ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORES: TORRES VEGA, MARCO DAVID

VACA GALLARDO, CRISTIAN SANTIAGO

TEMA: SISTEMA EMBEBIDO DE COMUNICACIÓN DIGITAL DE VOZ UTILIZANDO PERIFÉRICOS DE RADIO UNIVERSAL USRP-E110.

DIRECTOR: ING. DANIEL ALTAMIRANO

CODIRECTOR: ING. PAUL BERNAL

SANGOLQUÍ, 16 JULIO 2014

Declaración de Responsabilidad

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

MARCO DAVID TORRES VEGA Y
CRISTIAN SANTIAGO VACA GALLARDO

DECLARAMOS QUE:

El proyecto de grado denominado "SISTEMA EMBEBIDO DE COMUNICACIÓN DIGITAL DE VOZ UTILIZANDO PERIFÉRICOS DE RADIO UNIVERSAL USRP-E110.", ha sido desarrollado en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie, de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 16 de Julio del 2014.

_____ _____

Marco David Torres Vega Cristian Santiago Vaca Gallardo

Autorización de publicación

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

MARCO DAVID TORRES VEGA Y
CRISTIAN SANTIAGO VACA GALLARDO

Autorizamos a la Universidad de las Fuerzas Armadas - ESPE la publicación, en la biblioteca virtual de la Institución del trabajo "SISTEMA EMBEBIDO DE COMUNICACIÓN DIGITAL DE VOZ UTILIZANDO PERIFÉRICOS DE RADIO UNIVERSAL USRP-E110.", cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría

Sangolquí, 16 de Julio del 2014.

_____ _____

Marco David Torres Vega Cristian Santiago Vaca Gallardo

Certificado de tutoría

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICADO

Ing. Daniel Altamirano

Ing. Paul Bernal

CERTIFICAN

Que el trabajo titulado "SISTEMA EMBEBIDO DE COMUNICACIÓN DIGITAL DE VOZ UTILIZANDO PERIFÉRICOS DE RADIO UNIVERSAL USRP-E110.", realizado por Marco David Torres Vega y Cristian Santiago Vaca Gallardo, ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la Universidad de las Fuerzas Armadas - ESPE en su reglamento.

Debido a que se trata de un trabajo de investigación recomiendan su publicación.

Sangolquí, 16 de Julio del 2014.

Ing. Daniel Altamirano	Ing. Paul Bernal
DIRECTOR	CODIRECTOR

DEDICATORIA

Dedico este trabajo a las personas más importantes en mi vida, mis padres Flor y Marco, que siempre estuvieron listos para brindarme incondicionalmente su ayuda. Dedico esta tesis también a mi segunda madre quien ha estado conmigo en todo este camino hasta llegar al día de hoy, mamita Maruja. Ahora es mi turno de devolver un poco de todo lo inmenso que me han otorgado.

Marco D. Torres Vega

A mi abuelo Joaquín, donde te encuentres sé que siempre estuviste junto a mí, brindándome fuerzas, dándome ejemplos dignos de superación y entrega, a mis padres, porque creyeron en mí y porque me sacaron adelante, porque gracias a ustedes, hoy puedo ver alcanzada mi meta, ya que siempre estuvieron impulsándome en los momentos más difíciles de mi carrera, y porque el orgullo que sienten por mí, fue lo que me hizo ir hasta el final. Va por ustedes, por lo que valen, porque admiro su fortaleza y por lo que han hecho de mí.

Cristian S. Vaca Gallardo

AGRADECIMIENTO

A mis padres Flor y Marco, y también a mi abuelita. Millones de gracias por haberme dedicado su tiempo, esfuerzo y confianza para poder lograr mis sueños a pesar de lo distantes que se veían, gracias por su motivación y por darme la mano para caminar conmigo cuando sentía que el camino se terminaba, espero no defraudarlos y continuar cosechando triunfos junto a ustedes.

A mis maestros, quienes con sus lecciones y experiencias me enseñaron que en un aula no solamente se aprende lo escrito en un libro, sino también se aprende prepararse para los retos que impone la vida. Gracias Ing. Altamirano por brindarnos la oportunidad de trabajar con Ud. en este proyecto, y sobretodo muchas gracias por su paciencia y tiempo.

Finalmente, a todos mis amigos, con quienes compartí alegrías, tristezas y malas noches. Con quienes en conjunto estuvimos para brindarnos apoyo durante todo el tiempo que pasamos dentro y fuera de estas aulas, gracias en verdad.

Marco D. Torres Vega

A ustedes amigos por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida. Mil palabras no bastarían para agradecerles su apoyo, su comprensión y sus consejos en los momentos difíciles.

A todos, espero no defraudarlos y contar siempre con su valioso apoyo, sincero e incondicional.

