lo realiza según la ecuación dada anteriormente.

Reference: este parámetro de tipo real y se encarga de ajustar la amplitud de referencia. El bloque AGC intentara mantener la salida al nivel acotado mediante el ajuste de la ganancia.

Gain: al igual que los dos parámetros anteriores este también es de tipo real y se encarga de asignar la ganancia inicial y que para la mayoría de aplicaciones se establece en 1.0

Max Gain: este parámetro es de tipo real y se encarga de establecer el valor máximo que podría llegar a tener el parámetro *Gain* lo que significa que el bloque AGC no podrá superar este valor con el ajuste de ganancia, cabe resaltar que si el valor de este parámetro se pone en 0 querrá decir que no existirá un valor máximo de ganancia.

11.1.2.2. Feed Forward AGC

Este bloque *Feed Forward AGC* lo que hace es hallar el valor máximo dentro del buffer de acuerdo al número especificado de muestras y luego lo normaliza para que su valor máximo corresponda al valor de referencia. Este AGC produce un retardo en la señal ya que el cálculo que se maneja dentro del bloque es aproximado lo que quiere decir que el nivel de salida no será igual al nivel de referencia, sin embargo esto no suele ser un problema en la realización de proyectos. En la Figura 124 se observa el bloque *Feed Forward AGC* y sus propiedades.

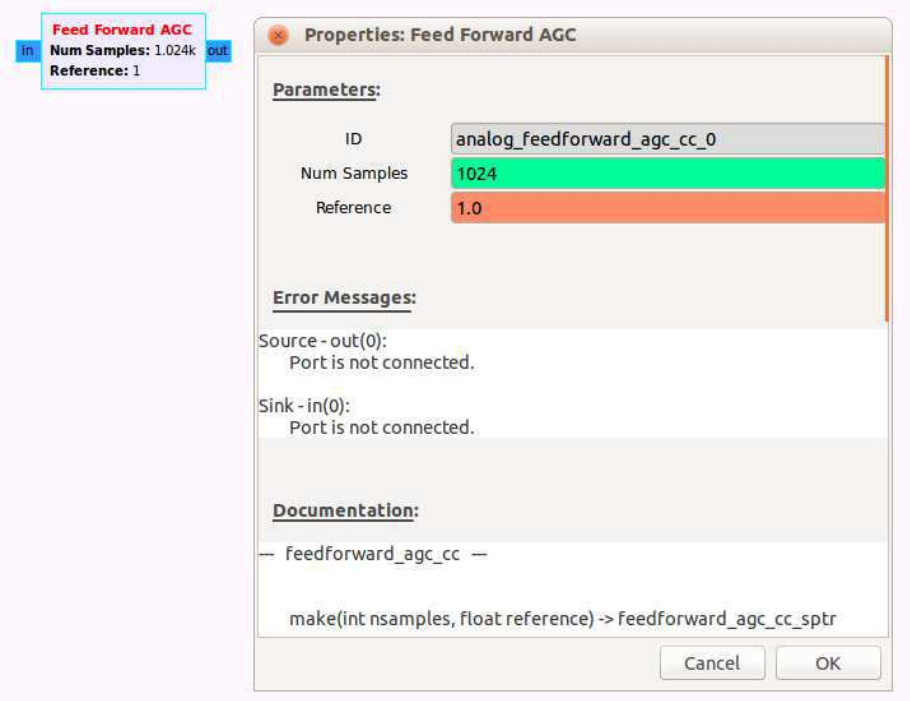

Figura 124. Bloque Feed Forward AGC y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

Num Samples: este parámetro es de tipo int (entero) y hace referencia al número de muestras en cada buffer.

Reference: este parámetro es de tipo real y es el valor que se encarga de normalizar lo buffer.

11.1.2.3. Power Squelch (Silenciador de energía)

Este bloque implementa un control de silenciamiento el cual está basado en el nivel de la potencia de la señal entrante. En la Figura 125 se observa el bloque gráficamente y sus propiedades.

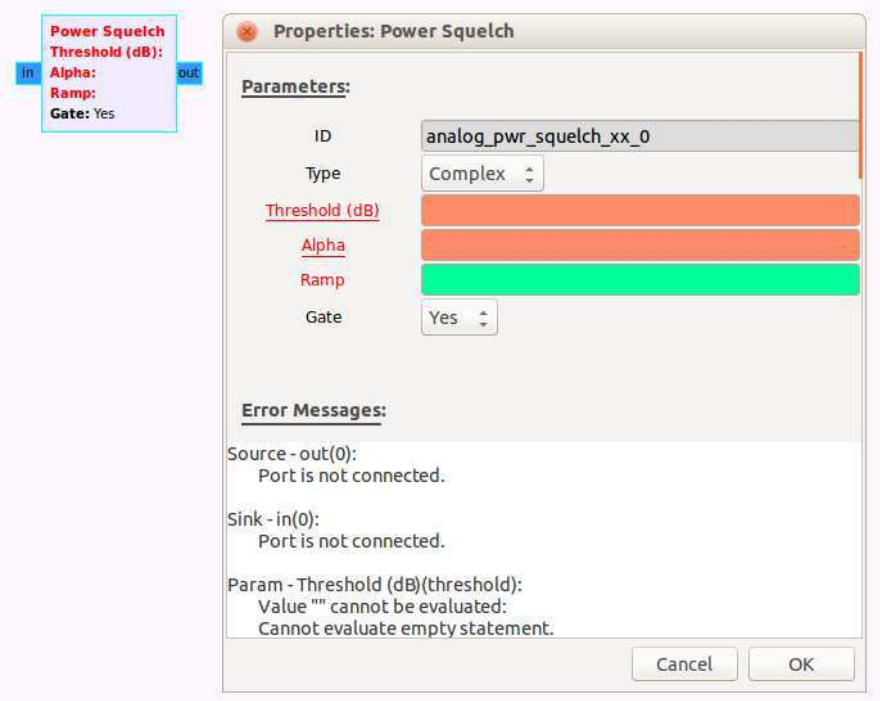

Figura 125. Bloque Power Squelch y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cinco parámetros más en sus propiedades, los cuales son:

Type: establece el tipo de dato de entrada y salida los cuales pueden ser de forma complex o float. Para complex quiere decir que los flujos de datos de la entrada y la salida son complejos y para float estos flujos en su entrada y su salida son reales. En la Figura 126 se pueden observar estos dos tipos de datos.

Complejo	Las corrientes de entrada y salida son complejas.
Flotador	Las corrientes de entrada y salida son reales.

Figura 126. Tipos de datos para Type del bloque Power Squelch. (DocBook, 2013)

Threshold (dB): el tipo de dato de este parámetro es real y establece un umbral en dB donde el control del silenciador cambia de estados entre MUTED y UNMUTED, esto lo hace

si el parámetro *Ramp* está habilitado, donde estaría en un estado de ATTACK cuando va de MUTED a UNMUTED y un estado DECAY cuando va de UNMUTED a MUTED.

Alpha: maneja datos de tipo real y es quien controla el filtrado del nivel de potencia de la señal mediante la ecuación "y[i] = (1-Alpha) * y[i-1] + Alpha * x[i]."

Ramp: este parámetro maneja datos de tipo int y establece el número de muestras que ocurrirán en las transiciones de MUTED a UNMUTED y de UNMUTED a MUTED.

Gate: este parámetro específica si el silenciador está interrumpiendo la señal cuando está en MUTED, para esto se establecen dos opciones YES o NO. Cuando está en YES el silenciador se activa cuando MUTED y se establece la salida en 0 y cuando está en NO el silenciador no se está activando.

11.1.2.4. Rail

Este bloque establece un límite fuerte entre los valores más altos y más bajos en su salida, el efecto que produce es similar a un amplificador operacional que está siendo impulsado a la saturación. En la Figura 127 se observa el bloque Rail gráficamente y sus propiedades.

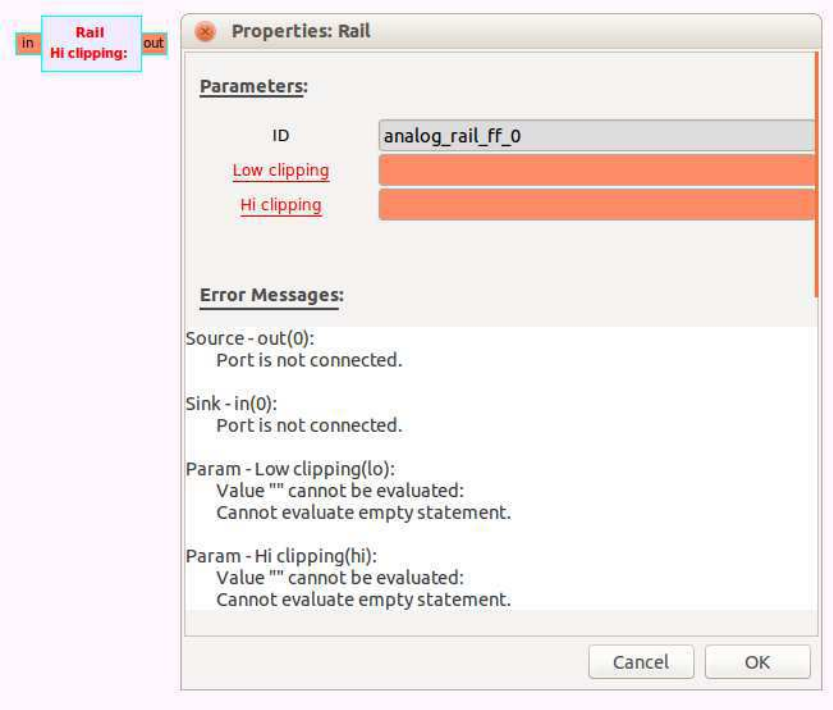

Figura 127. Bloque Rail y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

Low clipping: maneja datos de tipo real y establece el valor mínimo de la potencia que será emitido desde el bloque.

Hi clipping: también maneja datos de tipo real y lo que este parámetro hace es establecer el valor máximo de la potencia que será emitido desde el bloque.

11.1.2.5. Mute

Este bloque se utiliza para silenciar una señal, este establece la salida en 0 cuando *Mute* se instaura en *True.* En la Figura 128 se observa el bloque *Mute* gráficamente junto con sus propiedades.

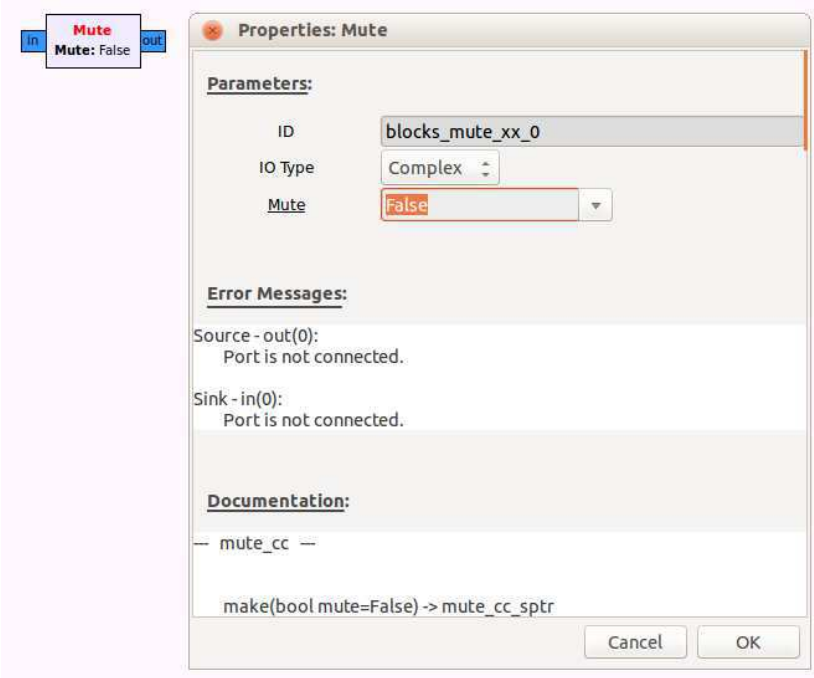

Figura 128. Bloque Mute y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

IO Type: este parámetro establece el tipo de dato que se manejara en los flujos de entrada y salida. Maneja cuatro tipos de datos: Complex, Float, Int y Short, tanto Complex como Float ya se han explicado (ver Figura 126), para el caso del dato tipo Int significa que se establece los flujos de entrada y de salida en un dato entero de 32bits y el tipo Short quiere decir que estos flujos se establecen en un dato entero pero de 16bits. En la Figura 129 se puede observar este tipo de datos.

Complejo	Establece las secuencias de entrada y salida en complejas.
Flotador	Establece las corrientes de entrada y salida en reales.
Int	Establece las secuencias de entrada y salida en un entero de 32 bits.
Corto	Establece las secuencias de entrada y salida en un entero de 16 bits.

Figura 129. Tipos de datos para IO Type del bloque Mute. (DocBook, 2013)

Mute: este parámetro es de tipo raw y en este especifica si se quiere silenciar la entrada, contiene dos opciones *True* y *False,* estas se pueden seleccionar en el menú desplegable y también se puede escribir el nombre de una variable en este mismo cuadro.

11.1.2.6. Sample and Hold (Muestreo y retención)

Este bloque se encarga de implementar una operación de muestreo y retención, en este bloque encontramos dos entradas *Input, Ctrl* y una salida *Output.* Cuando el valor de la entrada *Ctrl* es diferente de cero la señal que se encuentra en la entrada *Input* pasa a *Output*, pero cuando *Ctrl* es igual a cero la señal que está en *Output* se mantiene hasta que *Ctrl* sea nuevamente diferente a cero. En la Figura 130 se observa gráficamente el bloque *Sample and Hold* y sus propiedades.

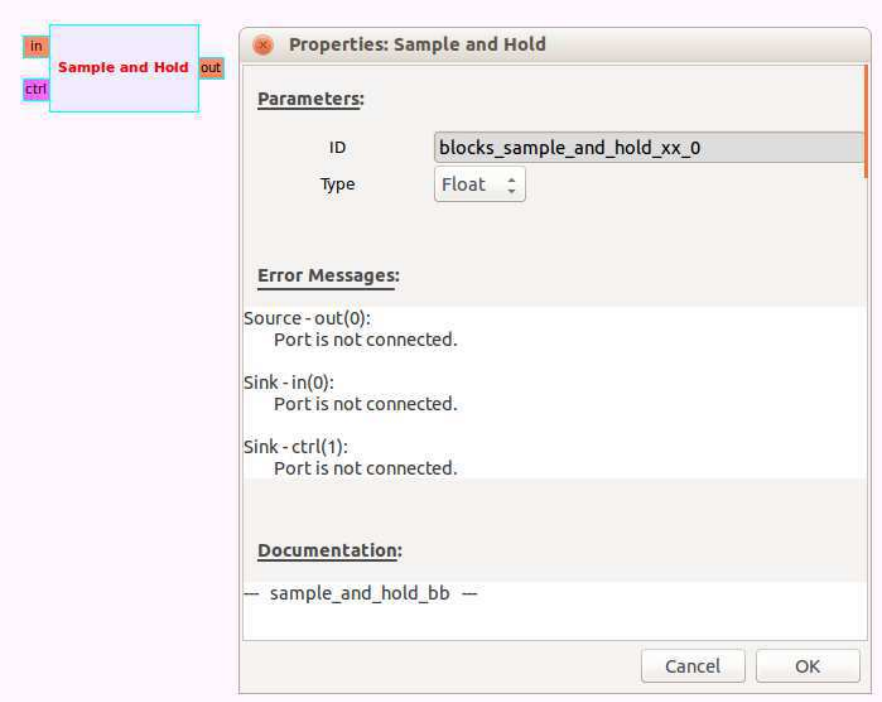

Figura 130. Bloque Sample and Hold y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* un parámetro más en sus propiedades, el cual es:

Type: este parámetro establece el tipo de datos en los flujos de entrada y salida. Estos datos son Float, Int, Short y Byte, los tres primeros ya se explicaron anteriormente (ver Figura 129) y el último *Byte* establece el flujo de entrada y de salida en bytes de 8 bits.

11.1.2.7. Moving Average (Media móvil)

Este bloque establece un filtro básico de promedio móvil, lo que ayuda a suavizar señales ruidosas de forma muy sencilla, para obtener un promedio real es necesario establecer el parámetro *Scale* en 1. En la Figura 131 se observa el bloque *Moving Average* de manera gráfica junto con sus propiedades.

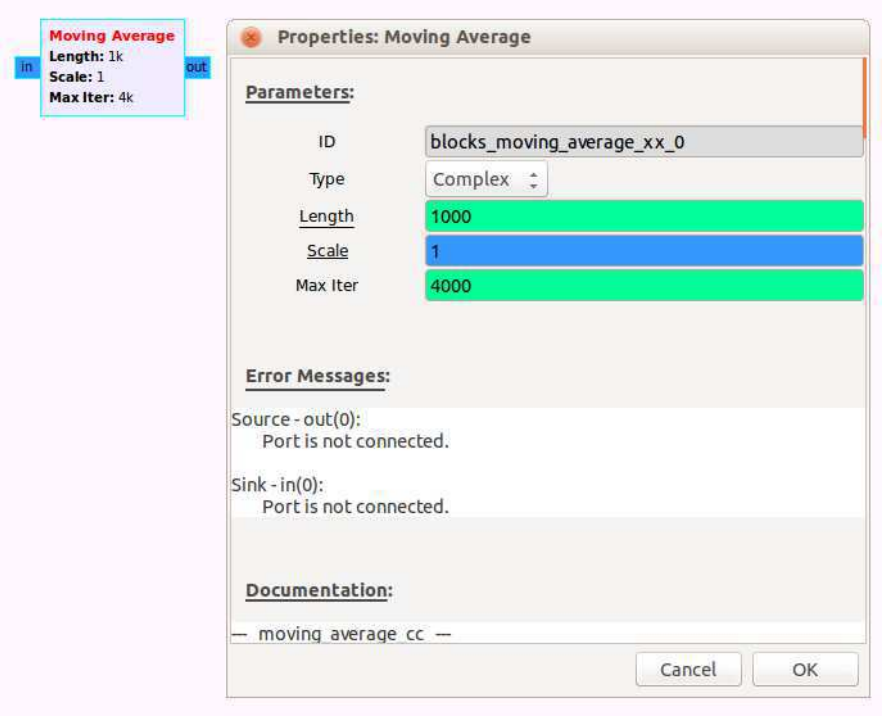

Figura 131. Bloque Moving Average y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cuatro parámetros más en sus propiedades, los cuales son:

Type: establece el tipo de datos del flujo de entrada y salida, este bloque maneja los mismos datos de la Figura 2 (ver Figura 129).

Length: este parametro maneja datos tipo Int y es el que establece la longitud que tendrá el filtro de promedio.

Scale: este maneja los mismos tipos de datos que el parámetro *Type* (ver Figura 129).

Max Iter: este es de tipo Int y generalmente de deja el valor prederminado a no ser que deba cambiarlo obligatoriamente. Este parametro lo que hace es limitar el tiempo que pasa sin que se limpie el acumulador, esto es realmente necesario ya que ayuda a controlar la inestabilidad numérica de los datos *Complex* y *Float.*

11.1.2.8. Threshold (Límite)

Este bloque implementa un comparador configurable donde la salida pasa de 0.0 a 1.0 cuando la señal de entrada transita da abaja hacia arriba del nivel alto, y la salida pasa de 1.0 a 0.0 cuando la señal de entrada pasa de arriba abajo del nivel bajo. Siempre el nivel bajo debe ser menor que el nivel alto. En la Figura 132 se muestra el bloque *Threshold* y sus propiedades.