Cristian S. Vaca Gallardo

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto contempla el diseño e implementación de un transmisor y receptor de voz en tiempo real con modulación *GMSK* sobre Dispositivos Embebidos de Radio Universal *USRP E110* y *GNU Radio* como herramienta de *Software*. En la primera fase los diseños tanto del transmisor como del receptor fueron desarrollados en un computador con herramientas de programación en bloques proporcionados por el *GNU Radio* utilizando *Ubuntu* como sistema operativo. Se realizó la simulación del canal inalámbrico para evaluar el comportamiento del sistema; verificado esto, se inició el procedimiento de traducción de los diagramas de bloques diseñados para embeberlos dentro de los dispositivos *USRP*. Se realizaron experimentos para el análisis de la eficiencia del sistema implementado en cuanto a parámetros de ancho de banda, potencia de transmisión y recepción, constelación, distancia efectiva entre dispositivos, etc. También se evaluó parámetros subjetivos en cuando a la calidad del audio en la transmisión, basado en encuestas de *Mean Opinion Score* y *tests* de intelegibilidad segmental. El sistema está en capacidad de transmitir y recibir señales de audio con claridad en la transmisión hasta distancias efectivas de hasta 8 metros en un ambiente de laboratorio.

Palabras clave: GNU Radio, USRP E110, SDR, Radio Definido por Software, Sistema de Comunicaciones, Modulación GMSK.

ABSTRACT

This Project contains the design and implementation of a real-time voice transmitter and receiver with GMSK modulation over Universal Radio Embedded Devices (USRP E110) and GNU radio as software tool. The first part will show the designs of the transmitter and the receiver as well, both stages were developed with programming tools provided by GNU Radio software using a PC with Ubuntu as operative system. Once both stages were designed, a channel simulation was done to evaluate the system performance. After this, the process of translating the flow graphs began, so they can be embedded inside the USRP devices. Experiments to analyze the success of the implemented system were also performed, regarding the facts such as bandwidth parameters, transmission and reception power, constellation, the effective distance between devices, etc. Subjective parameters were proved as well, regarding the audio quality in the transmission, based on Mean Opinion Score surveys and segmental intelligibility tests. As a result of this project, the system is successfully capable of transmitting and receiving clear audio signals up to a distance of 8 meters in a laboratory environment.

Key words: GNU Radio, USRP E110, SDR, Software Defined Radio, Communications System, GMSK Modulation.

CAPITULO 1

INTRODUCCIÓN

1.1 Antecedentes

El siglo XX ha sido en general la época en la cual las aplicaciones de Radio Basadas en *hardware HDR* han tenido su auge. Dichas aplicaciones se basaban en amplificadores de baja frecuencia, etapas de frecuencia intermedia, sintonizadores, detectores etc., todos estructuras de *hardware* con poco o nada de *software* de control, y como resultado de ello se tenía equipos de grandes dimensiones, con alto costo de fabricación y corto tiempo de vida puesto que no era posible su modificación para adaptarse a un nuevo estándar de comunicaciones.

La manera en que se percibe un equipo de radio cambia radicalmente cuando Joe Mitola, a mediados de los años 90 inicia investigaciones para elaborar sistemas que no tuviesen dependencia exclusiva del *hardware* y este pueda ser reemplazado por equipos de *software* (Cognitive Radio, 2000).

A inicios del 2000, Gerard Youngblood, un radioaficionado estadounidense propone un nuevo concepto de sistemas de radiocomunicaciones (Gerald Youngblood, 2002), los cuales son desarrolados mediante programas y pasan a ser conocidos como

SDR Software Defined Radio y se definen como: *"Un radio en el cual alguna o todas las funciones de la capa física son definidas por software"* (Forum, 2007).

En general, un dispositivo *SDR* básico se encuentra formado por una computadora equipada con una tarjeta de audio u otro conversor analógico digital precedido de algún adaptador de Radio Frecuencia *RF*. En su mayoría un *SDR* utiliza dispositivos digitales programables para mejorar el procesamiento de señal necesario para transmitir y recibir información. La tecnología *SDR* se vale de los Procesadores Digitales de Señales *DSP* y/o Arreglos de Compuertas de Campo Programables *FPGA* para realizar el procesamiento de señales haciendo que se pueda reestructurar un sistema de comunicaciones simplemente cambiando el *software* programable.

En Ecuador, el hablar de Radio Definido por *software* es un tema relativamente nuevo, poco se ha avanzado al respecto sobre esta tecnología y su desarrollo. Algunas tesis han sido presentadas sobre el tema en el año 2010 y 2011 por parte de estudiantes de la Escuela Politécnica del Ejército quienes han desarrollado temas como: *"Emulador de un sistema de comunicaciones utilizando tecnología SDR"* (Quiroz, 2010) e implementaciones en *software* propietario (*Matlab*) usando tecnología de Radio Definido por *Software* (Angulo Hugo, 2011).

Los ejemplos que ilustran la adopción y transición hacia la tecnología *SDR* son abundantes, entre ellos podemos recalcar que más del 93% de la infraestructura en el mercado de las comunicaciones móviles en Estados Unidos utiliza tecnología *SDR* (Forum, 2009). Y el crecimiento a futuro para soportar voz, video y datos móviles demandará el establecimiento de más estaciones base *SDR*, por otro lado, cerca de un billón de Radios Definidos por *Software* fueron utilizados en aplicaciones de terminales móviles, hoy en día casi todos los dispositivos de radio usados para comunicaciones militares utilizan tecnología *SDR* debido a su flexibilidad, también los satélites comerciales han hecho uso de esta tecnología para el procesamiento de señales en banda base

y frecuencia intermedia *IF* (Lee, 2004).

Actualmente el principal reto para un equipo *SDR* es igualar en eficiencia a un sistema *HDR*. Este parámetro de eficiencia puede ser medido mediante el costo por bit de información, potencia consumida por bit de información, y el volumen físico consumido por bit de información, finalmente basado en las afirmaciones anteriores, se puede afirmar que la tecnología *SDR* permite equipos flexibles con un tiempo de vida más largo, un menor costo de fabricación y sin el desperdicio de componentes electrónicos cuando se realicen actualizaciones del sistema.

1.2 Justificación e Importancia

Los sistemas tradicionales basados en aparatos de radio frecuencia se ven limitados en cuanto a su capacidad de modificación principalmente por el hecho de que sus procesos se encuentran definidos en su mayoría en la capa física. Como resultado de esto, los costos de producción de estos dispositivos son altos y poseen una flexibilidad mínima en cuanto a soportar múltiples estándares. Por otro lado, la tecnología de Radio Definido por *Software* provee una solución eficiente y de bajo costo a este problema, permitiendo el funcionamiento multimodo, multibanda y/o el establecimiento de dispositivos multifuncionales que utilicen tecnología inalámbrica, los cuales pueden ser mejorados utilizando únicamente actualizaciones por *software*.

El presente proyecto se enfoca en el desarrollo e implementación de un sistema embebido de comunicación digital de voz. El dispositivo dentro del cual se encontrará el sistema es el *USRP* Modelo E110 perteneciente a la compañía *Ettus Research*.

La tecnología *SDR* ha logrado que el uso de tecnologías de *RF* que respalden la operación de *SDR* sobre un rango espectral amplio empiece a crecer y madurar, y por primera vez el uso de *SDR* se ha visto como una tecnología que permita el

acceso dinámico al espectro con sistemas de radio de funcionalidades inteligentes o cognitivos, lo que a futuro alcanzará el establecimiento de una tecnología que permita al usuario final el acceso a comunicaciones inalámbricas de forma ubicua, permitiendo que éste se comunique con quien desee, cuando lo desee y de cualquier manera que sea apropiada.

Por ello, el desarrollo de un sistema de comunicación de voz digital que aporte características de flexibilidad y bajo costo dentro de un dispositivo de reducidas dimensiones resulta bastante beneficioso tanto para el establecimiento de sistemas de comunicaciones comerciales, militares y comunitarios, como la apertura de nuevos campos de investigación en el área de comunicaciones, las tecnologías inalámbricas y el desarrollo de *software* embebido.

Por otro lado, Ecuador no cuenta con las herramientas ni el capital necesario para fabricar dispositivos de Radio Definido por *hardware*, y adquirirlos de países extranjeros resulta un proceso demasiado costoso. Sin embargo, las características de la tecnología *SDR* permiten embeber y personalizar todo un sistema de comunicaciones dentro de un solo dispositivo, ofreciendo una alternativa de bajo costo y sobretodo flexible para el desarrollo de aplicaciones de comunicación.

Finalmente, debido a los puntos mencionados anteriormente ha quedado claro que Ecuador se queda corto en cuanto al tema de investigación y desarrollo de tecnologías *SDR*, y, a pesar de haberse desarrollado proyectos alrededor de éste tema, se requiere una investigación más detallada y profunda que permita el desarrollo de sistemas de comunicaciones hechos en el país.

Los temas desarrollados hasta ahora en Ecuador se han valido de herramientas de *software* propietario y plataformas *SDR* que requieren una interfaz a un PC. Los desarrolladores del presente proyecto apuntan al uso de *software* libre para el desarrollo

de un sistema de comunicación digital de voz que opere independientemente dentro de un sistema embebido. Éstos son algunos de los motivos por el cual este tipo de investigaciones resulta útil para colocar al país en el mismo nivel de conocimientos que el resto de naciones desarrolladas.

1.3 Alcance del Proyecto

El presente proyecto entregará como producto final un sistema de comunicación digital de voz con tecnología *SDR*, el cual se implementará mediante el uso de Dispositivos Periféricos de Radio Universal USRP - *(Universal Software Radio Peripherical)* Modelo *E110* desarrollado bajo plataformas de *software* libre con distribuciones de *Linux*. La tecnología *SDR* permite operar dentro de cualquier frecuencia, sin embargo, el proyecto será desarrollado en concordancia con las bandas de frecuencia de las tarjetas *Daughterboard* y antenas que se encuentren disponibles en el laboratorio. El resultado será un sistema *SDR* que permita una comunicación confiable entre transmisor y receptor.