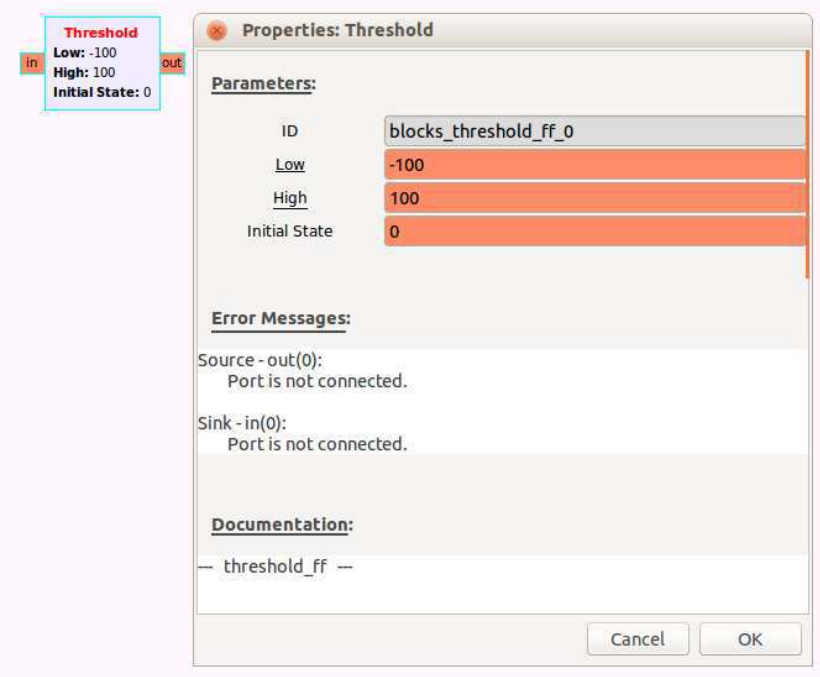

Figura 132. Bloque Threshold y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

Low: este parámetro maneja datos de tipo real y establece el nivel para la transición de alto a bajo.

High: tipo de datos real y establece el nivel para la transición de bajo a alto.

Initial State: es de tipo real y establece si el estado inicial de la salida estará en 0.0 o 1.0.

11.1.3. Waveform Generators (Generadores de forma de onda)

11.1.3.1. Signal Source (Fuente de señal)

Este bloque se utiliza para generar diferentes tipos de señales, tales como: seno (sine), coseno (cosine), cuadrado (square), diente de cierra (sawtooth) y triangulo (triangle). En la Figura 133 se observa el bloque *Signal Source* de forma gráfica junto con sus propiedades.

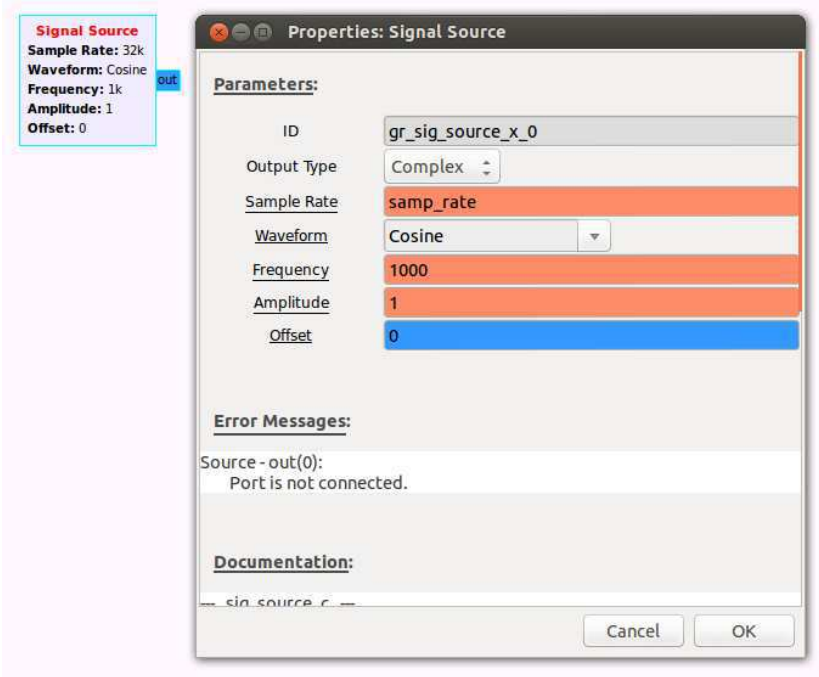

Figura 133. Bloque Signal Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* seis parámetros más en sus propiedades, los cuales son:

Output Type: este parámetro especifica el tipo de datos en la salida. Esta puede tomar un tipo Complex que hace referencia a un valor complejo, tipo Float donde toma un valor real, tipo Int de un entero de 32 bits y tipo Short donde es un entero de 16 bits. En la Figura 134 se muestran estos tipos de datos.

Complejo	La salida es de valor complejo.
Flotador	La producción es de valor real.
Int	La salida es un entero de 32 bits.
Corto	La salida es un entero de 16 bits.

Figura 134. Tipos de datos para Output Type del bloque Signal Source. (DocBook, 2013)

Sample Rate: los datos manejados en este parámetro son de tipo real y este especifica la frecuencia de muestreo de salida.

Waveform: en este parámetro se especifica la forma de onda que se quiere que salga. Los tipos de onda posibles se observan en la figura 135.

Constante	Output es un valor constante igual al parámetro Amplitude más el parámetro Offset. Tenga en cuenta que la amplitud es sólo real, mientras que el desplazamiento puede ser complejo. El bloque de fuente constante proporciona la misma funcionalidad.
Seno	La salida es una onda sinusoidal con amplitud de pico configurada por el parámetro Amplitud y el valor promedio ajustado por el parámetro Offset.
Coseno	La salida es una onda de coseno con amplitud de pico configurada por el parámetro Amplitud y el valor medio ajustado por el parámetro Offset.
Cuadrado	La salida es una onda cuadrada con amplitud pico a pico configurada por el parámetro Amplitud y el valor medio establecido por Offset + Amplitude / 2. Obsérvese que en el caso Complejo, la señal imaginaria es simplemente otra onda cuadrada que ha sido desplazada por noventa grados.
Triángulo	La salida es una onda triangular con amplitud pico a pico configurada por el parámetro Amplitud y el valor medio ajustado por Offset + Amplitud / 2. Obsérvese que en el caso Complejo, la señal imaginaria es simplemente otra onda triangular que ha sido desplazada por noventa grados.
Diente de sierra	La salida es una onda de diente de sierra positiva con amplitud pico a pico configurada por el parámetro Amplitud y el valor medio ajustado por Offset + Amplitud / 2. Obsérvese que en el caso Complejo, la señal imaginaria es simplemente otra onda de diente de sierra que ha sido desplazada por noventa grados.

Figura 135. Tipos de onda para el bloque Signal Source. (DocBook, 2013)

Frequency: maneja datos de tipo real y especifica la frecuencia de salida de la fuente de la señal. En este se puede observar como aparece el efecto *Aliasing* si la *Frequency* se fija mayor que el *Sample Rate.*

Amplitude: este parámetro maneja datos de tipo real y especifica el pico de amplitud de las ondas seno y coseno, al igual que la amplitud pico a pico de las ondas triangulo, cuadrado

y diente de sierra. Cuando se usa la salida Constant, normalmente es establecida en 0 y se utiliza el parámetro Offset.

Offset: en este parámetro se manejan datos Complex y con este se especifica el desplazamiento que se agrega a la forma de onda generada.

11.1.3.2. Constant Source (Fuente Constante)

Este bloque lo que hace es emitir un valor contante a lo largo del tiempo sin variación alguna. En la Figura 136 se muestra este bloque de forma gráfica y sus propiedades.

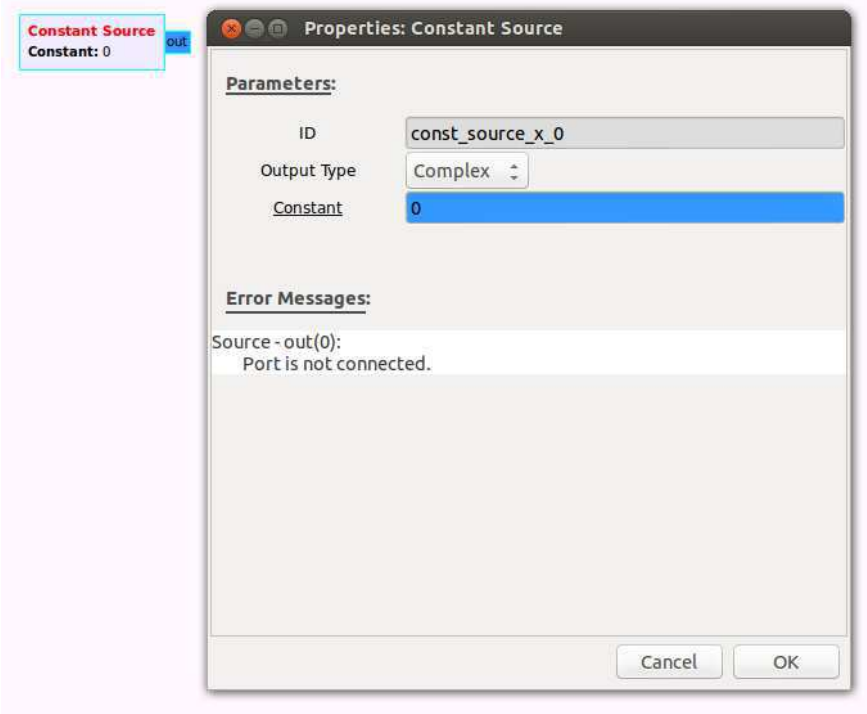

Figura 136. Bloque Constant Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

Output Type: especifica el tipo de dato en su salida (ver Figura 134).

Constant: este valor maneja el mismo tipo de dato de *Output Type* (ver Figura 134) y lo que hace es especificar el valor contante de la salida.

11.1.3.3. Noise Source (Fuente de ruido)

Este bloque se encarga de implementar una fuente de ruido, la distribución de este ruido se puede escoger dentro de las distribuciones estándar que contiene. En la Figura 137 se observa de forma gráfica este bloque y sus propiedades.

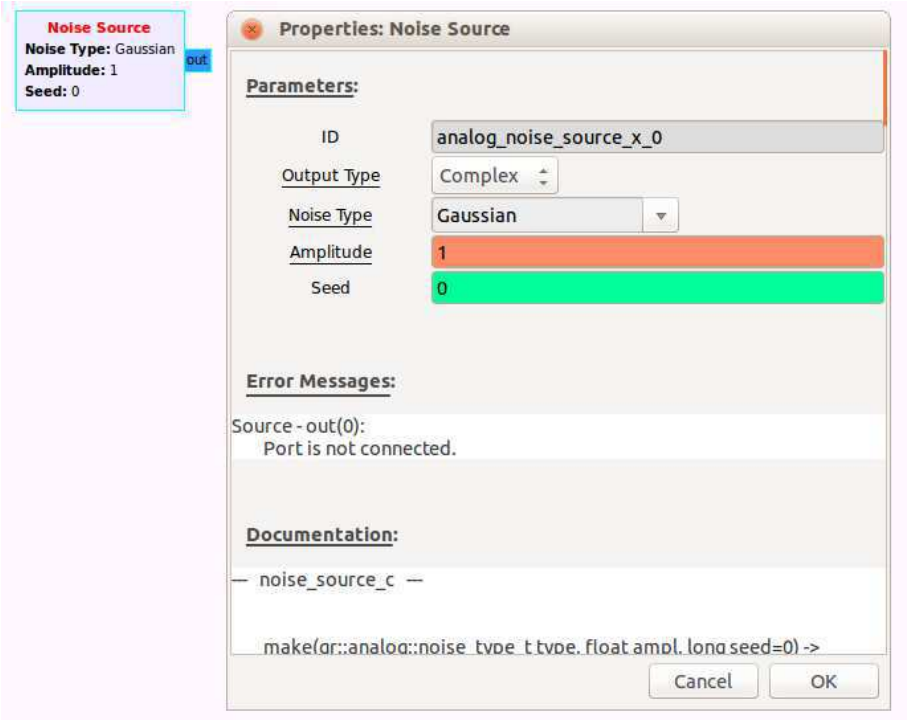

Figura 137. Bloque Noise Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cuatro parámetros más en sus propiedades, los cuales son:

Output Type: este parámetro establece el flujo de datos de la salida. En la Figura 138 se especifican los tipos de datos: *Complex, Float, Int y Short.*

Complejo	La transmisión de salida es compleja.
Flotador	La transmisión de salida es real.
Int	El flujo de salida es un entero de 32 bits.
Corto	La secuencia de salida es un entero de 16 bits.

Figura 138. Tipos de datos para Output Type del bloque Noise Source. (DocBook, 2013)

Noise Type: con este parámetro se selecciona la distribución estadística del ruido. En la Figura 139 se muestran las opciones dadas para esta distribución, en estas se encuentran la distribución Uniforme, distribución Gaussiana, distribución Laplaciana y distribución de Impulso.

Uniform	Selects a uniform distribution.
Gaussian	Selects a Gaussian distribution.
Laplacian	Selects a Laplacian distribution.
Impulse	Selects an Impulse distribution.

Figura 139. Tipos de distribución estadística del ruido. (DocBook, 2013)

Amplitude: en este parámetro se manejan datos de tipo real y con él se establece la amplitud de la salida, esto afectara directamente las características de la estadística de la fuente de ruido, por ejemplo se afectara la desviación estándar de la fuente Gaussiana.

Seed: este parámetro es de tipo Int y con él se establece el *Seed* (en español seria semilla) en el generador de números aleatorios que se utiliza para crear la fuente de ruido, generalmente el valor predetermino es suficiente para la mayoría de propósitos.

11.1.3.4. Random Source (Fuente aleatoria)

Este bloque genera *Num Samples* números aleatorios dentro del rango [min, max). También repite muestras si se específica. Al establecer min = 0 y max = 2, se generara la secuencia 01110101… En la Figura 140 se observa el bloque *Random Source* y sus propiedades.

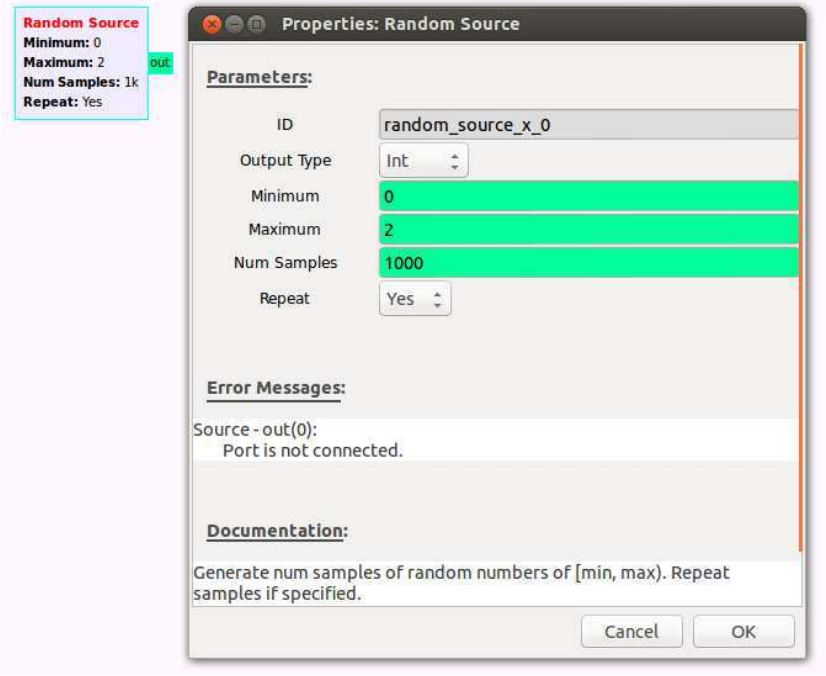

Figura 140. Bloque Random Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cinco parámetros más en sus propiedades, los cuales son:

Output Type: específica el tipo de dato de la salida, en este encontramos el tipo Int, Short y Byte, el primero es un entero de 32 bits, el segundo es un entero de 16 bits y el tercero es un byte de 8 bits.

Minimum: el tipo de dato manejado es Int y establece el valor mínimo que se producirá en la salida.

Maximum: hace referencia al valor máximo que se producirá en la salida, será máximo 1.

Num Samples: es el número de muestras que se emitirán.

Repeat: con este parámetro se selecciona si se quiere repetir o no la secuencia. Se elige YES para repetir la secuencia luego de las muestras de *Num Samples,* y se escoge NO para no repetir luego de las muestras de *Num Samples* y no habrá nada en salida de la *Random Source* y el diagrama de flujo aparecerá suspendido.

11.1.3.5. VCO

Este bloque implementa un oscilador controlado por voltaje. La frecuencia en la salida es linealmente proporcional a la entrada de voltaje multiplicado por la sensibilidad. En la Figura 141 se muestra al bloque *VCO* de forma gráfica junto con sus propiedades.

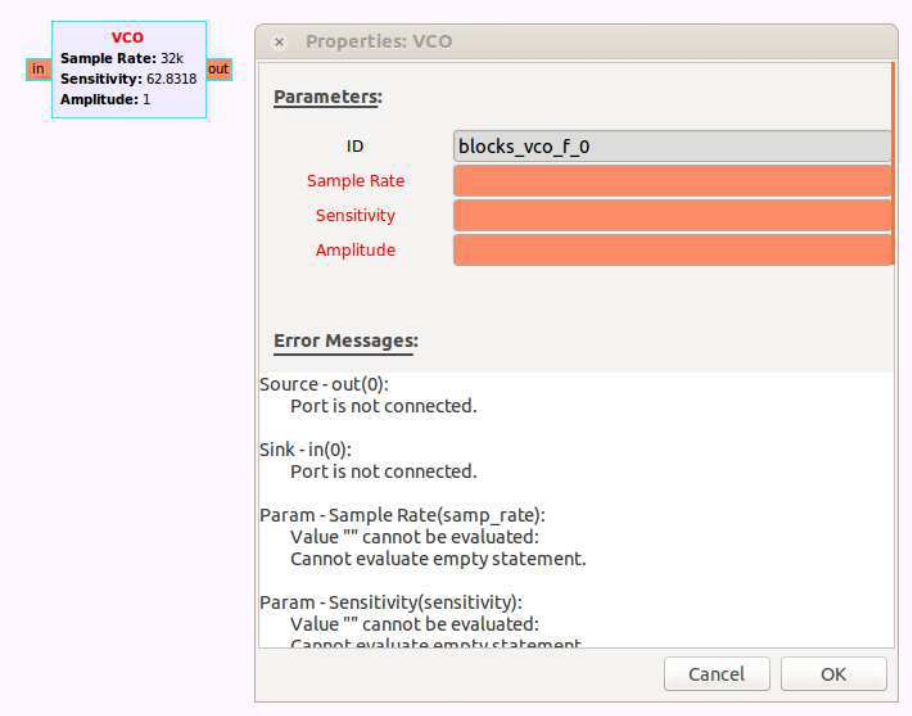

Figura 141. Bloque VCO y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

Sample Rate: datos de tipo real. Establece la velocidad de muestreo para *VCO*. Para la mayoría de proyectos se puede establecer esta velocidad a la variable de tasa de muestreo samp_rate.

Sensitivity: este parámetro maneja datos de tipo real y establece una constante escalar para el *VCO*.

Amplitude: establece la amplitud de salida para el *VCO* y maneja datos de tipo real.