El *software* del sistema de comunicaciones se encontrará implementado bajo la distribución *Angstrom* de *Linux* y desarrollado con la herramienta gratuita de código abierto *GNU Radio*, que usan bloques de procesamiento de señales y lenguaje de programación *Python* permitiendo implementar sistemas en tiempo real, de alto rendimiento, todo dentro de una interfaz amigable.

Por otro lado, la modulación, la codificación de canal y la codificación de fuente a implementarse están definidos en los bloques de procesamiento de señales que se encuentren disponibles dentro de la herramienta *GNU Radio*, y en base a los cuales se desarrollará el sistema, un esquema básico del sistema propuesto se muestra en la Figura 1.

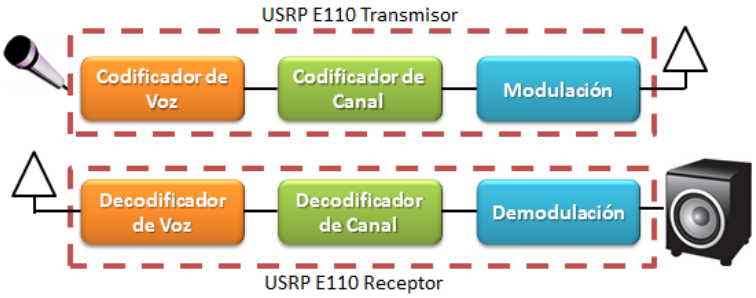


Figura 1: Esquema de Modulación.

El nexo entre la herramienta de programación *GNURadio* (Radio, 2014) y las instrucciones que serán implementadas dentro del dispositivo *USRP E110* es el lenguaje *Python* (Python, 2014), el cual permitirá crear los archivos que serán embebidos dentro del dispositivo *SDR*. Para la traducción desde el lenguaje de alto nivel *GNURadio* y el lenguaje de programación reconocible dentro de la *FPGA* del *USRP E110* se requiere la implementación de ciertas librerías y algoritmos de procesamiento que permitan reconocer las instrucciones en un lenguaje de bajo nivel para su posterior embebido.

1.4 Objetivos

1.4.1 General

- Diseñar e implementar un sistema de comunicación digital de voz usando tecnología SDR.

1.4.2 Específicos

- Investigar acerca del desarrollo de aplicaciones sobre la plataforma GNU Radio.
- Establecer la estructura del sistema de comunicación mediante bloques disponibles en GNU Radio.

- Traducir el diagrama de bloques del sistema de comunicación desarrollado para embeberlo en el USRP.
- Analizar la eficiencia del sistema implementado en cuanto a parámetros de ancho de banda y potencia de transmisión.
- Evaluar el sistema implementado mediante parámetros subjetivos para determinar la percepción del usuario sobre el rendimiento del sistema.

CAPITULO 2

RADIO DEFINIDO POR SOFTWARE - SDR

2.1 Estado del arte

En la actualidad la comunicación en la sociedad se ha hecho cada vez más dependiente de medios inalámbricos que ofrezcan un intercambio de información eficiente, confiable, ecológica y barata. Los dispositivos actuales se encuentran migrando desde una tecnología implementada en *hardware* al diseño de procesamiento por *software*, el cual es realizado por transmisores y receptores digitales, y son capaces de ejecutar una variedad de operaciones en banda base, como por ejemplo modulación, codificación de fuente y canal, *FEC* y ecualización. Aunque en sus inicios estos transmisores y receptores digitales se encontraban implementados usando circuitos integrados y otras maneras de electrónica no programable, el advenimiento de mejores y mayores funcionalidades programables en banda base para estos dispositivos ha desembocado en mejoras en tecnología de microprocesadores a lo largo de los últimos años. Como consecuencia, existe ahora un nuevo campo de investigación y desarrollo, el cual se denomina Radio Definido por Software *SDR*.

La principal característica de un dispositivo con tecnología *SDR* es que constituye una clase de radios reconfigurables/reprogramables cuyas funcionalidades de la capa física pueden ser significativamente modificadas mediante cambios en *software* para permitir a un *SDR* la capacidad de implementación de distintas funciones de procesamiento en distintos momentos de tiempo pero en la misma plataforma, también define en *software* varias características de radio en banda base y posee cierto nivel de control en *software* en las operaciones de *Front-End*.

El fundador de *GNU Radio* Eric Blossom ha definido a la tecnología *SDR* como:

"Software radio es la técnica de obtención de código lo más cerca a la antena como sea posible. Esto transforma los problemas de hardware en problemas de software" (Blossom, 2009)

Al referirse a *"lo más cerca a la antena"* Blossom desea interpretar la finalidad principal de las plataformas *SDR*: La implementación de la capa física de un sistema de radio mediante algoritmos *software*, de manera que el diseño e implementación de sistemas de radio dependa cada vez más de la programación *software* antes que de circuitos físicos. En capítulos siguientes se describirá con mayor detalle la arquitectura y funcionamiento de una plataforma *SDR*.

Uno de los primeros proyectos conocidos de implementación del concepto de *SDR* fue en el proyecto militar estadounidense *SpeakEasy* en el año 1991 (R. I. Lackey, 1995), cuyo principal objetivo era la implementación de mas de diez tipos de tecnologías de comunicaciones inalámbricas (usadas en ese momento por EE.UU) en un equipo programable, el cual debería operar en una banda de frecuencias desde los 2MHz hasta los 200MHz y con la capacidad de actualizar su código para los estándares futuros. No fue sino hasta 1995 que se hizo posible la consecución de los objetivos planteados, dejando sentado un precedente y abriendo un nuevo campo a la investigación científica, sin embargo el proyecto entregado solamente era capaz de mantener

una comunicación *Half Duplex*, por lo que se modificaron los alcances del proyecto y lo enfocaron hacia nuevos aspectos como disminución de peso y costo, incremento en la capacidad de procesamiento y simultaneidad de comunicaciones, arquitectura de software libre, etc.

SpeakEasy fue el primer proyecto conocido que trabajó con *FPGA* para procesamiento de datos digitales radiados, pero actualmente *SDR* es una tecnología sobre la cual se pueden encontrar distintas alternativas para su implementación, como por ejemplo: implementaciones de *SDR* usando *DSP* (Schiphorst, 2000), la implementación de *SDR* para radio aficionados usando la tarjeta de sonido de un PC (Youngblood, 2003), kits de desarrollo como *SoftRock-40* que trabaja con el software libre *PowerSDR* desarrollado por la empresa *FlexRadio* y proyectos como el de *VirginaTech* que consistió en la programación de Radios Software con características reconfigurables (Tranter, 1998).

Los esfuerzos en el desarrollo e investigación de proyectos relacionados con la tecnología *SDR* han desembocado en la creación del *SDR Forum* compuesto por mas de cien empresas, instituciones y organizaciones (*Altera, Xilinx, NASA, Virginia Tech, Toshiba, Samsung, Lockheed, Motorola, QUALCOMM, Hitachi, Ohio Aerospace Institute*, entre otras), cuyo objetivo es promocionar el uso de *SDR* desarrollando estándares y especificaciones del mismo, divulgando la tecnología y sus capacidades para el soporte de necesidades militares, civiles y comerciales.

Desde el punto de vista técnico, *SDR* constituye una alternativa importante para la integración y convergencia de tecnologías inalámbricas, y de hecho durante los últimos años ha sido estudiado y desarrollado con este objetivo, pero su implementación comercial aún enfrenta varios retos de tipo económico, pues los costos que supone la adquisición de un dispositivo que soporte esta tecnología son elevados. A pesar de esto esperamos que a corto plazo la tecnología brinde la oportunidad para que estos tipos

de sistemas se masifiquen y se establezcan estándares de operación para lograr un verdadero soporte a nivel de capa física para la convergencia e interoperabilidad entre dispositivos, para mediante la evolución de este concepto llegar a la implementación de radio cognitiva, este es el fin último de las plataformas SDR.

2.2 Equipos de Radio Convencionales

Los equipos *SDR* pueden ser repartidos en generaciones (EA1DDO, 2012), las cuales son:

- I Generación: Utiliza el método de fase *(Phasing method)*. Fueron las primeras pruebas a partir de los años 50 usando señales 'I' de fase y 'Q' de cuadratura. El mejor ejemplo de ello sería el transmisor *Central Electronics CE100/200*.

- II Generación: En esta generación estarían casi todos los dispositivos *SDR* que conocemos actualmente, aquellos que usan detectores de cuadratura y fase *(QSD Quadrature sampling Detector)*. El desencadenante de la popularidad de los SDR hoy en día, fueron dos hechos:

  - Por un lado, el proyecto liderado por Dan Tayloe, el N7VE, creó un mezclador o detector de alto rendimiento y muy sencillo electrónicamente, usando muy pocos componentes. Lo denominó *Zero IF Quadrature Product Detector*.

  - El segundo desencadenante fue Gerald AC5OG (ahora K5SDR) con la publicación de un artículo donde se describía el primer *SDR* de segunda generación para radioaficionados, equipado con el detector de Tayloe. Fue el famoso *Flex SDR-1000*.

  A partir de ese momento empezaron a aparecer más dispositivos *SDR* basados en ese modelo. En esta clase estarían desde el *SDR-1000* del año 2007, hasta

los *Flex 1500, 3000 y 5000*, pasando por los *SoftRock* y dispositivos similares, los cuales están en capacidad de usar las tarjetas de sonido del PC o poseen una propia tarjeta interna como en el caso de las *Flex*.

- III Generación: En esta generación se encuentran los que usan un chip digitalizador de RF directamente, llamado *DDC - Digital Down Conversión* para la parte Rx y *DUC Digital Up Conversion* en la sección de Tx. Ese DDC es un chip digitalizador específico. En esta categoría estaría el *HPSDR (High Performance Software Defined Radio)*, *Perseus*, *QS1R*, *Ettus*, y algunas otras empresas fabricantes.