11.1.4. Misc (Miscelánea)

11.1.4.1. Virtual Source (Fuente Virtual)

Este bloque es usado para dar orden a un diagrama de flujo complejo, este se usa emparejándolo con un bloque de *Virtual Sink* y básicamente es como si se dibujara la conexión (cable) entre dos bloques. En la Figura 142 se observa el bloque *Virtual Source* y sus propiedades.

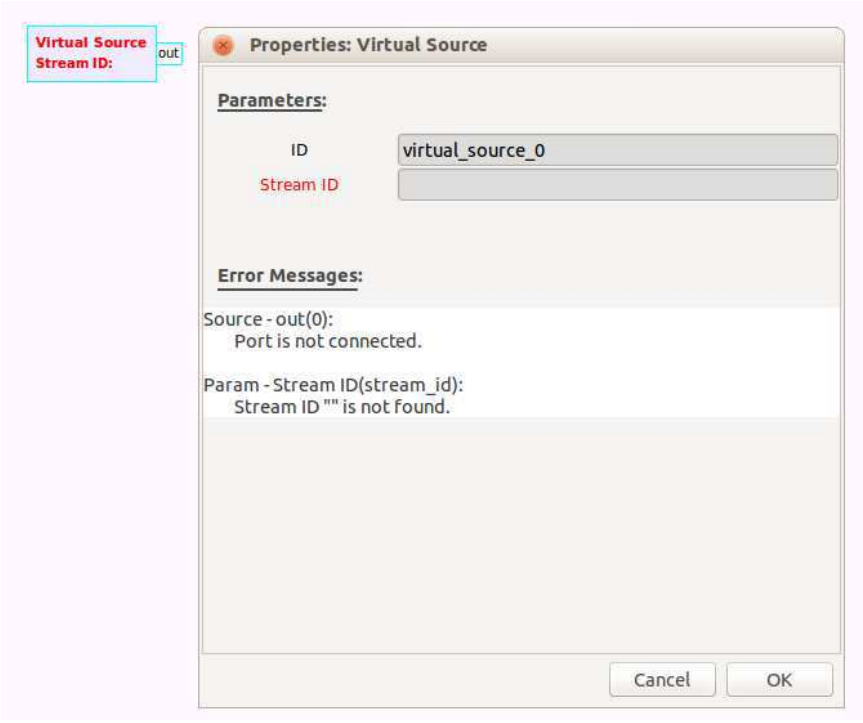

Figura 142. Bloque Virtual Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* un parámetro más en sus propiedades, el cual es:

Stream *ID*: es de tipo *stream_id* y básicamente especifica el *ID* de flujo del *Virtual Sink* para poder leerse y que se puedan emparejar correctamente.

11.1.4.2. Virtual Sink (Receptor Virtual)

Este bloque al igual que el anterior es usado para dar orden a un diagrama de flujo complejo, este se usa emparejándolo con un bloque de *Virtual Source* y básicamente es como si se dibujara la conexión (cable) entre dos bloques. En la Figura 143 se observa el bloque *Virtual Sink* y sus propiedades.

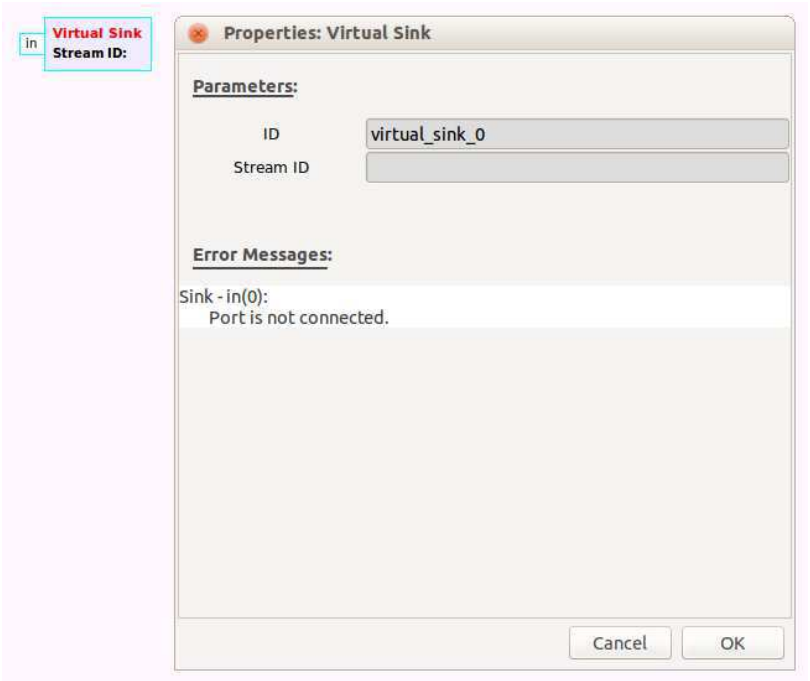

Figura 143. Bloque Virtual Sink y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* un parámetro más en sus propiedades, el cual es:

Stream *ID*: es de tipo *stream_id* y es el identificador único para el flujo, la *Virtual Source* tendrá este mismo identificador para así poder conectarse correctamente.

11.1.4.3. Note (Nota)

Este bloque no afecta de ninguna manera el flujo de señal y se usa básicamente para agregar una nota al diagrama de flujo. En la Figura 144 se muestra el bloque de forma gráfica y sus propiedades.

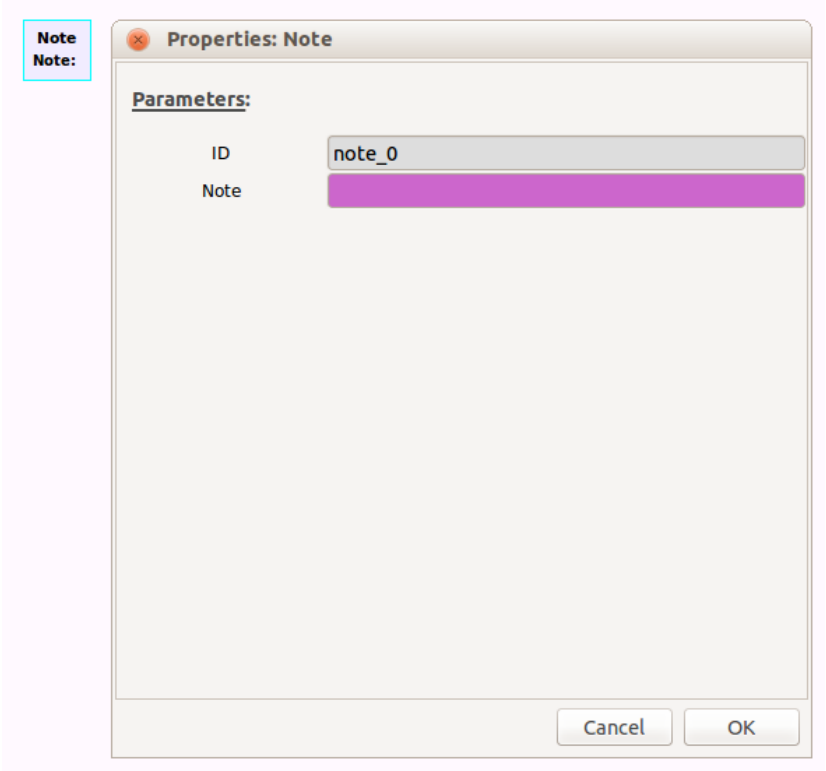

Figura 144. Bloque Note y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* un parámetro más en sus propiedades, el cual es:

Note: es te tipo cadena y precisamente es la cadena que se mostrar en el bloque de notas, es decir, lo que se quiere anotar para tener en cuenta.

11.1.4.4. Import (Importar)

Este bloque importa bloques Python adicionales en el espacio de nombres. En la Figura 145 se observa el bloque *Import* y sus propiedades.

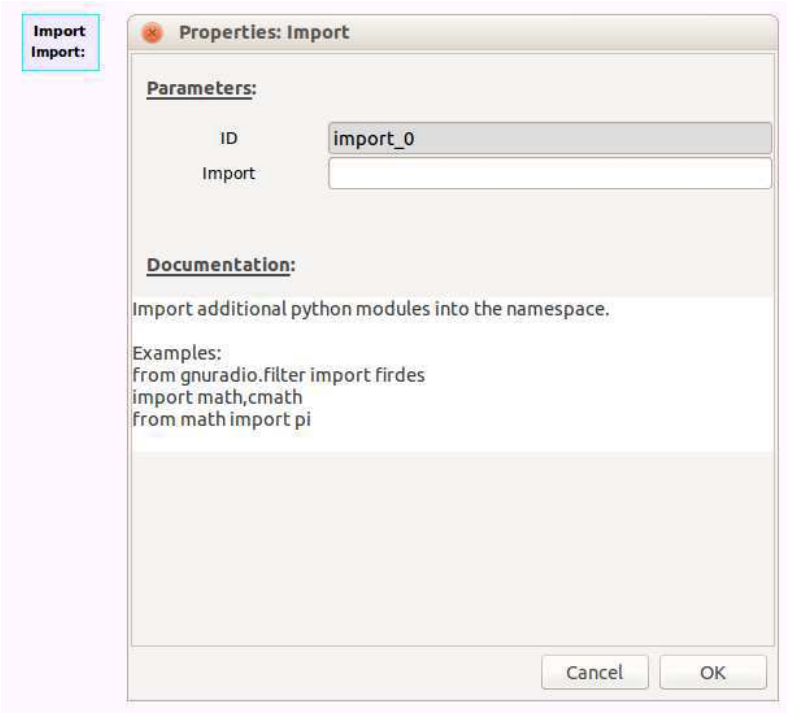

Figura 145. Bloque Import y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* un parámetro más en sus propiedades, el cual es:

Import: este parámetro es de tipo *import* y con él se especifica el boque que se desea importar por ejemplo, `import cmath`.

11.1.4.5. Selector

Este bloque actúa como un multiplexor o de multiplexor para múltiples trayectos de señales, se debe conectar el receptor *(Sink)* en el índice de entrada a la fuente en el índice de salida y se dejan los demás puertos desconectados. En la Figura 146 se observa el bloque mencionado con sus propiedades.

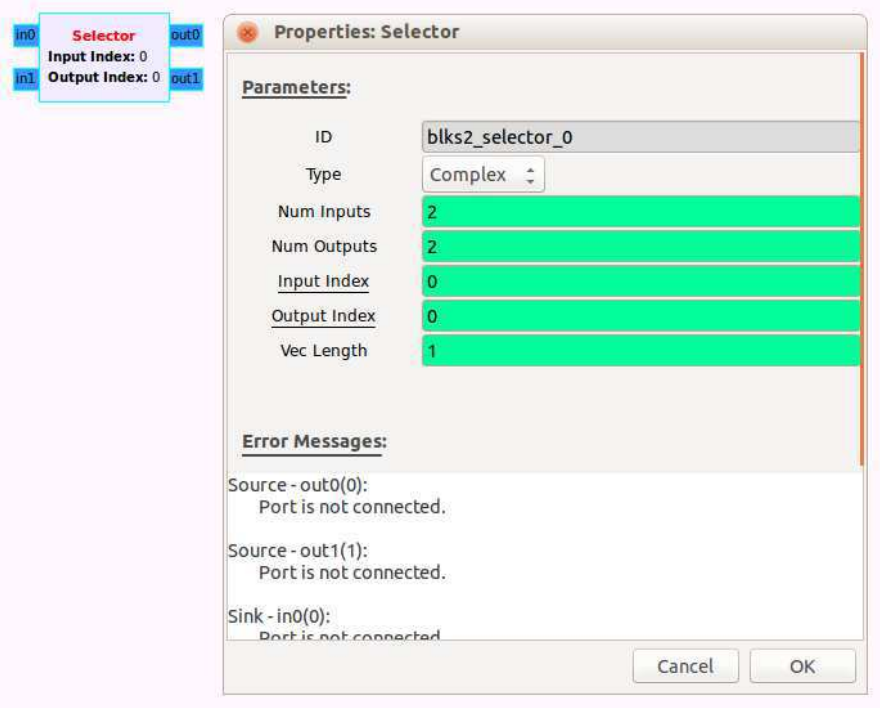

Figura 146. Bloque Selector y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* seis parámetros más en sus propiedades, los cuales son:

Type: especifica el tipo de dato de los flujos de entrada y salida. Maneja los tipos de datos mencionados anteriormente: *Complex, Float, Int, Short* y *Byte.* En la Figura 147 se especifican estos tipos de datos.

Complejo	Las corrientes de entrada y salida son complejas.
Flotador	Las corrientes de entrada y salida son reales.
Int	Las secuencias de entrada y salida son de un entero de 32 bits.
Corto	Las secuencias de entrada y salida son de 16 bits enteros.
Byte	Las corrientes de entrada y salida son bytes de 8 bits.

Figura 147. Tipos de datos para el bloque Selector. (DocBook, 2013)

Num Inputs: es de tipo Int y especifica el número de entradas. Las entradas serán etiquetadas de in0 a in(N – 1).

Num Outputs: es de tipo Int y especifica el número de salidas. Estas salidas serán etiquetadas de out0 a out(N – 1).

Input Index: este parámetro también es de tipo Int y con él se especifica la entrada seleccionada, el valor debe estar entre 0 y (Num Imputs – 1)

Output Index: también es de tipo Int y especifica la entrada seleccionada, el valor debe estar entre 0 y (Num Outputs – 1)

Vec Length: este parámetro es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Usualmente en las aplicaciones se utiliza el valor por defecto 1.

11.1.4.6. Valve (Valvula)

Este bloque como su nombre lo indica es una válvula y se utiliza cuando la válvula está cerrada (no abierta) y se conecta la salida a la entrada. En la Figura 148 se observa el bloque *Valve* y sus propiedades.

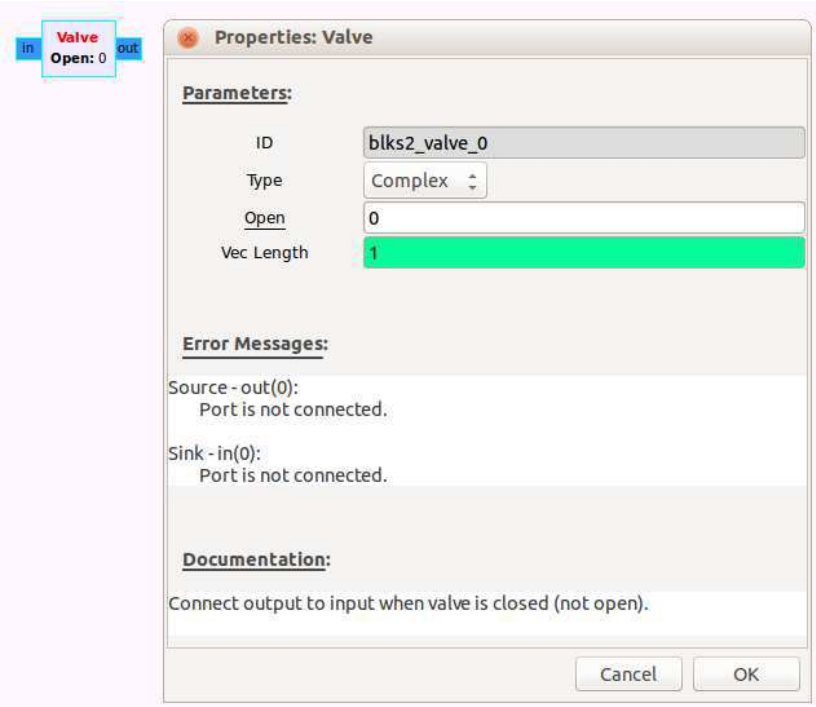

Figura 148. Bloque Valve y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

Type: especifica el tipo de dato del flujo de entrada y salida (ver Figura 147).

Open: es de tipo raw y cuando este parámetro evalua *True* la salida será desconectada de la entrada.

Vec Length: este parámetro es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Típicamente las aplicaciones utilizan el valor por defecto 1.

11.1.4.7. Throttle (Acelerador)

Este bloque se encarga de limitar el rendimiento de datos a la velocidad de muestreo especificada. Esto evita que GNU Radio consuma todos los recursos de la CPU cuando el diagrama de flujo no está siendo regulado por hardware externo (es decir, source / sink de audio o source / sink USRP). En la Figura 149 se muestra el bloque *Throttle* de forma gráfica junto con sus propiedades.

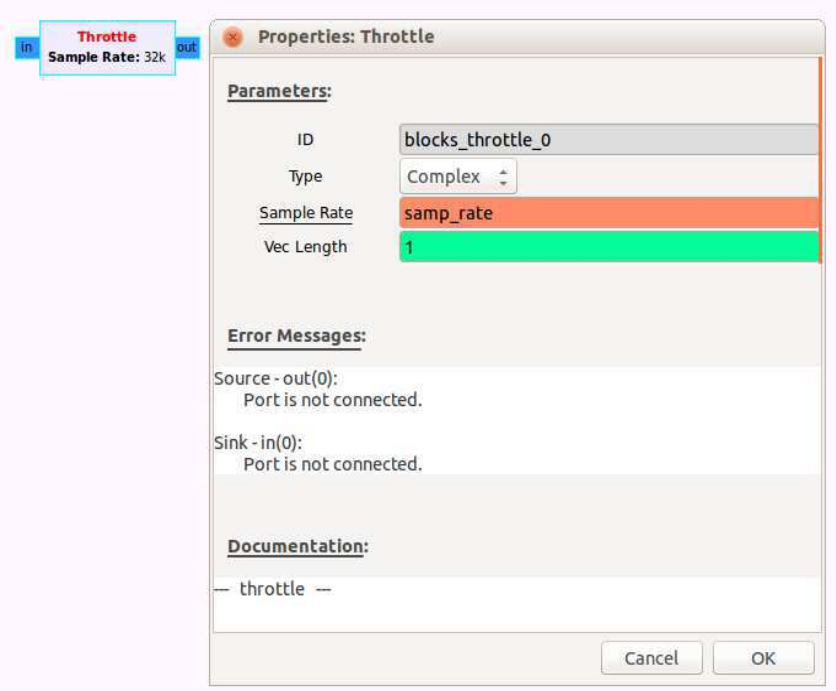

Figura 149. Bloque Throttle de forma junto con sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

Type: especifica el tipo de dato de entrada y salida, en la Figura 150 se muestran estos tipos.

Complex	Input and output are complex values.
Float	Input and output are floating point (real) values.
Int	Input and output are integer values.
Short	Input and output are short integer values.
Byte	Input and output are byte values.

Figura 150. Tipos de datos para el bloque Throttle. (DocBook, 2013)

Sample Rate: maneja datos de tipo real y especifica la velocidad de muestreo para limitar el diagrama de flujo.

Vec Length: datos tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.4.8. Delay (Retrasar)

Este bloque es el encargado de efectuar un retardo igual al número especificado de muestras. En la Figura 151 se observa este bloque junto con sus propiedades.