Actualmente se cuenta con nuevos equipos de radio software, con mucha más capacidad y prestaciones que sus predecesores, la compañía *Ettus Research* es la encargada de la fabricación y comercialización de distintos modelos de dispositivos *SDR* con características específicas para distintas áreas de trabajo. Estos nuevos dispositivos poseen arquitecturas que permiten trabajar en conjunto con un computador mediante comunicación *USB Gigabit/Ethernet* como es el caso de los dispositivos *USRP N210* hasta tener todo un sistema embebido dentro del equipo que permita su funcionamiento independiente, con su propio procesador y sistema operativo como es el *USRP E110*, el cual es usado en el desarrollo del presente proyecto.

2.3 Tecnología SDR

El concepto de *SDR* ha ido evolucionando conforme el pasar de los años, pero principalmente los avances conseguidos obedecen esencialmente a la configuración básica mostrada en la Figura 2, la cual ilustra los tres bloques funcionales que conforman la arquitectura general de un dispositivo *SDR*: la sección de radiofrecuencia *RF*, la sección de frecuencia intermedia *IF* y la sección de Banda Base (Tech, 2002).

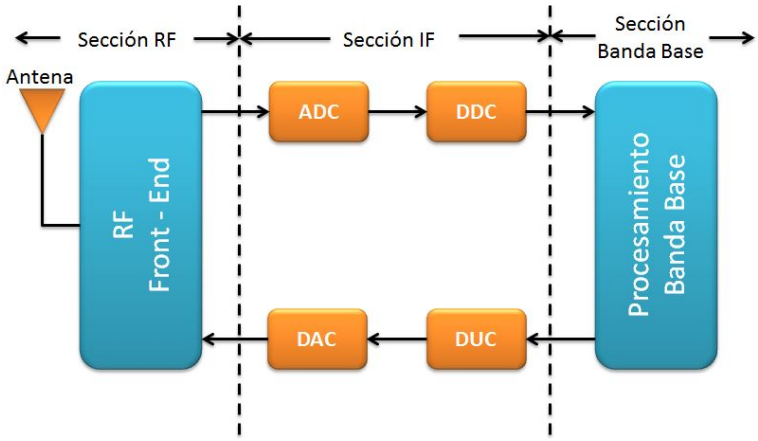


Figura 2: Diagrama de bloques funcionales de un SDR.

De los tres bloques, la sección de *RF* es la que casi siempre trabaja con *hardware* analógico mientras que las secciones de *IF* y Banda Base se implementan generalmente con módulos *hardware* digitales. La sección de *RF*, también llamada *Front-End* es la responsable de la transmisión y recepción de señales de radiofrecuencia para adecuarlas y convertirlas en frecuencias intermedias (en el caso de la recepción) o la amplificación y modulación de señales de *IF* para adecuarlas a la transmisión en el aire (en el caso de la transmisión).

La sección de *IF* es la encargada de pasar la señal de frecuencia intermedia a banda base y digitalizarla (en la recepción) o pasar la señal de banda base a *IF*. Los módulos *ADC/DAC* realizan la conversión digital/analógica de la señal. Los módulos *DDC/DUC* son los encargados de bajar digitalmente la señal de *IF* a Banda Base o subir de Banda Base a *IF* respectivamente.

La sección de Banda Base es la encargada de todo el procesamiento en Banda Base de la señal (tiempos de bit, ecualización, saltos en frecuencia, etc.). En esta configuración de *SDR* los módulos *DDC/DUC* y la sección de banda base son los que más millones de instrucciones por segundo requieren, motivo por el cual son implementados normalmente en *FPGA*.

Entonces, un *SDR* como tal es un dispositivo que realiza algunas tareas complejas de manera simultánea para permitir una transmisión y recepción homogénea de datos.

2.3.1 Arquitectura SDR

La estructura básica de un *SDR* con *GNU Radio* como plataforma *software* específica y *hardware USRP E110* se detalla en la Figura 3. La sección analógica es la encargada de realizar todas aquellas operaciones que no pueden ser efectuadas en el dominio digital, como lo son la alimentación a la antena, procesos de filtrado y combinación en *RF*, preamplificación, amplificación, y generación de la frecuencia de referencia

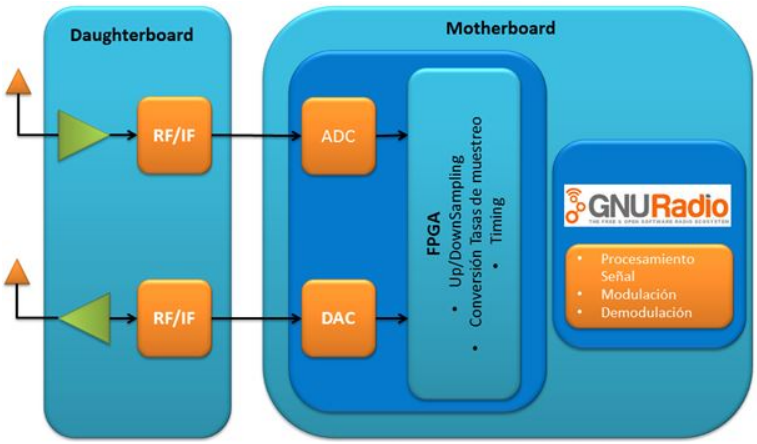


Figura 3: Arquitectura SDR con GNU Radio.

De la Figura 3 podrá notarse que la estructura de una plataforma *SDR* se divide prácticamente en dos bloques. El bloque de la izquierda corresponde al *Front-End* del *hardware* cuya representación física son las tarjetas hijas *Daughterboards*, las cuales sirven como la interfaz hacia el domino de *RF*. En el segundo bloque se implementa la sección que hará de interfaz entre los procesos digitales y analógicos del sistema, y también dentro de este bloque se encuentra el software de desarrollo *GNU Radio* donde se realiza el procesamiento completo de la señal (todo en *software*).

Para ser más precisos, se detallará el proceso básico de interacción entre bloques.

La interfaz hacia el mundo analógico está dada como ya se ha mencionado en el bloque de la izquierda de la Figura 3. Una señal de *RF* puede ser recibida o transmitida mediante antenas, o también puede ser conectada directamente vía cables *SMA* hacia los puertos del *Front-End* de *RF* en las placas hijas *(Daughterboards)*. El camino superior (la flecha que apunta hacia la placa hija) indica la vía de recepción Rx, mientras que el camino inferior describe la vía de transmisión Tx, Ambos caminos pueden operar de forma autónoma. En las tarjetas *Daughterboards* la frecuencia de operación es bastante amplia y dependerá del modelo de tarjeta disponible. En la *Motherboard* de los dispositivos *USRP E110*, las señales análogas son convertidas a muestras digitales y mezcladas en banda base dentro de la *FPGA*, también dentro de la *FPGA* se realizan procesos de decimación e interpolación de la tasa de muestro de la señal.

Los datos muestreados por la *FPGA* son enviados hacia el procesador del dispositivo *USRP E110* en donde la herramienta *GNU Radio* controla las capacidades adicionales de procesamiento de señales. *GNU Radio* es una herramienta de código abierto, la cual ofrece diversos bloques de procesamiento de señales pre-ensamblados para el tratamiento de formas de onda y análisis del desarrollo de *software* radio. Mayor información acerca del *software GNU Radio* se detallará en el Capítulo 3. del presente documento.

La idea de la arquitectura es que las etapas de conversión analógico/digital estén lo más cercanas posible a la antena, de hecho la separación de portadoras y la conversión de frecuencias *up/down* son desempeñadas por los recursos de procesamiento digital, al igual que la codificación de canal y las modulaciones.

En general, muchas de las operaciones del procesamiento de la señal se realizan dentro de un procesador, en lugar de utilizar dispositivos de *hardware* de propósito específico, esta es una de las grandes ventajas de la tecnología *SDR* puesto que permite realizar cambios en la configuración del sistema simplemente manipulando el *software*.

El *software* del sistema se encuentra estratificado en capas de la misma manera que sucede en muchas otras arquitecturas, con el objetivo de hacer de este un software de tipo modular y adaptable al *hardware* sobre el cual operará. Al realizar diseños en *SDR* se busca que la arquitectura de software se encuentre basada completamente en objetos, de tal manera que el *hardware* pueda ser de alguna manera "mapeado" a estos objetos para proporcionar servicios de comunicación entre capas utilizando interfaces estándar. El resto de componentes que integran el software son los comunes a los equipos terminales y a los equipos de transmisión e interconexión (sistemas operativos, API, controladores, etc.) que facilitan el desarrollo de aplicaciones por parte de los programadores. En el Capítulo 3. se detalla la estratificación del *software* que compone la plataforma de trabajo usada en el presente proyecto.

El *hardware* existente dentro del subsistema digital es el encargado de proporcionar toda la flexibilidad y reconfigurabilidad física al sistema. Normalmente este *hardware* se encuentra constituido por dispositivos *DSP*, siendo también frecuentes las implementaciones con *FPGA* y *ASIC* para el desarrollo de las diversas funciones que desempeña este subsistema, para las cuales cada tipo de dispositivo ofrece ventajas y desventajas significativas en su elección como plataforma *hardware* de implementación.

2.3.2 Características de los dispositivos SDR

Debido a la naturaleza programable de una plataforma *SDR*, sus bloques funcionales pueden ser cambiados en tiempo real y sus parámetros de operación pueden ser ajustados bien por un operador humano o por un proceso automatizado. En general, las siguientes características deben estar siempre presentes cuando se habla de una plataforma *SDR* (*Digital Communication Systems Engineering with Software-Defined Radio*, 2013)

- Multifuncionalidad: Debe poseer la habilidad de soportar múltiples tipos de radio funciones utilizando la misma plataforma digital de comunicaciones.
- Movilidad Global: Operación transparente con distintas redes de comunicación localizadas en distintas partes del mundo.
- Eficiencia de potencia y tamaño: Muchos estándares de comunicaciones pueden ser soportados solo con una plataforma *SDR*.
- Fácil de fabricar: Las funciones de banda base pasan a ser problema de *software* no de *hardware*.
- Fácil de actualizar: Las actualizaciones de *Firmware* pueden ser ejecutadas en la plataforma *SDR* para permitir la funcionalidad con los últimos estándares de comunicación.