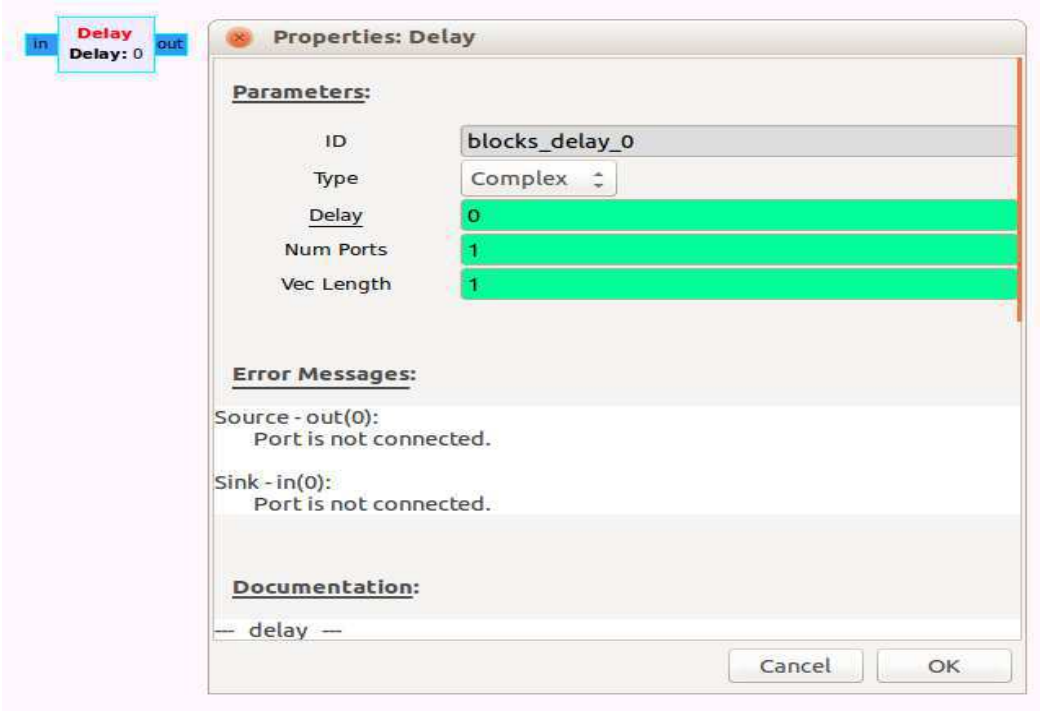

Figura 151. Bloque Delay y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cuatro parámetros más en sus propiedades, los cuales son:

Type: especifica el tipo de dato de entrada y salida (ver Figura 150).

Num Ports: datos tipo Int, especifica el número de entradas / salidas. Esencialmente crea líneas de retardo paralelas.

Vec Length: este parámetro es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Típicamente las aplicaciones utilizan el valor por defecto 1.

11.1.4.9. Null Source (fuente nula)

Este bloque emite ceros para cada muestra. Esto puede ser útil si se desea crear un valor complejo a partir de un valor de punto flotante con la parte real o imaginaria puesta a cero. En la Figura 152 se observa el bloque *Null Source* y sus propiedades.

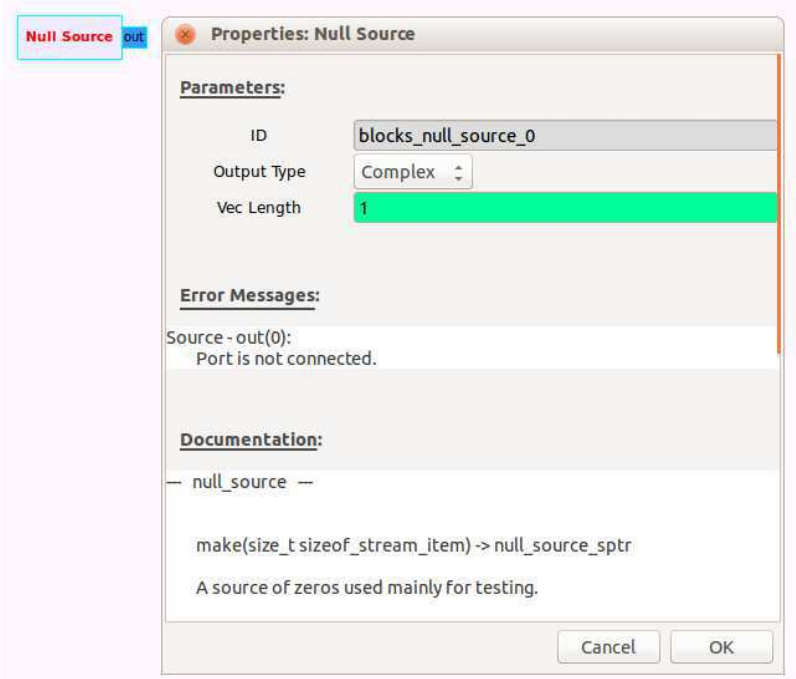

Figura 152. Bloque Null Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

Output Type: específica el tipo de datos de la salida, en la Figura 153 se muestran estos tipos de datos.

Complex	Output is complex value (0+j0).
Float	Output is floating point (real) value 0.
Int	Output is integer (32-bit) value 0.
Short	Output is short integer (16-bit) value 0.
Byte	Output is byte (8-bit) value 0.

Figura 153. Tipos de datos para el bloque Null Source. (DocBook, 2013)

Vec Length: datos tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.4.10. Null Sink

Este bloque descarta todos los datos de entrada. Esto puede ser útil si sólo se requiere una salida de un bloque de salida múltiple, pero GNU Radio requiere que todas las salidas estén conectadas. En la Figura 154 se muestra este bloque con sus propiedades.

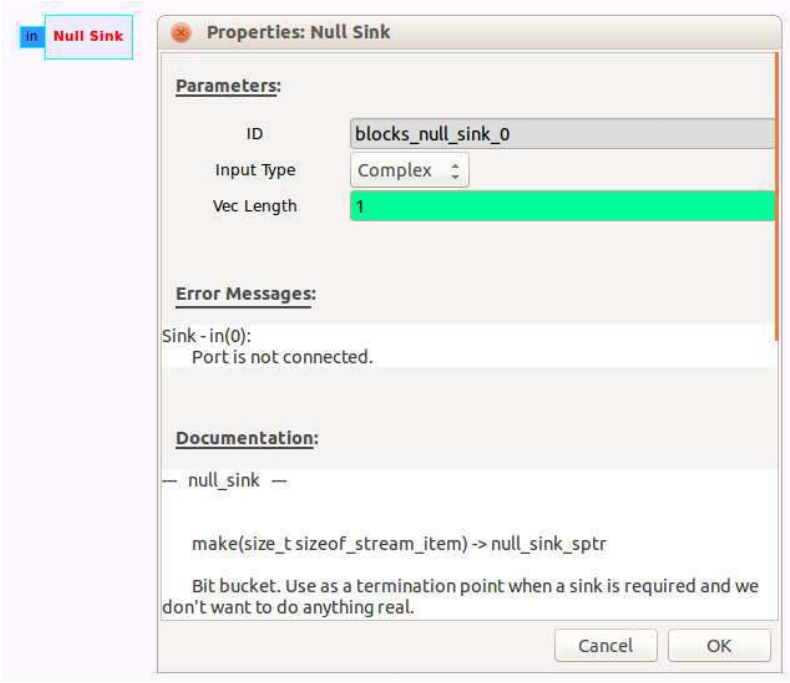

Figura 154. Bloque Null Sink y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

Input Type: específica el tipo de flujo de datos de la entrada, en la Figura 155 se muestran estos tipos de datos.

Complex	Input stream is complex.
Float	Input stream is real.
Int	Input stream is 32-bit integer.
Short	Input stream is 16-bit integer.
Byte	Input stream is 8-bit byte.

Figura 155. Tipos de datos para el bloque Null Sink. (DocBook, 2013)

Vec Length: datos tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.5. Variables

11.1.5.1. Variable

Este bloque se encarga de asignar un valor a una variable única. Este bloque de variables no tiene representación gráfica y dicha variable puede ser referenciada (por *ID*) de otros bloques en el diagrama de flujo. En la Figura 156 se muestra el bloque *Variable* en su forma visual.

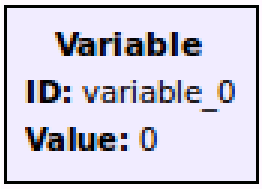

Figura 156. Bloque Variable. (Sotfware GRC, 2017)

En esta Figura se evidencia aparte del *ID* un parámetro más llamado:

Value: este parámetro es de tipo raw y especifica el valor de la variable.

11.1.5.2. Variable Config (Configuración de variables)

Este bloque es la representación de una variable que se puede leer a partir de un archivo de configuración, como forma de hacer que el diagrama de flujo guarde la configuración entre las diferentes ejecuciones. En la Figura 157 se muestra el bloque *Variable Config* y sus propiedades.

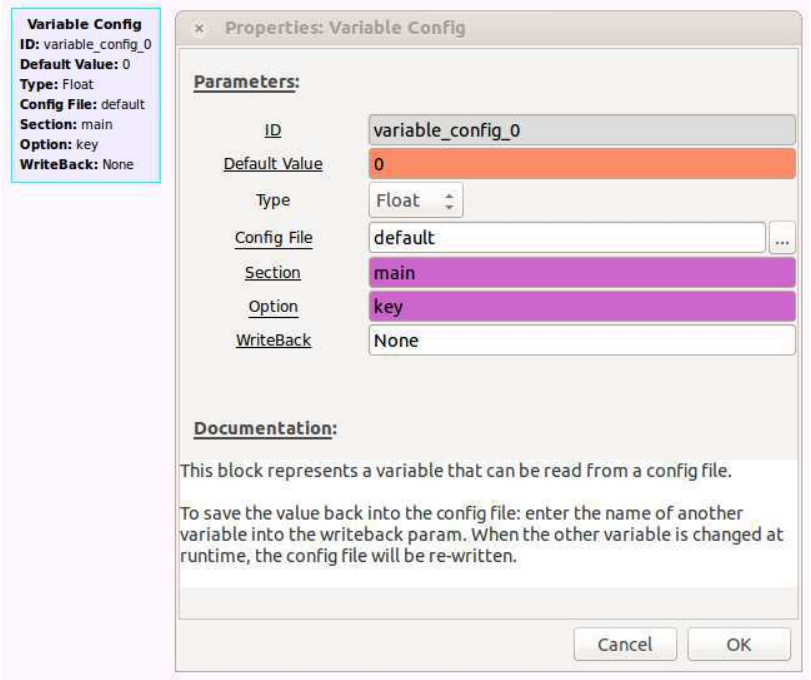

Figura 157. Bloque Variable Config y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* seis parámetros más en sus propiedades, los cuales son:

Default Value: su tipo de datos es igual que en el parámetro *Type* y establece el valor predeterminado si el parámetro nombrado en el campo *Option* no se encuentra en el archivo de configuración especificado.

Type: especifica el tipo de dato de la variable. En la Figura 158 se indican los tipos de datos.

Float	Variable is interpreted as a real value.
Int	Variable is interpreted as an integer value.
Bool	Variable is interpreted as a Boolean (True/False) value.
String	Variable is interpreted as a string.

Figura 158. Tipos de datos. (DocBook, 2013)

Config File: maneja tipos de datos file_open y especifica el archivo que se utilizará para guardar la configuración del diagrama de flujo. En algunas versiones de GNU Radio, este archivo ya existe. Si no este no aparece puede crear un nuevo archivo vacío en la línea de comandos utilizando el comando "touch" o crear un archivo a través del explorador de archivos de Linux.

Section: este parámetro maneja datos de tipo string y específica en qué sección se colocará la variable. Esto permite agrupar las variables en diferentes secciones dependiendo de la función.

Option: también maneja datos de tipo string y es el encargado de llamar la variable en el archivo de configuración. El archivo de configuración será texto sin formato, y se puede editar con un editor de texto.

WriteBack: maneja datos de tipo raw y se utiliza para guardar nuevamente el valor en el archivo de configuración, introduciendo el nombre de otra variable en el parámetro *WriteBack*. Cuando se cambia la otra variable en tiempo de ejecución, el archivo de configuración se volverá a escribir.

11.1.5.3. Parameter (Parámetro)

Este bloque representa un parámetro (diferente a los parámetros dentro de cada bloque) para el gráfico de flujo. Un parámetro se puede utilizar para pasar argumentos de línea de comandos a un bloque superior. O también, pueden pasar los argumentos a un bloque jerárquico instanciado. El valor del parámetro no puede depender de ninguna variable. Cuando type no es None, este parámetro también se convierte en una opción de línea de comandos del formulario: - [short_id] - [id] [value] El campo Short ID puede dejarse en blanco.

En este bloque encontramos 4 parámetros en sus propiedades los cuales son:

Label: este es de tipo string y se debe dejar en blanco para usar el parámetro id como etiqueta.

Value: maneja los mismos tipos de datos que *Type* y establece el valor predeterminado si no se especifica en la linea de comandos.

Type: este parámetro especifica el tipo de datos de la variable resultante. En la Figura 159 se especifican estos tipos de datos.

None	No type.
Complex	Parameter is interpreted as complex.
Float	Parameter is interpreted as real.
Int	Parameter is interpreted as an integer.
Long	Parameter is interpreted as a long integer.
String	Parameter is interpreted as a string.

Figura 159. Tipos de datos para Type en el bloque Parameter. (DocBook, 2013)

Short ID: tipos de datos string, es un formulario opcional de forma *short-hand* para el parámetro. Este se puede dejar en blanco.

11.1.6. Audio

11.1.6.1. Wav File Source (Fuente de archivos Wav)

Este bloque lo que hace es crear un origen de datos de un archivo de audio de onda. Este archivo puede ser capturado en GNU Radio con un *Wav File Sink* o creado en un editor de audio como Audacity (con las opciones WAV apropiadas) o en MATLAB usando el comando "wavwrite". En la Figura 160 se observa el bloque *Wav File Source* y sus propiedades.

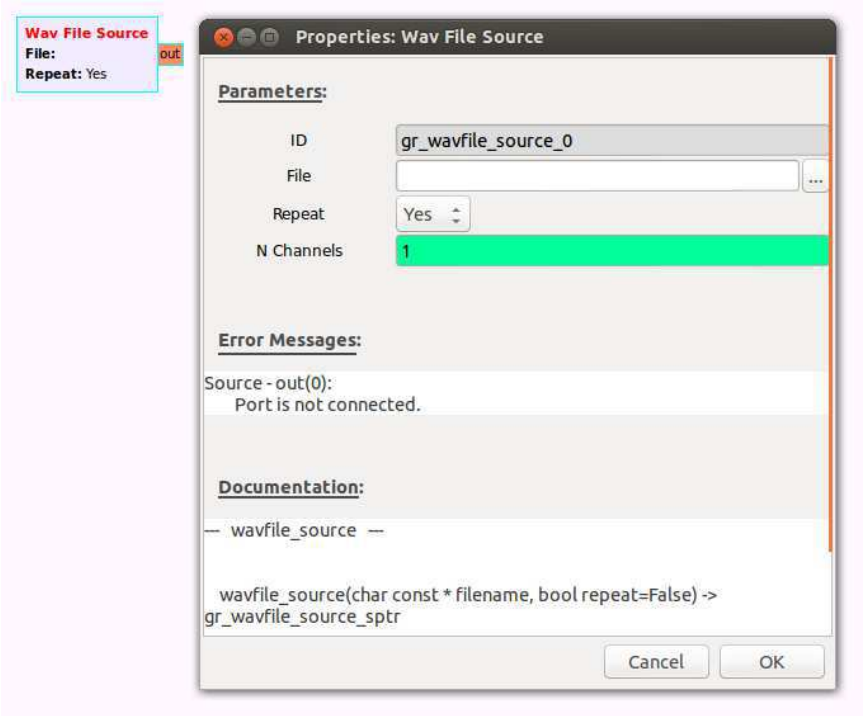

Figura 160. Bloque Wav File Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

File: este parámetro es de tipo filename y establece el nombre de archivo que se usara como origen de datos.

Repeat: este especifica si se reproduce o no el archivo en el bucle. Para esto se cuentan con dos opciones *"Yes y No"*, con *Yes* se reproducirá continuamente el archivo en el bucle y con *No* no se emite ninguna muestra una vez que el archivo haya llegado al final. Esto puede hacer que el diagrama de flujo parezca colgado.

N Channels: maneja datos tipo Int y especifica el número de canales que se van a leer del archivo. Los archivos Wav suelen ser mono (un canal, que representa una única secuencia de datos) o estéreo (dos canales, que representan dos flujos de datos independientes o los componentes en fase y en cuadratura de una forma de onda compleja).

11.1.6.2. Wav File Sink (Sink de Archivos Waw)

Este bloque Escribe una secuencia tipo float en un archivo Wav. Este archivo se puede reproducir en GNU Radio con una Fuente de Archivo Wav o reproducirse en cualquier otro reproductor de archivos Wav. También se puede cargar en MATLAB utilizando el comando "wavread". En la Figura 161 se muestra este bloque con sus características.

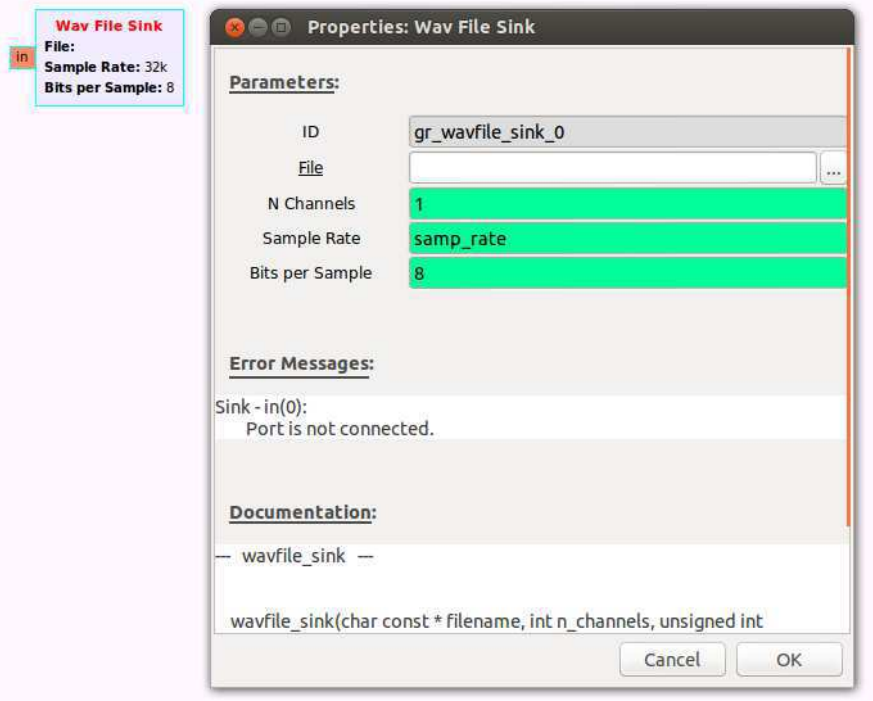

Figura 161. Bloque Wav File Sink y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cuatro parámetros más en sus propiedades, los cuales son:

File: tipo file_save y establece el nombre del archivo a escribirse.

N Channels: es de tipo Int y especifica el número de canales que se van a leer del archivo. Los archivos Wav suelen ser mono (un canal, que representa una única secuencia de datos) o estéreo (dos canales, que representan dos flujos de datos independientes o los componentes en fase y en cuadratura de una forma de onda compleja).

Sample Rate: este parámetro es de tipo Int y establece la velocidad de muestreo que se escribirá en el encabezado del archivo Wav. Para la compatibilidad con otras aplicaciones debe ser una de las tasas de muestreo estándar para archivos Wav como 8000, 11025, 22050, 44100, 48000, etc. La velocidad de muestreo del flujo de entrada debe coincidir.

Bits per Sample: de tipo Int y establece la profundidad de bits del archivo de audio. Normalmente, la resolución de 16 bits es suficiente.

11.1.6.3. Audio Sink

Este bloque representa al hardware de salida de audio dentro del grafico de GRC, la señal se reproducirá a través de los altavoces a menos que haya algo conectado a la salida de auriculares en el panel. En la Figura 162 se muestra al bloque *Audio Sink* de forma visual.

Figura 162. Bloque Audio Sink. (Sotfware GRC, 2017)

Este bloque presenta cuatro parámetros los cuales son:

Sample Rate: de tipo Int y especifica la frecuencia de muestreo que se va a utilizar. No todas las frecuencias de muestreo son compatibles con el hardware de audio. Para aplicaciones típicas, esto debe establecerse en 48kHz.

Device Name: parámetro de tipo string, en este se deja el nombre de dispositivo en blanco para elegir el dispositivo de audio por defecto.

OK to Block: en este parámetro el valor predeterminado es *True.* Para dejarlo como predeterminado o no, existen dos opciones: *Yes* y *No,* la primera lo deja como ajuste predeterminado y la segunda lo deja como ajuste no predeterminado.

Num Inputs: es de tipo Int específica el número de canales a escribir en el dispositivo de salida de audio. Para aplicaciones típicas con transmisores I / Q como el kit Softrock, se establece este valor en 2. Para aplicaciones de salida de audio mono, se debe ajustar este valor en 1.

11.1.6.4. Audio Source (Fuente de audio)

Este bloque representa al hardware de entrada de audio en el gráfico de GRC. En la Figura 163 se observa a este bloque en su forma gráfica.

Figura 163. Bloque Audio Source. (Sotfware GRC, 2017)

Este bloque presenta cuatro parámetros los cuales son:

Sample Rate: de tipo Int y especifica la frecuencia de muestreo que se va a utilizar. No todas las frecuencias de muestreo son compatibles con el hardware de audio. Para aplicaciones típicas, esto debe establecerse en 48kHz.

Device Name: de tipo Int y se debe dejar el nombre del dispositivo en blanco para elegir el dispositivo de audio por defecto.

OK to Block: en este parámetro el valor predeterminado es *True.* Para dejarlo como predeterminado o no, existen dos opciones: *Yes* y *No,* la primera lo deja como ajuste predeterminado y la segunda lo deja como ajuste no predeterminado.

Num Outputs: es de tipo Int específica el número de canales a leer en el dispositivo de entrada de audio. Para aplicaciones típicas con receptores I / Q como el kit Softrock, se establece este valor en 2. Para aplicaciones de entrada mono, se debe ajustar este valor en 1.

11.1.7. Boolean Operators

11.1.7.1. And (Y)

Este bloque implementa la función lógica out = in0 AND in1 AND… AND in(N-1). Como se sabe en GRC se trabaja con señales digitales, es decir en forma binaria de “1” y “0” este bloque de expresión booleana *AND* básicamente es como si se trabajara Figuras de verdad que para el caso de *AND* su Figura se muestra en la Figura 164, está básicamente muestra que para que la operación *AND* entregue un uno en la salida, es necesario que todas las entradas también estén en uno, basta con que alguna de ellas este en cero para que en la salida automáticamente se vea un cero.

A	B	Salida
0	0	0
0	1	0
1	0	0
1	1	1

Figura 164. Figura de verdad Operación AND. (Electrontools, s.f.)

En la Figura 165 se observa el bloque *AND* con sus propiedades.

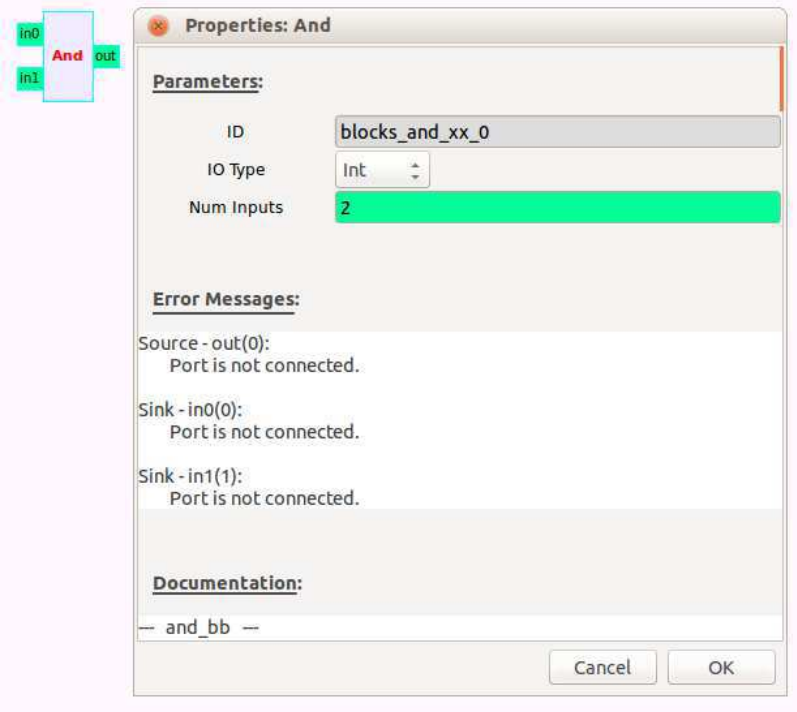

Figura 165. Bloque AND y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos de la entrada y la salida, los tipos son Int, Short y Byte. En la Figura 166 se muestran estos tipos de datos.

Int	Sets the input and output to integer (signed, 32-bit) values.
Short	Sets the input and output to short (signed, 16-bit) values.
Byte	Sets the input and output to byte (unsigned, 8-bit) values.

Figura 166. Tipos de datos para IO Type. (DocBook, 2013)

Num Inputs: este parámetro es de tipo Int y establece el número de entradas para el bloque.

11.1.7.2. And Const (Y Constante)

Este bloque implementa la función out = in AND const. Esto puede ser útil para ocultar ciertos bits en un byte (obligarlos a cero). En la Figura 167 se muestra el bloque *And Const* y sus propiedades.

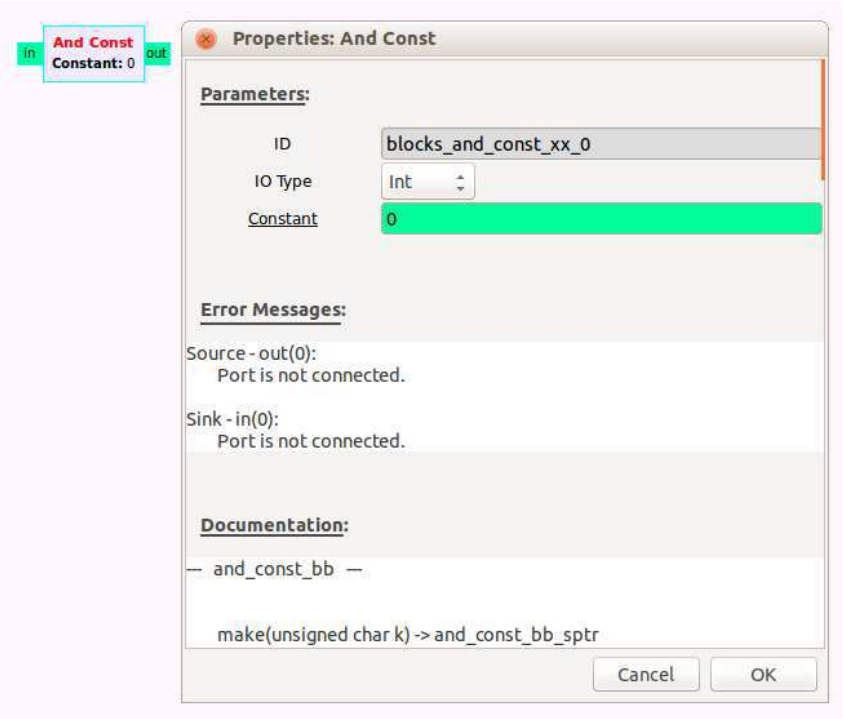

Figura 167. Bloque And Const y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos de la entrada y la salida (ver Figura 166).

Constant: define el valor constante que se utilizara como and-mask.

11.1.7.3. Not (No)

Con este bloque se implementa la función lógica booleana *Not,* esta realiza la siguiente operación out = NOT in. Es decir que invierte todos los bits de entrada, prácticamente lo que

hace es negarlos, si entra un 1 sale un 0 y viceversa. En la Figura 168 se muestra este bloque con sus propiedades.

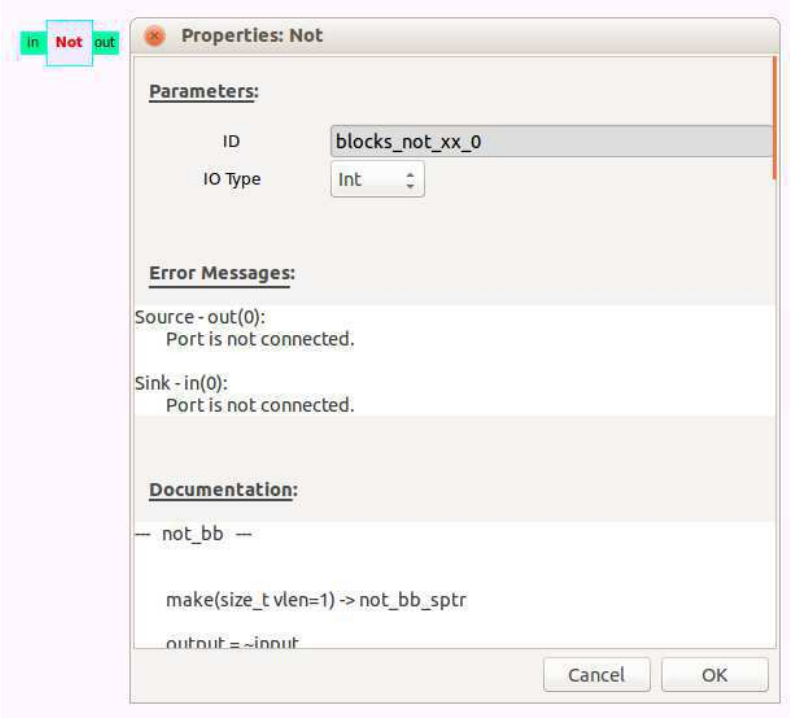

Figura 168. Bloque Not y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* un parámetro más en sus propiedades, el cual es:

IO Type: especifica el tipo de datos de la entrada y la salida (ver Figura 166).

11.1.7.4. Or (O)

Este bloque implementa la función lógica booleana *Or* out = in0 OR in1 OR ... OR in(N-1). La función *Or* puede ser útil si se necesita forzar algunos bits a 1. En la Figura 169 se muestra el bloque *Or* y sus propiedades.

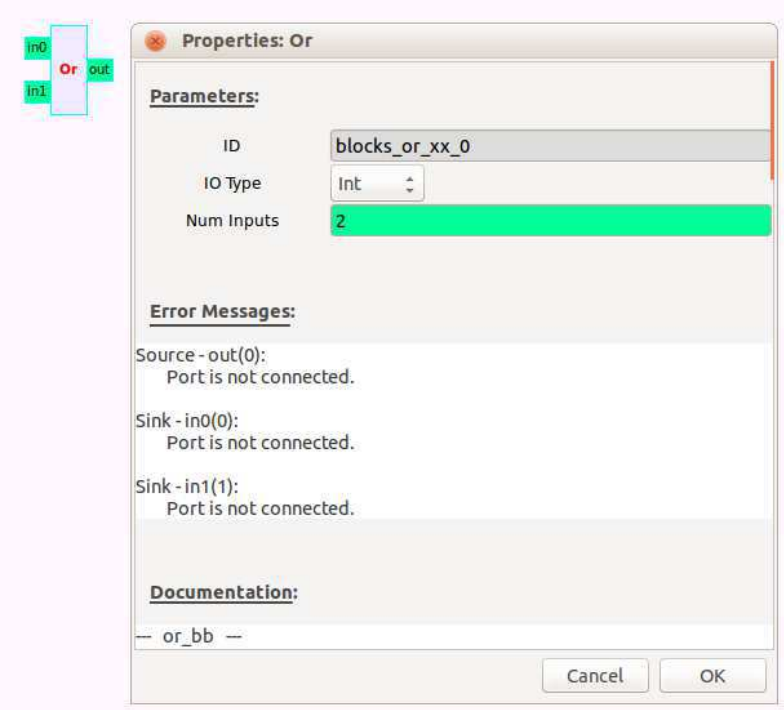

Figura 169. Bloque Or y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos de la entrada y la salida (ver Figura 166).

Num Inputs: este parámetro es de tipo Int y establece el número de entradas para el bloque.

11.1.7.5. Xor

Este bloque implementa la función lógica booleana *Xor* out = in0 XOR in1 XOR ... XOR in(N-1). La función *Xor* puede ser útil si desea invertir ciertos bits. Cuando las entradas contienen bits iguales su salida será 0 de lo contrario será 1. En la Figura 170 se observa este bloque y sus respectivas propiedades.

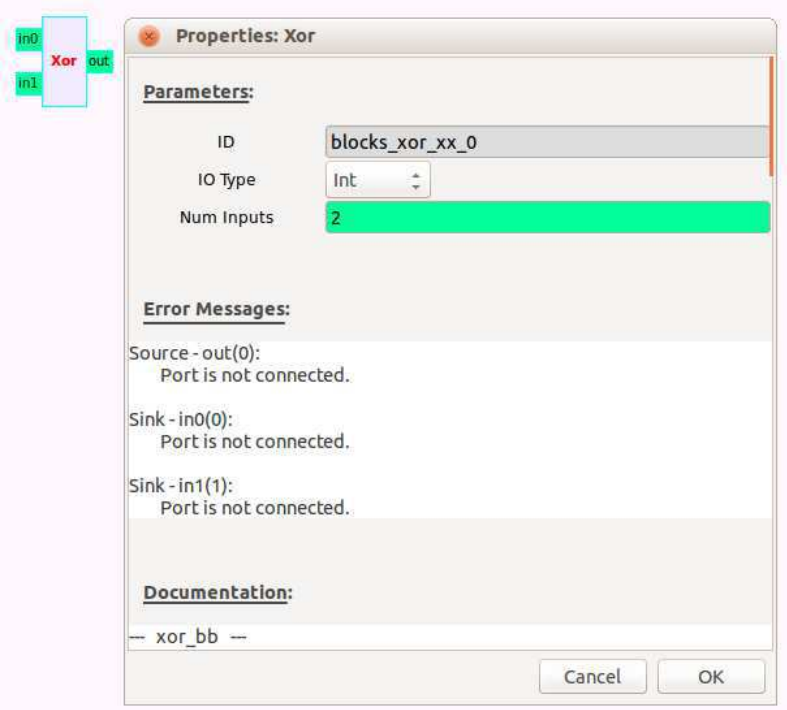

Figura 170. Bloque Xor y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos de la entrada y la salida (ver Figura 166).

Num Inputs: este parámetro es de tipo Int y establece el número de entradas para el bloque.

11.1.8. File Operators

11.1.8.1. Wav File Source

Este bloque es el mismo *Wav File Source* que forma parte del módulo Audio (ver numeral 11.1.5.1).

11.1.8.2. Wav File Sink

Este bloque es el mismo *Wav File Sink* que forma parte del módulo Audio (ver numeral 11.1.5.2).

11.1.8.3. File Source (Origen del archivo)

Este bloque lee valores de datos sin formato en formato binario del archivo especificado. Este archivo puede ser: un archivo capturado mediante el uso de un bloque *File Sink*, generado con un programa de computadora o guardado desde un editor de audio como Audacity (usando opciones de formato RAW). En la Figura 171 se observa el bloque *File Source* y sus propiedades.

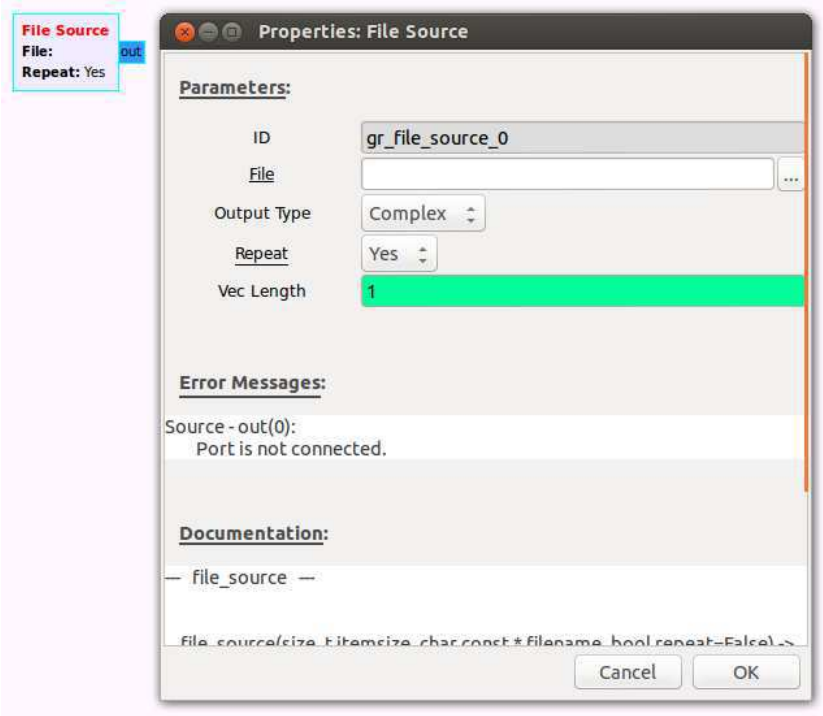

Figura 171. Bloque File Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cuatro parámetros más en sus propiedades, los cuales son:

File: es de tipo file_open y especifica el nombre de archivo de entrada.

Output Type: este parámetro especifica el tipo de datos de la salida (ver Figura 153).

Repeat: con este parámetro se selecciona si se desea leer el archivo una vez o reproducirlo en el bucle. Permite dos opciones *Yes* y *No,* con Yes establece que cuando se alcanza el final del archivo, vuelve al inicio del mismo y continua, y con No se especifica que solo se leera el archivo hasta que se alcance el final y no se repetirá.

Vec Length: de tipo Int, con este parámetro se especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.8.4. File Sink

Este bloque emite valores de datos sin formato en formato binario hacia el archivo especificado. Este archivo se puede leer en cualquier entorno de programación que permita leer archivos binarios (MATLAB, C, Python, etc.). También se puede reproducir en GRC utilizando un bloque apropiado de *File Source*. En la Figura 172 se muestra este bloque de forma gráfica con sus respectivas propiedades.

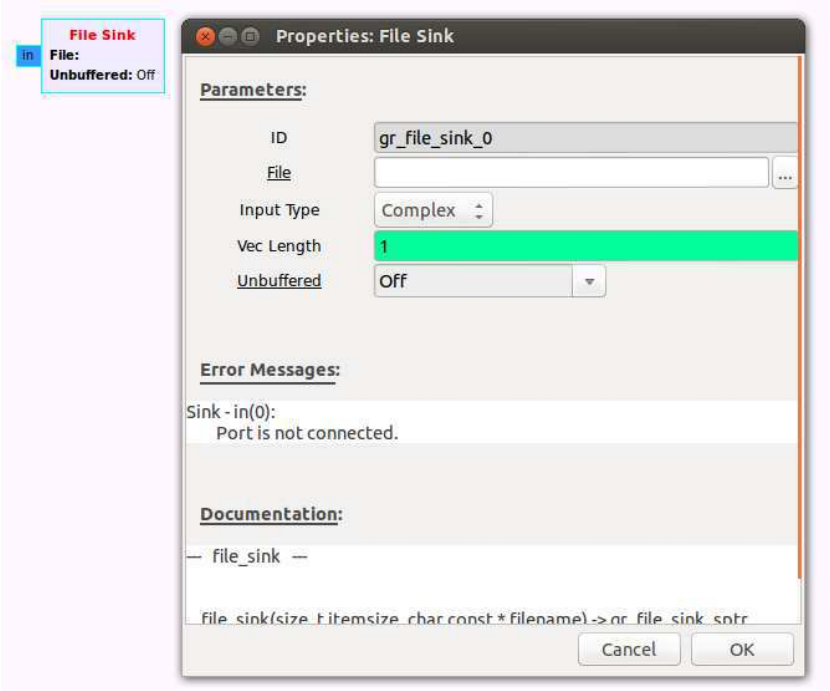

Figura 172. Bloque File Sink y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cuatro parámetros más en sus propiedades, los cuales son:

File: es de tipo file_open y especifica el nombre de archivo de salida.

Output Type: este parámetro especifica el tipo de datos de la entrada (ver Figura 155).

Vec Length: de tipo Int, con este parámetro se especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

Unbuffered: especifica si la salida está almacenada en la memoria intermedia. Si la salida es *Unbuffered*, los datos serán arrojados al archivo cada vez que se llama a la función de trabajo. Esto puede hacer que el diagrama de flujo se ejecute lento debido al tiempo necesario para acceder al disco cada vez.

11.1.9. Math Operators (Operadores matemáticos)

11.1.9.1. Add (Adición)

Este bloque implementa la función out = in0 + in1 + ... + in(N-1), básicamente lo que hace este operador matemático es sumar las entradas y el resultado emitido en la salida. En la Figura 173 se observa el bloque *Add* y sus propiedades.

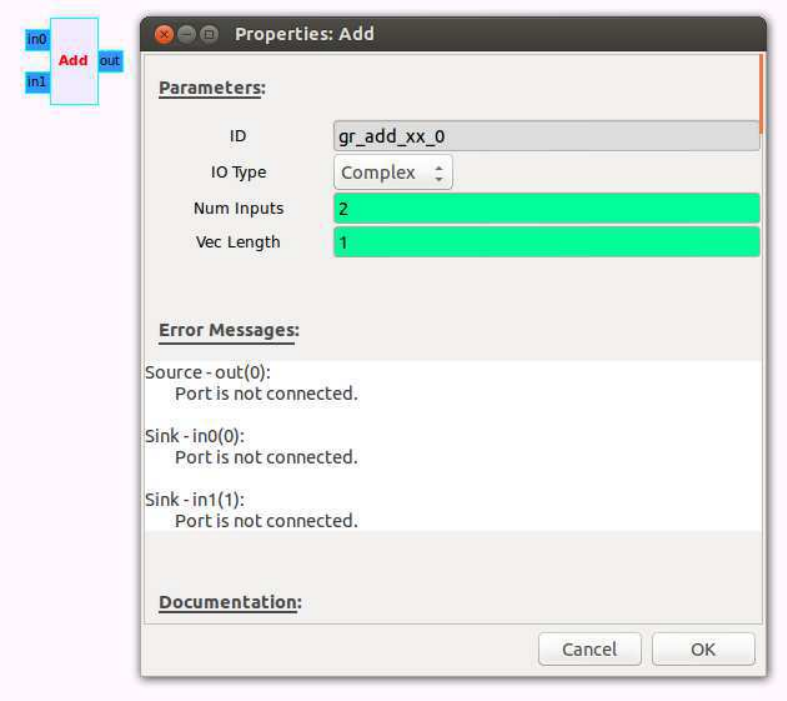

Figura 173. Bloque Add y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos de las entradas y la salida: *Complex, Float, Int y Short*. En la Figura 174 se relacionan estos tipos de datos.

Complex	Inputs and output are complex values.
Float	Inputs and output are floating point (real) values.
Int	Inputs and output are integer values.
Short	Inputs and output are short integer values.

Figura 174. Tipos de datos para IO Type bloque Add. (DocBook, 2013)

Num Inputs: este parámetro es de tipo Int y especifica el número de salidas. Estas salidas serán etiquetadas desde in0 hasta in(N-1)

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.9.2. Add Const (Adición Constante)

Este bloque implementa la función out = in + Constant, básicamente realiza una suma entre la entrada y una constante que se determina en las propiedades del bloque. En la Figura 175 se muestra este bloque con sus propiedades.

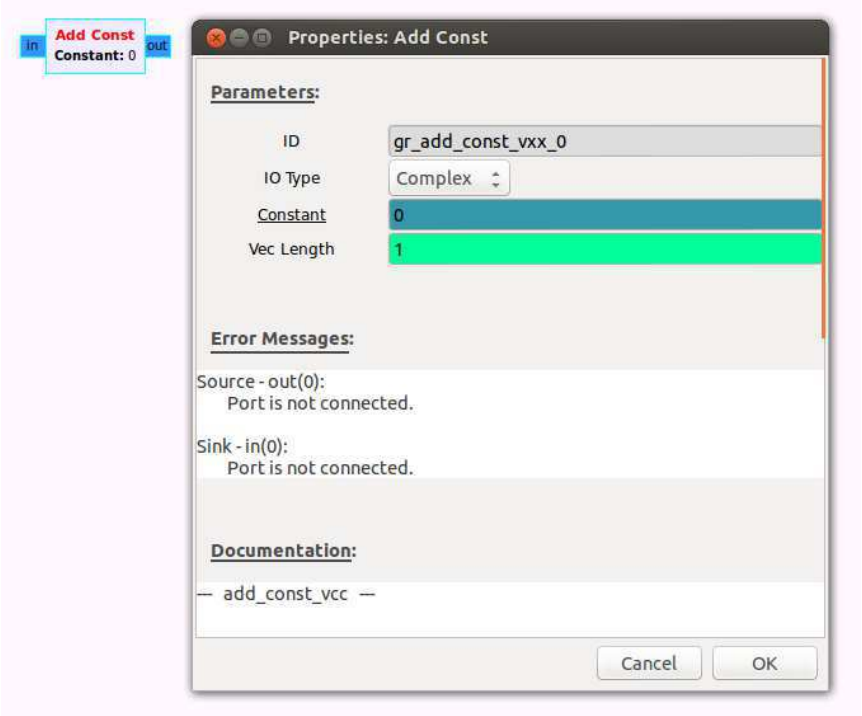

Figura 175. Bloque Add Const y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos de la entrada y salida. (Ver Figura 174).

Constant: especifica la constante para adicionar.

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.9.3. Divide (Dividir)

Este bloque implementa la función out = in0/in1…… /in(N-1). Es decir que en la salida se emite el resultado de la división de las entradas. En la Figura 176 se muestra el bloque *Divide* y sus propiedades.

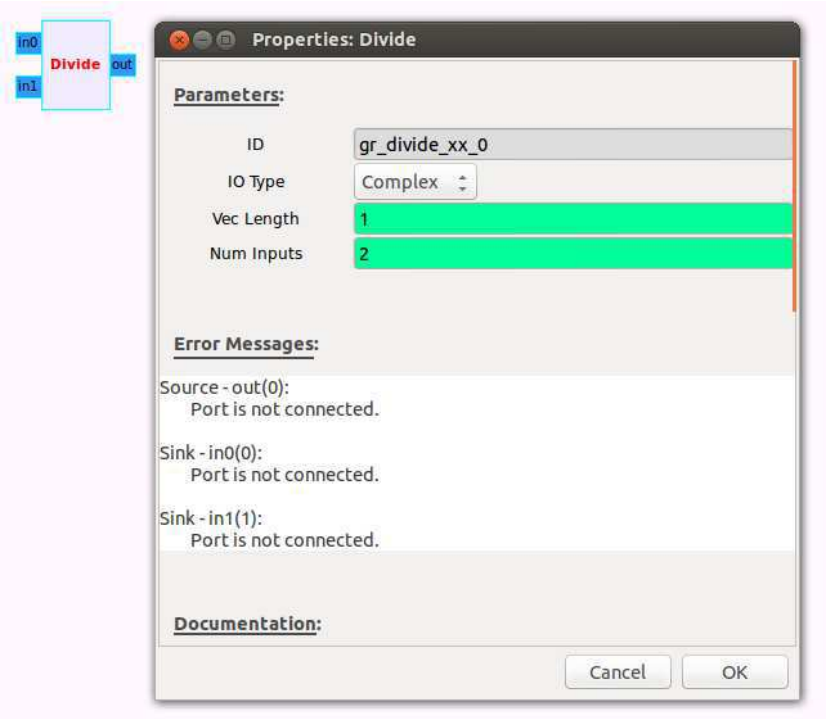

Figura 176. Bloque Divide y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de dato de las entradas y la salida (ver Figura 174).

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

Num Inputs: este parámetro es de tipo Int y especifica el número de salidas. Estas salidas serán etiquetadas desde in0 hasta in(N-1). Al establecer este valor en 1 calcula el recíproco de in0.

11.1.9.4. Multiply (Multiplicar)

Este bloque implementa la función out = in0 x in1 x ... x in(N-1). Básicamente este bloque realiza una multiplicación con las señales de entrada y el resultado es emitido en la salida. En la Figura 177 se observa este bloque en su forma gráfica con sus propiedades.

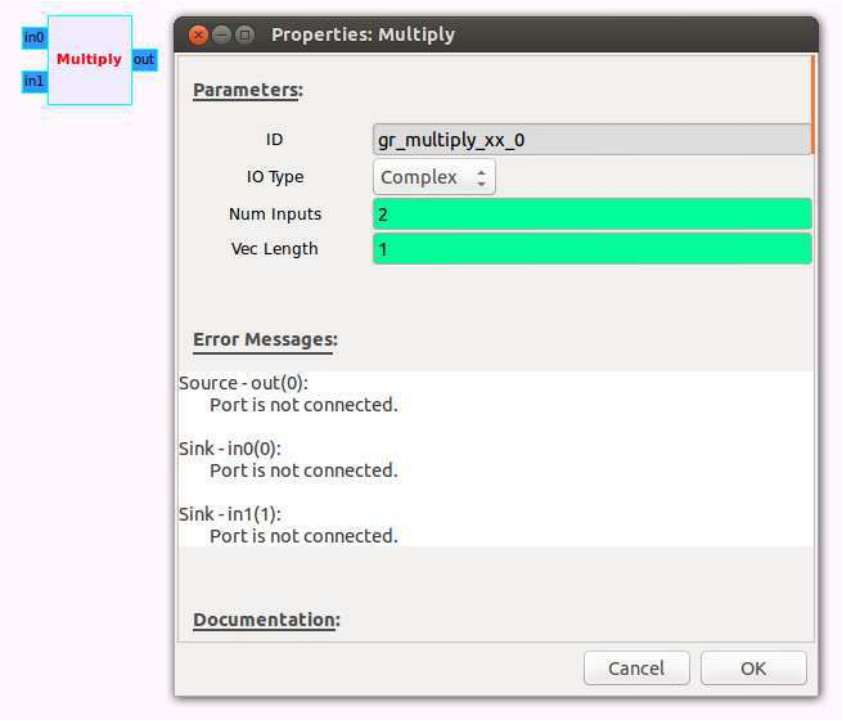

Figura 177. Bloque Multiply y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de dato de las entradas y la salida (ver Figura 174).

Num Inputs: este parámetro es de tipo Int y especifica el número de salidas. Estas salidas serán etiquetadas desde in0 hasta in(N-1).

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.9.5. Multiply Const (Multiplicar Constante)

Este bloque implementa la función out = in x Constant. Es decir que en este bloque se efectúa una multiplicación entre la entrada y una constante que se define en las propiedades. En la Figura 178 se muestre el bloque *Multiply Const* y sus propiedades.

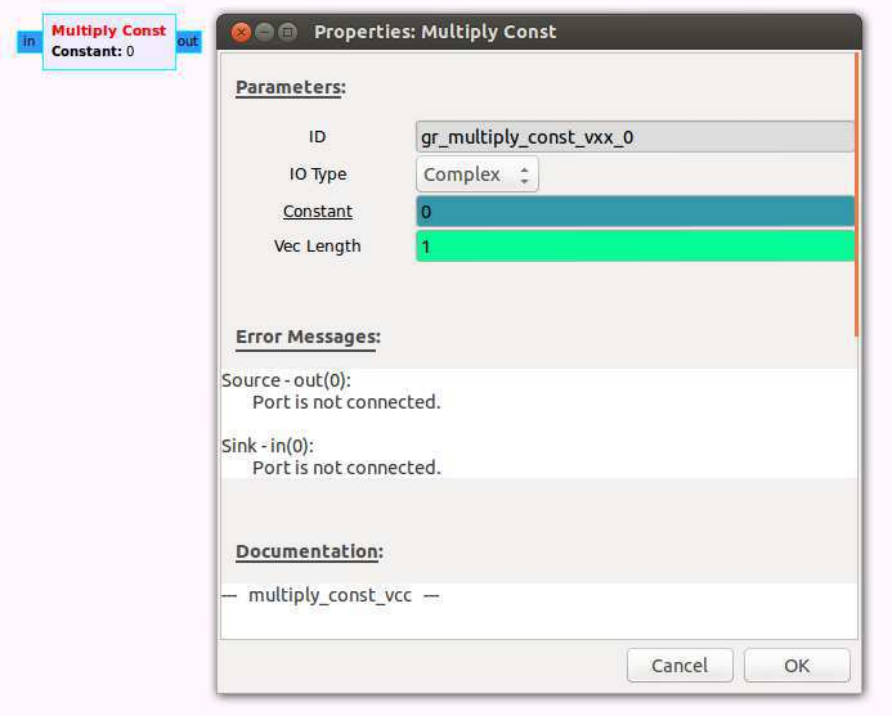

Figura 178. Bloque Multiply Const y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos de la entrada y salida. (Ver Figura 174).

Constant: especifica la constante para multiplicar con la entrada.

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.9.6. Subtract (Sustraer)

Este bloque implementa la función out = in0 – in1 - … - in(N-1). Básicamente lo que hace este bloque es una resta entre sus entradas y el resultado emitirlo en la salida. En la Figura 179 se observa el bloque *Subtrac* en su forma visual.

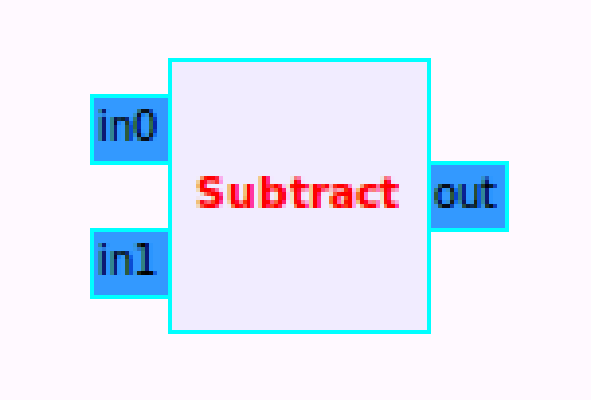

Figura 179. Bloque Subtrac. (Sotfware GRC, 2017)

Este bloque presenta tres parámetros en sus propiedades los cuales son:

IO Type: especifica el tipo de dato del flujo de las entradas y la salida (ver Figura 174).

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

Num Inputs: este parámetro es de tipo Int y especifica el número de salidas. Estas salidas serán etiquetadas desde in0 hasta in(N-1).

11.1.10. Message Tools (Herramientas de mensaje)

11.1.10.1. Message Source (Fuente del Mensaje)

Este bloque crea una fuente de datos a partir de un mensaje. El paso de mensajes es el mecanismo que permite que los datos fluyan entre las capas de Python y C ++ de GNU Radio. En la Figura 180 se observa este bloque con sus respectivas propiedades.

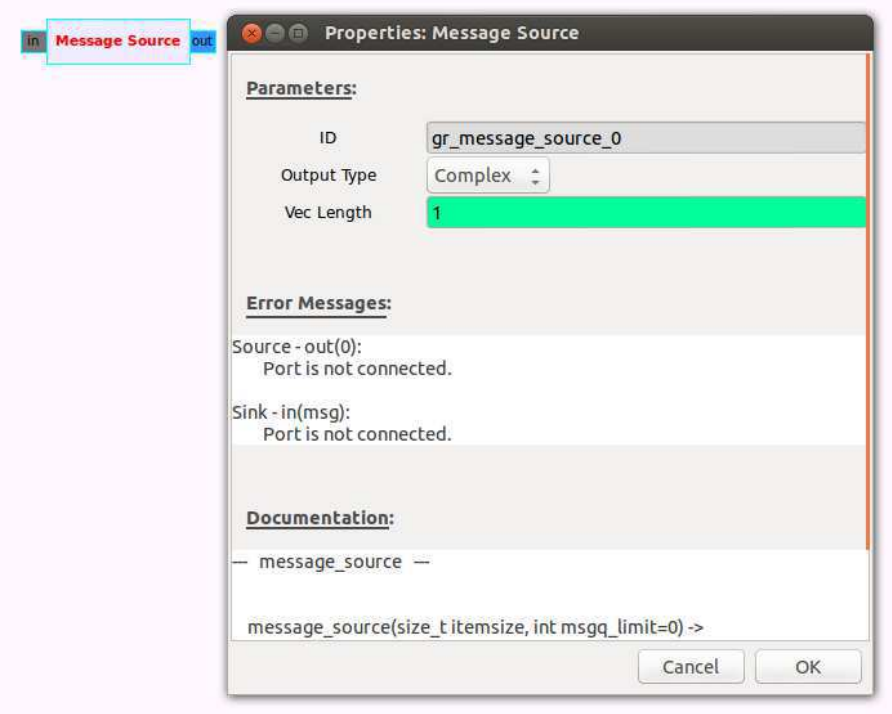

Figura 180. Bloque Message Source y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

Output Type: especifica el tipo de datos que se creara a partir del mensaje de entrada. En la Figura 181 se especifican los tipos de datos posibles: *Complex, Float, Int, Short y Byte.*

Complejo	La salida es de valor complejo.
Flotador	La producción es de valor real.
Int	La salida es un entero de 32 bits.
Corto	La salida es un entero de 16 bits.
Byte	La salida es un byte de 8 bits.

Figura 181. Tipos de datos para Output Type del bloque Message Source. (DocBook, 2013)

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.10.2. Message Sink

Este bloque crea mensajes de un flujo de datos entrante. El paso de mensajes es el mecanismo que permite que los datos fluyan entre las capas de Python y C ++ de GNU Radio.

Este bloque presenta tres parámetros aparte del *ID* los cuales se configuran en sus propiedades, estos parámetros son:

Input Type: especifica el tipo de datos del flujo de entrada. En la Figura 182 se especifican los tipos de datos posibles: *Complex, Float, Int, Short y Byte.*

Complejo	La transmisión de entrada es compleja.
Flotador	La transmisión de entrada es real.
Int	La secuencia de entrada es un entero de 32 bits.
Corto	El flujo de entrada es un entero de 16 bits.
Byte	El flujo de entrada es un byte de 8 bits.

Figura 182. Tipos de datos para Input Type del bloque Message Sink. (DocBook, 2013)

Don't Block: este parámetro se debe dejar como predeterminado, se encarga de bloquear o no bloquear el flujo de datos.

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.11. Stream Operators (Operadores de flujo)

11.1.11.1. Deinterleave (Desintercalado)

Este bloque entrelaza flujos de entrada. Los valores de entrada se asignarán sucesivamente a out0, out1, ..., out (N-1). Se debe tener en cuenta que la velocidad de datos en la salida es la velocidad de entrada dividida por la cantidad de flujos. En la Figura 183 se muestra el bloque *Deinterleave* y sus propiedades.

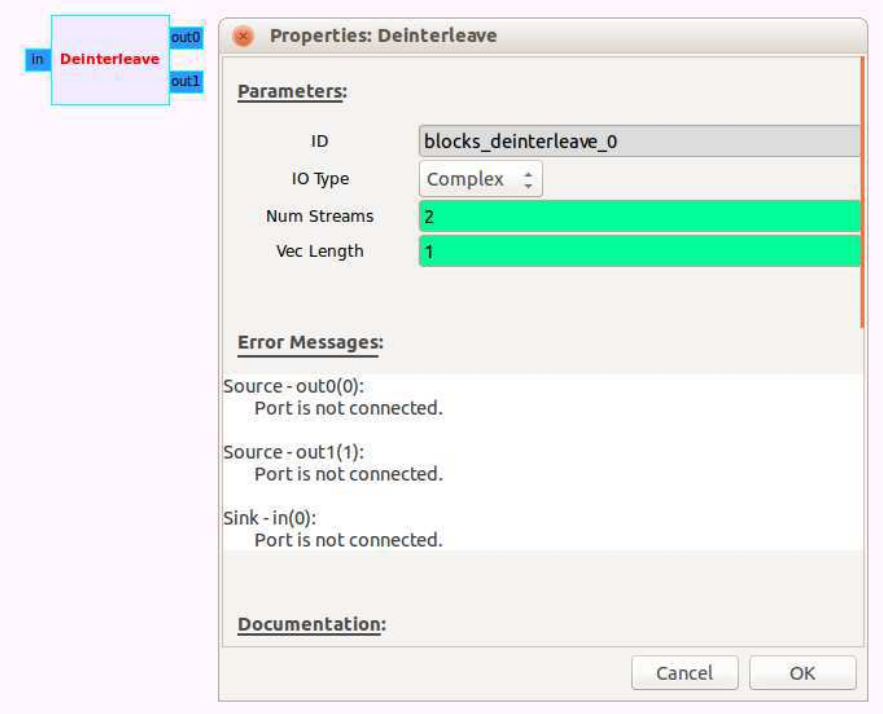

Figura 183. Bloque Deinterleave y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos del flujo de entrada y salida. (Ver Figura 147).

Num Streams: este parámetro es de tipo Int y especifica el número de flujos para el bloque.

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.11.2. Interleave (Intercalado)

Este bloque intercala las secuencias de entrada (in0, in1, ..., in(N-1)) en una única secuencia. Se debe en cuenta que la velocidad de datos de la secuencia entrelazada será la velocidad de las secuencias de entrada multiplicada por el número de secuencias. Todas las

secuencias de entrada deben tener la misma velocidad para una correcta operación. En la Figura 184 se observa este bloque con sus propiedades.

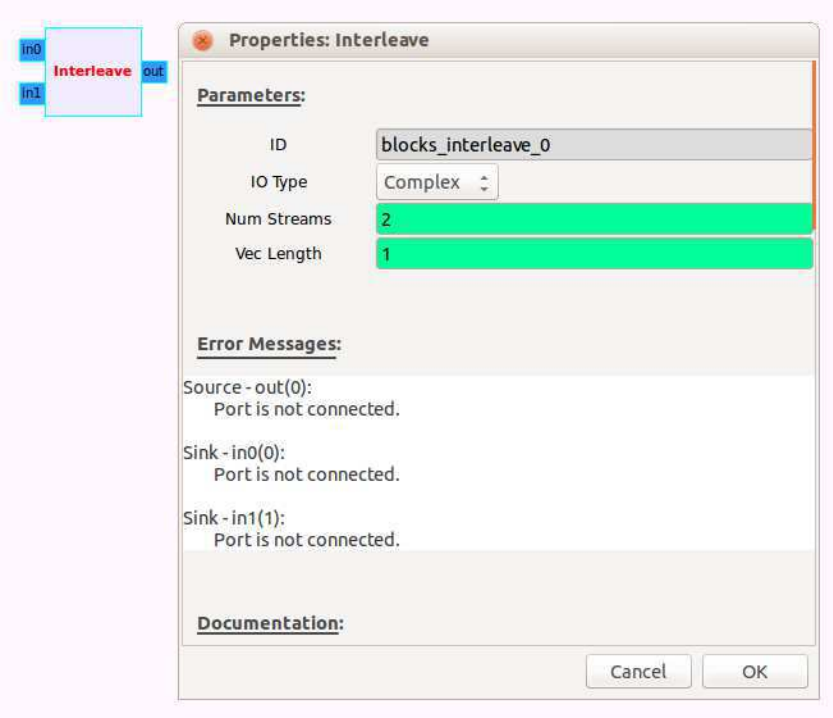

Figura 184. Bloque Interleave y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

IO Type: especifica el tipo de datos del flujo de entrada y salida. (Ver Figura 147)

Num Streams: este parámetro es de tipo Int y especifica el número de secuencias para intercalar. El bloque se creará con el número requerido de entradas.

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.11.3. Repeat (Repetir)

Este bloque interpola un flujo repitiendo cada valor de entrada un número específico de veces. Se debe tener en cuenta que la frecuencia de muestreo de salida es la frecuencia de muestreo de entrada multiplicada por el valor de interpolación especificado. En la Figura 185 se observa el bloque *Repeat* y sus propiedades.

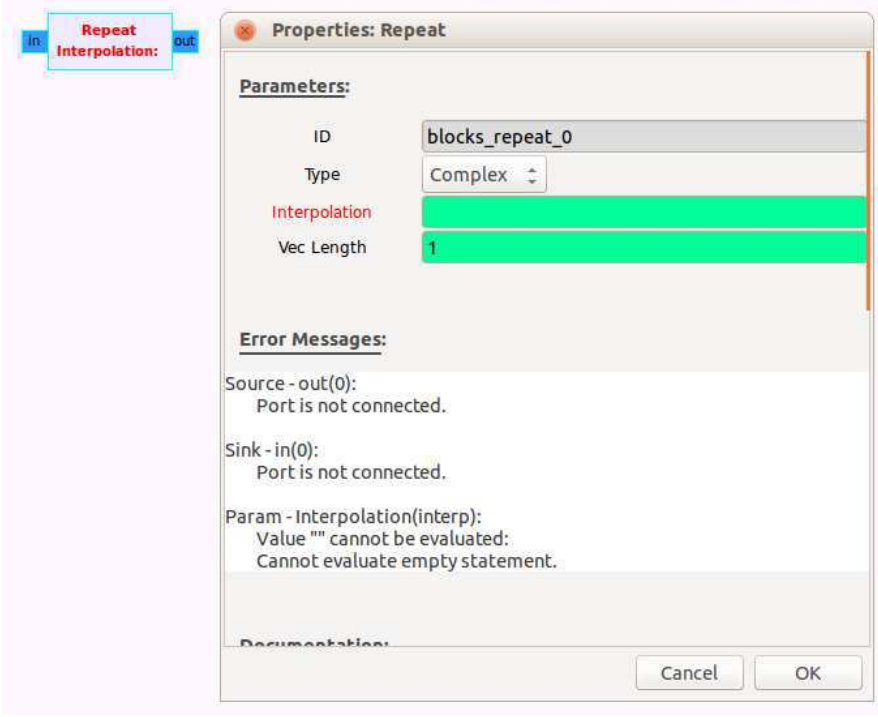

Figura 185. Bloque Repeat y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

Type: especifica el tipo de datos del flujo de entrada y salida. (Ver Figura 147)

Interpolation: este parámetro es de tipo Int y especifica el número de veces que se repetirá cada valor.

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.12. Type Converters

11.1.12.1. Char To Float

Este bloque convierte una secuencia del tipo de datos char (byte) al tipo de datos flotante (real). En la Figura 186 de muestra el bloque mencionado junto con sus propiedades.

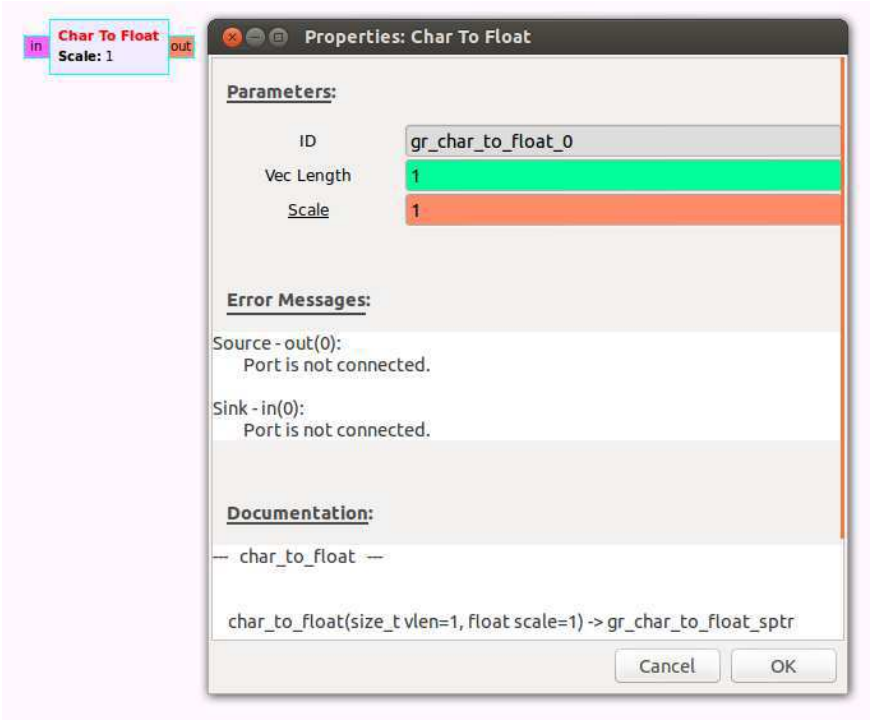

Figura 186. Bloque Char To Float y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* dos parámetros más en sus propiedades, los cuales son:

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

Scale: este parámetro es de tipo real y establece el factor de escala del valor de salida. El valor predeterminado de 1 dará 0 => 0.0, 1 => 1.0, 2 => 2.0, ..., 255 => 255.0. Si se desea normalizar el valor a 1.0, se ingresa 1.0 / 256.0.

11.1.12.2. Char To Short

Este bloque convierte una secuencia de datos de tipo char (byte) a tipo de datos corto (16 bits). En la Figura 187 se muestra el bloque *Char To Short* y sus propiedades.

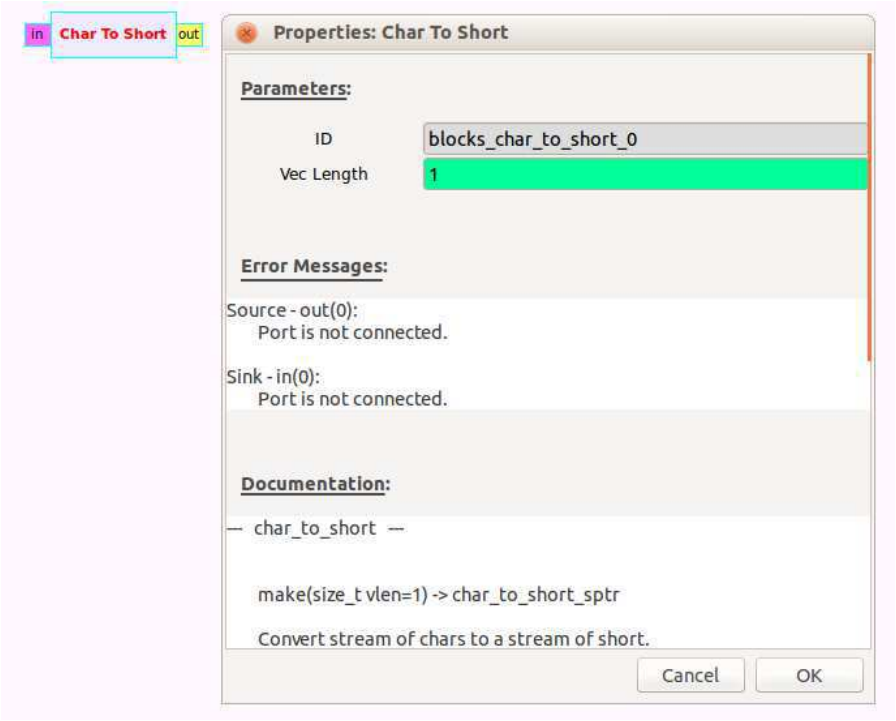

Figura 187. Bloque Char To Short y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta un parámetro más aparte del *ID* el cual es:

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.12.3. Complex To Float

Este bloque implementa las funciones: re = Real (in) im = Imag (in). Se debe tener en cuenta que el tipo de datos cambia de complejo a punto flotante (real). En la Figura 188 se muestra este bloque y sus propiedades.

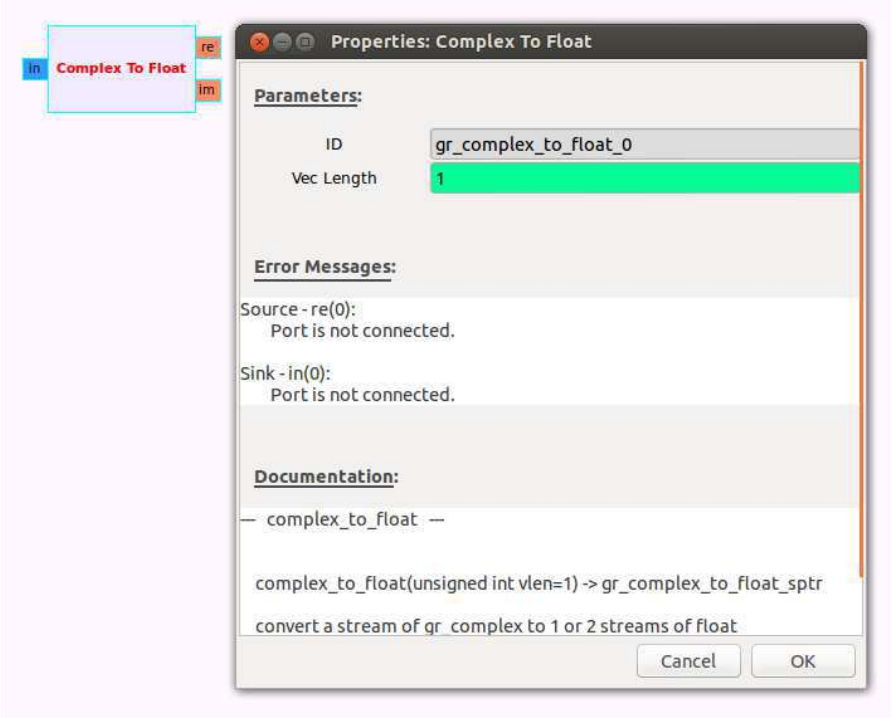

Figura 188. Bloque Complex To Float y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta un parámetro más aparte del *ID* el cual es:

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.12.4. Complex To Real

Este bloque implementa la función out = Real (in). Se debe tener en cuenta que el tipo de datos cambia de complejo a punto flotante (real). En la Figura 189 se observa este bloque con sus propiedades.

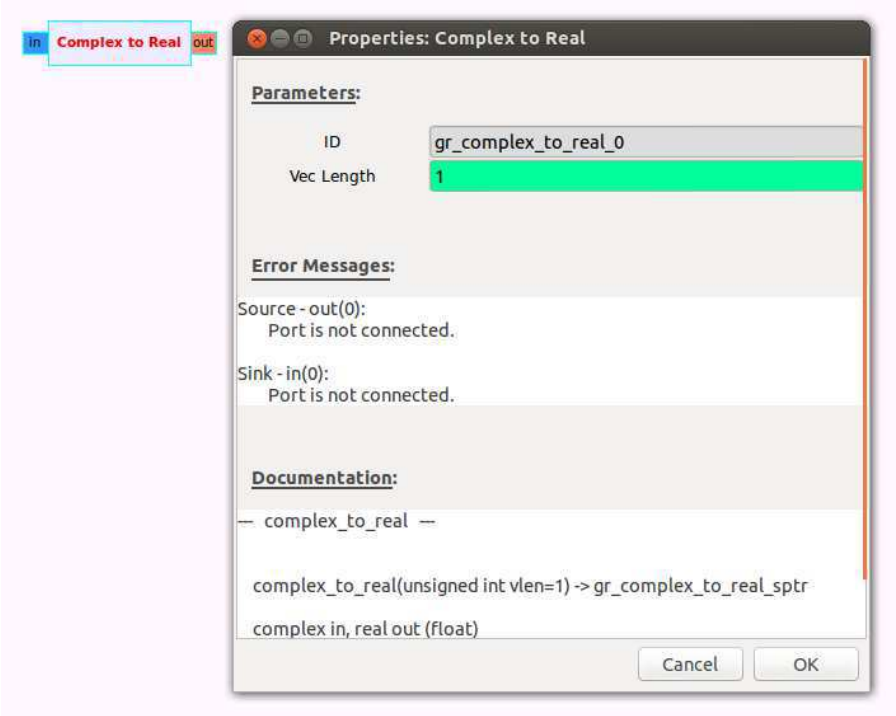

Figura 189. Bloque Complex To Real y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta un parámetro más aparte del *ID* el cual es:

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

11.1.12.5. Float To Int

Este bloque convierte del tipo de datos de punto flotante a un tipo de datos entero (32 bits). En la Figura 190 se muestra este bloque y sus propiedades.

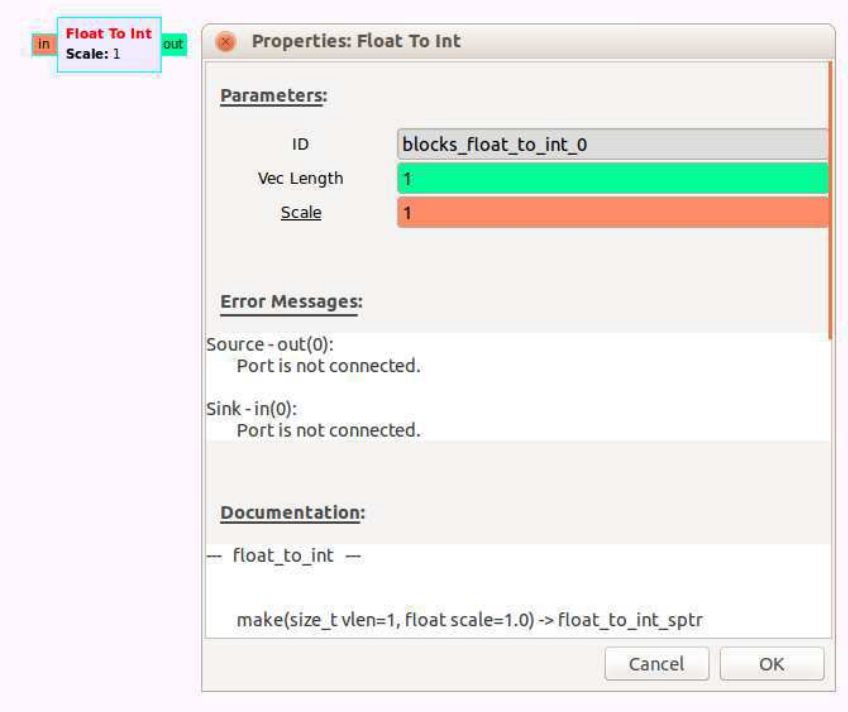

Figura 190. Bloque Float To Int y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta dos parámetros más aparte del *ID* los cuales son:

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

Scale: este parámetro es de tipo real y establece el factor de escala para la conversión.

11.1.12.6. Float To Short

Este bloque convierte un valor de punto flotante (real) en un entero corto de 16 bits. En la Figura 191 se observa el bloque *Float To Short* y sus propiedades.

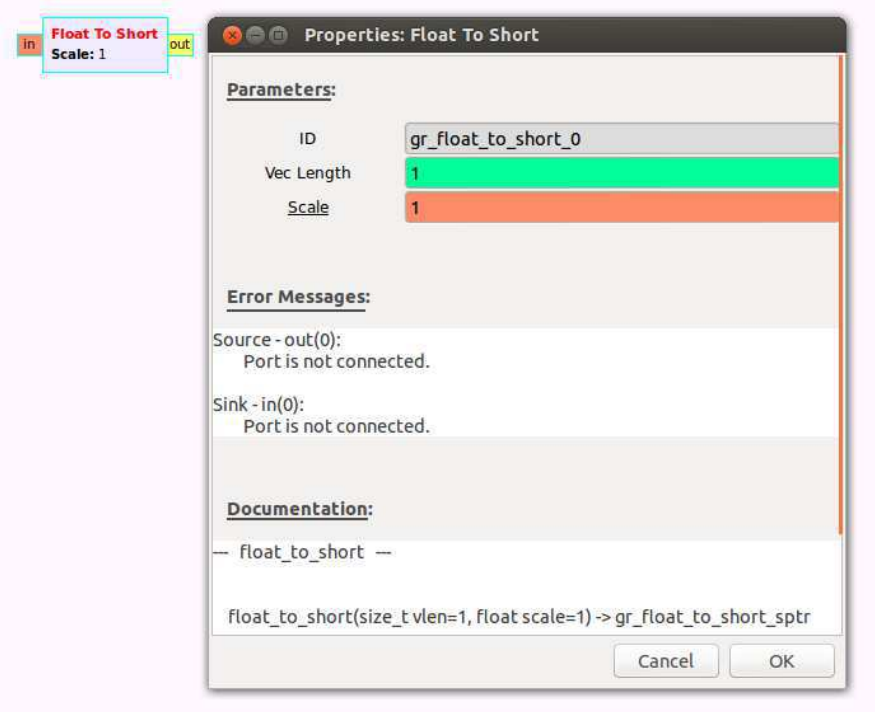

Figura 191. Bloque Float To Short y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta dos parámetros más aparte del *ID* los cuales son:

Vec Length: es de tipo Int y especifica la longitud del vector para el procesamiento del vector. Las aplicaciones típicas utilizarán el valor predeterminado de 1.

Scale: este parámetro es de tipo real y establece el factor de escala para la conversión. Por ejemplo: para normalizar una onda sinusoidal con amplitud de 1.0 a la gama completa de entero corto, utilice un factor de escala de 32768. Los valores fuera del rango [-32768, 32767] saturarán en lugar de envolver.

Para resumir, básicamente los bloques de este módulo "***Type Converters"*** lo que hacen es convertir de un tipo de datos a otro como se puede evidenciar en la explicación de los bloques anteriores. A este módulo pertenecen más bloques, como por ejemplo: Complex to Mag,

Complex to Mag^2, Complex to Arg, Float To Char, etc., los cuales funcionan bajo las mismas características de los bloques explicados.

11.1.13. Filters

11.1.13.1. Low Pass Filter (Filtro de paso bajo)

Este bloque es un filtro pasa bajo y es una conveniente envoltura para un filtro FIR y una función de generación de grifos firdes. La frecuencia de muestreo, la frecuencia de corte y el ancho de transición están en Hertz. El parámetro beta solo se aplica a la ventana de Kaiser. En la Figura 192 se observa el bloque *Low Pass Filter* y sus propiedades.

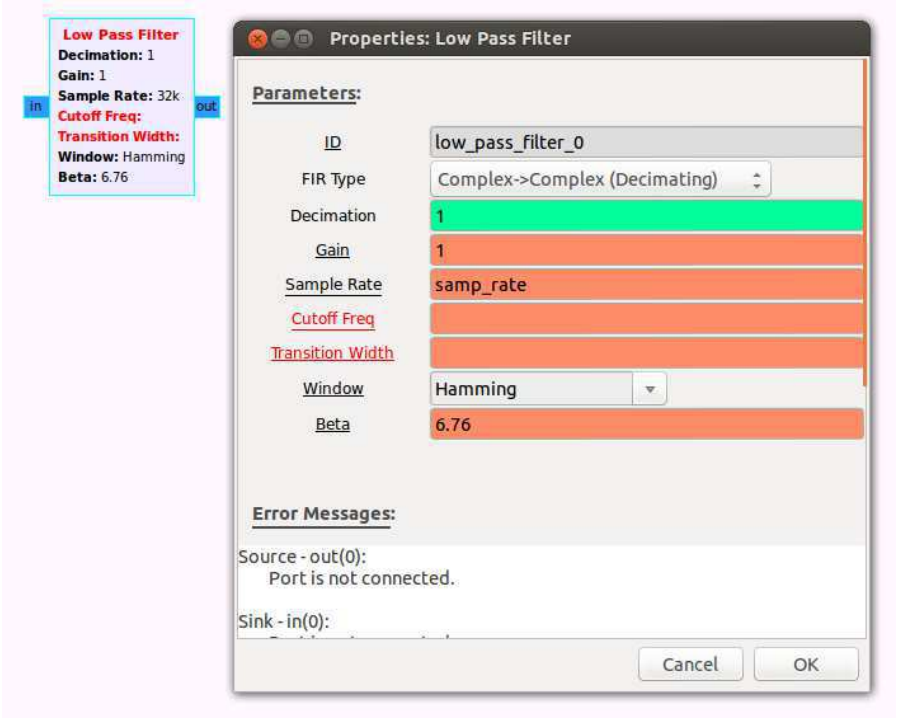

Figura 192. Bloque Low Pass Filter y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* ocho parámetros más en sus propiedades, los cuales son:

FIR Type: especifica el tipo de datos de las secuencias de entrada y salida, con la opción de diezmado o interpolación. En la Figura 193 se especifican estos tipos de datos donde se encuentran: *Complex->Complex* y *Float->Float.*

Complejo-> Complejo (diezmando)	Las corrientes de entrada y salida son complejas, con la opción de diezmar la salida.
Complejo-> Complejo (interpolación)	Las secuencias de entrada y salida son complejas, con la opción de interpolar la salida.
Flotador-> Flotador (diezmado)	Las corrientes de entrada y salida son reales, con la opción de diezmar la salida.
Flotador-> Flotador (Interpolación)	Las corrientes de entrada y salida son reales, con la opción de interpolar la salida.

Figura 193. Tipos de datos para FIR Type del bloque Low Pass Filter. (DocBook, 2013)

Decimation/Interpolation: este parámetro es de tipo Int y básicamente la decimation o interpolation se puede seleccionar a través del parámetro FIR Type. Si no es necesario volver a muestrear, ajuste este parámetro a 1.

Gain: es de tipo real y establece la ganancia del filtro.

Sample Rate: este parámetro es de tipo real y establece la frecuencia de muestreo del filtro, en Hz.

Cutoff Freq: de tipo real y establece la frecuencia de corte del filtro, en Hz.

Transition Width: de tipo real y establece el ancho de transición entre la banda de paso y la banda de parada. Un pequeño ancho de transición aumentará la longitud del filtro FIR.

Window: este parámetro especifica la función de ventana que se aplicará al filtro FIR, las funciones de ventana a elegir son: *Hamming, Hann, Blackman, Rectangular* y *Kaiser.*

Beta: es de tipo real y es un parámetro exclusivo para la ventana de Kaiser.

Los filtros: *High Pass Filter (Filtro de paso Alto), Band Pass Filter (Filtro de paso de banda)* y *Band Reject Filter (Filtro de rechazo de banda),* trabajan bajo las mismas características del *Low Pass Filter* (ver numeral 7.2.12.1) y contienen los mismos parámetros en sus propiedades.

11.1.13.2. DC Blocker (Bloqueador de DC)

Este bloque bloquea el componente DC de una señal. Esto puede ser útil cuando se trabaja con señales AM, ya que tomar la magnitud siempre será positivo e introducirá un sesgo DC en la señal. En la Figura 194 se observa este bloque con sus propiedades.

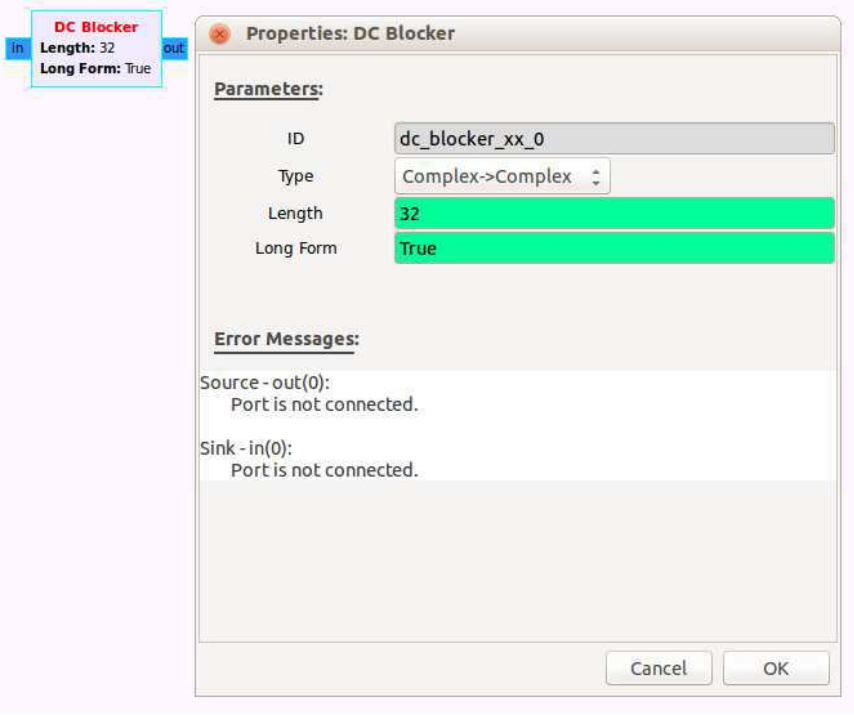

Figura 194. Bloque DC Blocker y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* tres parámetros más en sus propiedades, los cuales son:

Type: establece el tipo de datos del flujo de entrada y salida. En la Figura 195 se especifican estos tipos de datos: *Complex->Complex* y *Float->Float.*

Complejo-> Complejo	Las corrientes de entrada y salida son complejas.
Flotador-> Flotador	Las corrientes de entrada y salida son reales.

Figura 195. Tipos de datos para Type del bloque DC Blocker. (DocBook, 2013)

Length: de tipo Int y especifica la longitud de la línea de retraso utilizada para determinar el nivel de DC. Solo aplicable a la forma larga.

Long Form: este parámetro es de tipo bool y establece True = usa el bloqueador de DC de forma larga, con línea de retraso. False = usa el formulario corto (más rápido).

11.1.13.3. Hilbert

Este bloque realiza la transformación de Hilbert en la señal real entrante. La verdadera transformada de Hilbert es un filtro no causal. Esta versión discreta de la transformada de Hilbert funciona truncando el filtro a la cantidad especificada de tomas e introduciendo un retraso en la señal. En la Figura 196 se observa el bloque *Hilbert* y sus propiedades.

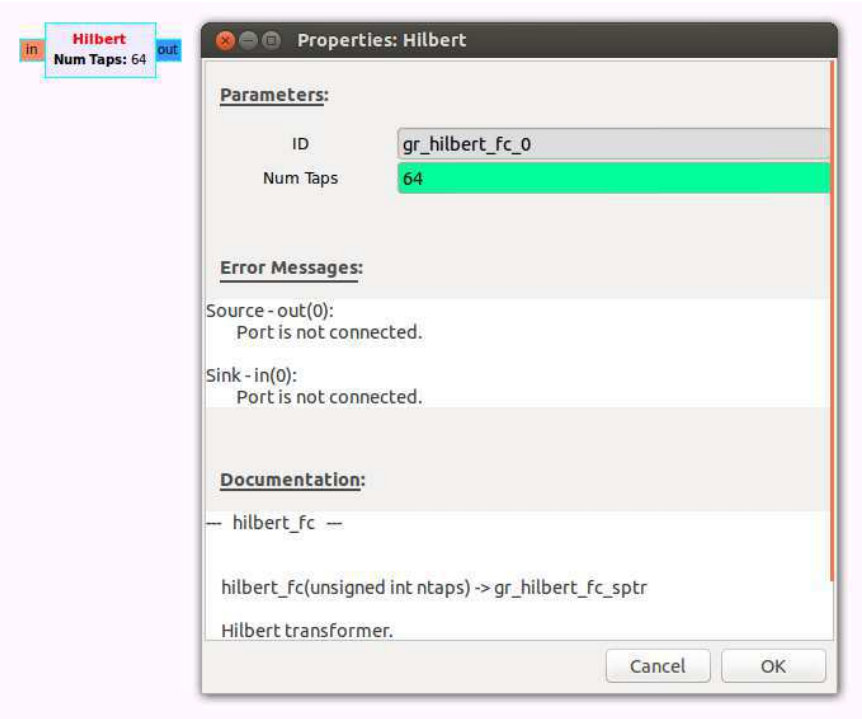

Figura 196. Bloque Hilbert y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta un parámetro más aparte del *ID* el cual es:

Num Taps: este parámetro es de tipo Int específica el número de toques para truncar el filtro de transformación.

11.1.13.4. Rational Resampler (Remuestrador Racional)

Este bloque es de *Interpolator* y *Decimator* combinados. Se usa para convertir de una frecuencia de muestreo a otra, siempre que puedan relacionarse por una relación: Fs_out = Fs_in x Interpolation / Decimation. Tenga en cuenta que todos los bloques que siguen este bloque en el gráfico de flujo deben esperar la tasa de muestreo de salida. En la Figura 197 se observa este bloque con sus respectivas propiedades.

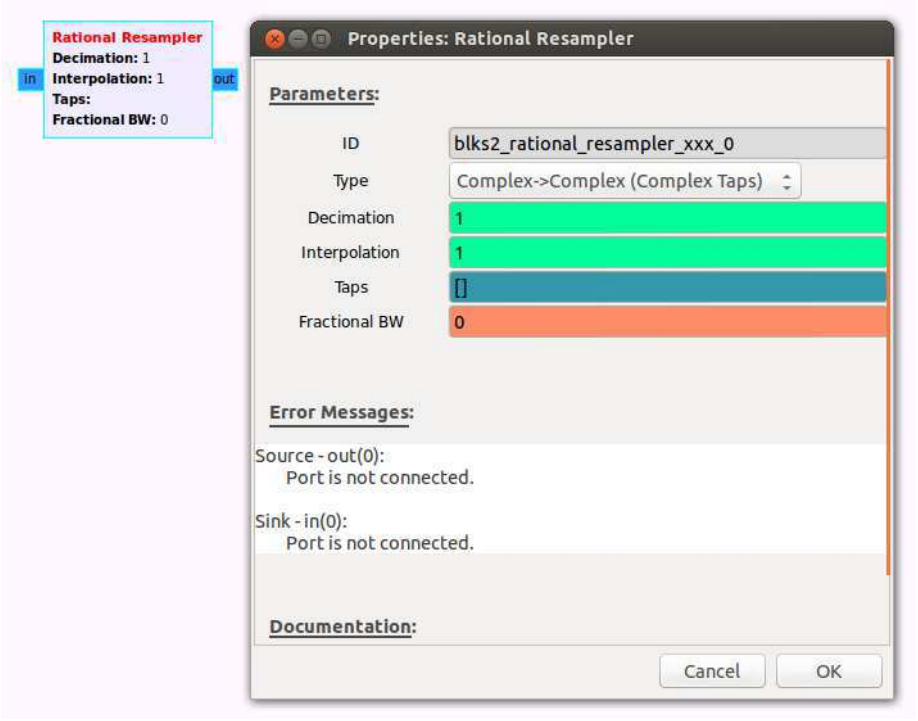

Figura 197. Bloque Rational Resampler y sus propiedades. (Sotfware GRC, 2017)

Como se puede observar este bloque presenta aparte del *ID* cinco parámetros más en sus propiedades, los cuales son:

Type: especifica el tipo de datos de las secuencias de entrada y salida. Para flujos complejos también especifica el tipo de filtro que se puede aplicar. En la Figura 198 se especifican estos datos que incluyen: *Complex->Complex* y *Float->Float.*

Complejo-> Complejo (Grifos complejos)	Las corrientes de entrada y salida son complejas. Además, el filtro antialiasing puede tener toques complejos.
Complejo-> Complejo (Real Taps)	Las corrientes de entrada y salida son complejas. El filtro antialiasing tiene toques reales.
Float-> Float (Real Taps)	Las corrientes de entrada y salida son reales.

Figura 198. Tipos de datos para Type del bloque Rational Resampler. (DocBook, 2013)

Decimation: este parámetro es de tipo Int y especifica el valor de ejecución.

Interpolation: de tipo Int y especifica el valor de interpolación.

Taps: su tipo es el mismo que se establece en el parámetro *Type.* Este se debe dejar vacío para que tome el valor automático. De lo contrario, se usa una expresión firdes para especificar el filtro que se utilizará.

Fractional BW: de tipo real y se establece a 0 para el valor automático.

11.1.13.5. Frequency Xlating FIR Filter (Filtro FIR de frecuencia Xlating)

Este bloque implementa un filtro de FIR de conversión de frecuencia. Esto a menudo se utiliza como un bloque de selección de canal de utilidad, ya que realiza una traducción de frecuencia, selección de canales y diezmado en un solo paso.

Este bloque presenta cinco parámetros aparte de *ID* en sus propiedades, los cuales son:

Type: especifica el tipo de flujo de entrada y salida. En la Figura 199 se especifica estos tipos de datos: *Complex->Complex, Float->Complex* y *Short Complex.*

Complejo-> Complejo (Grifos complejos)	Las corrientes de entrada y salida son complejas. Además, los toques de filtro pueden ser complejos.
Complejo-> Complejo (Real Taps)	Las corrientes de entrada y salida son complejas. Además, los toques de filtro deben ser reales.
Flotador-> Complejo (Grifos complejos)	El flujo de entrada es real mientras que el flujo de salida es complejo. Además, los toques de filtro pueden ser complejos.
Flotador-> Complejo (Grifería Real)	El flujo de entrada es real mientras que el flujo de salida es complejo. Además, los toques de filtro deben ser reales.
Short-> Complex (Complex Taps)	La secuencia de entrada es un entero corto de 16 bits mientras que la secuencia de salida es compleja. Además, los toques de filtro pueden ser complejos.
Short-> Complex (Real Taps)	La secuencia de entrada es un entero corto de 16 bits mientras que la secuencia de salida es compleja. Además, los toques de filtro deben ser reales.

Figura 199. Tipos de datos para Type del bloque Frequency Xlating FIR Filter. (DocBook, 2013)

Decimation: de tipo Int y especifica el factor de ejecución para la salida. La velocidad de muestreo de salida será la velocidad de muestreo especificada dividida por el factor de ejecución.

Taps: maneja el mismo tipo de datos establecido en el parámetro *Type*. Se ingresa una expresión de firdes válida o una lista de Python válida que contenga los toques (si diseña filtros usando otro método).

Center Frequency: de tipo real y específica la frecuencia central. Esta es la frecuencia que se desplazará a 0 Hz antes de que se aplique el filtro de selección de canal.

Sample Rate: este parámetro es de tipo real y especifica la frecuencia de muestreo, en Hz.

Los módulos y sus bloques explicados en esta sección son los que generalmente se usan con más frecuencia para los diferentes proyectos ejecutados con GRC, cabe aclarar que estos módulos no son la totalidad que se encuentran en el software, hay más módulos y más bloques los cuales se deberán usar en algunos proyectos eventualmente, sin embargo con los que se han explicado en este apartado dan una base consistente para iniciar en el mundo de GRC.