2.3.3 Tipos de SDR

Existe una clasificación de dispositivos conforme a la aplicación y configuración del *SDR*.

- Tipo I: Son en general implementaciones que emplean una tarjeta de audio convencional de una PC como digitalizador y software convencional de PC como elemento de procesamiento. Es probablemente el tipo que es más accesible para radioaficionados, de este tipo de dispositivos se desprende una subcategoría que los clasifica de acuerdo al manejo de la tarjeta de audio.

    1. Tipo Ia: El *SDR* se implementa alimentando a la tarjeta de sonido la salida de audio de un receptor convencional de comunicaciones.
    2. Tipo Ib: El *SDR* realiza el procesamiento introduciendo a la tarjeta de sonido una señal mono que representa una frecuencia intermedia de aproximadamente 12 kHz.

3. Tipo Ic: El *SDR* procesa introduciendo a la tarjeta de audio una señal I+Q que representa una frecuencia intermedia en el rango de frecuencias que la tarjeta de audio puede manejar. Este es el tipo quizás más potente, con mejor relación costo-prestación y de mayor atractivo para los aficionados.
4. Tipo Id: El *SDR* se procesa introduciendo una señal I+Q en un digitalizador y procesador de señales especializado (no en una tarjeta de audio).

- Tipo II: El *SDR* se implementa con un dispositivo especial que se encarga de capturar la señal desde la antena y la procesa a partir de allí.
- Tipo III: El *SDR* se implementa con un dispositivo especial que captura la señal desde una IF analógica y la procesa a partir de ahí.
- Tipo IV: El *SDR* es implementado por receptores especiales que toman la señal directamente desde su fuente en la frecuencia de trabajo y la procesan en toda la cadena.
- Tipo V: Este grupo representa la categoría de *SDRs online*, son radios *software* que están implementados en un servidor capaz de proveer parcial o totalmente la capacidad de procesamiento digital de señales, tienen cierta utilidad práctica para radio aficionados puesto que no se requiere de la compra de un equipo para el procesamiento de la señal.

2.3.4 Ventajas y Desventajas del Desarrollo de Aplicaciones con SDR

Las principales ventajas del uso de estas arquitecturas para la generación de sistemas de comunicaciones van desde facilidades para el *Roaming* global debido a las características de adaptabilidad de plataforma y protocolos hasta la versatilidad del *software*

SDR para proporcionar nuevos y mejores servicios sin necesidad de cambiar los terminales u otros equipos relacionados.

Desde el punto de vista legal, muchos aspectos regulatorios pueden verse facilitados, especialmente en temas como la certificación de terminales teniendo en cuenta que es solamente el *software* lo que debe certificarse. También, la tecnología *SDR* es ampliamente utilizada en sectores como la telefonía celular y comunicaciones militares, ya que en estos sectores se requiere de procesamiento en tiempo real y sus necesidades cambian casi constantemente.

En otros aspectos, las características de "solamente *software*" que ofrece *SDR* permite contribuir con el cuidado del medio ambiente al evitar el desperdicio de componentes de *hardware* y basura tecnológica.

El usuario final de *SDR* se ve beneficiado con características como la calidad de servicios de soporte de los operadores, ya que las actualizaciones, corrección de *bugs*, adición de servicios y seguimiento a los usuarios, se realizará de manera más eficiente.

Pero también existen ciertos problemas y limitantes relacionados con la implementación actual de esta tecnología:

- La capacidad de *software* requerido para la configuración de los dispositivos será mayor conforme se vayan reemplazando los componentes de la capa física, esto exigirá complejidad en estos elementos, y como primera consecuencia la necesidad de mayor capacidad de almacenamiento del dispositivo incrementará gradualmente.
- El tiempo de configuración de los dispositivos *hardware* aumenta conforme la complejidad del software.
- Los procesos de estandarización se desarrollarán a un ritmo más lento que los

procesos de investigación y desarrollo.

- La implementación en *software* hará que el sistema sea más vulnerable a amenazas informáticas como virus.

- Conforme incremente la complejidad del sistema surgirán inconvenientes relacionados al procesamiento como son la velocidad de muestreo y la capacidad de manejo de señales por parte del dispositivo digitalizador, ya que como sabemos el propio procesamiento introduce ya problemas que distorsionan la señal, lo que no ocurre en un sistema análogo.

Entonces, el futuro y el alcance de la tecnología *SDR* va muy de la mano con el desarrollo que presenten campos como la nanotecnología o la microelectrónica, esto en términos de miniaturización de componentes, incremento en la capacidad de procesamiento, reducción de consumo de potencia, modularidad de componentes, etc. Y también de otro aspecto muy importante, el marco regulatorio para el establecimiento comercial, militar y civil de sistemas *SDR* que cumplan con las normas y estándares regulatorios exigidos en varios países.

2.4 Universal Software Radio Peripheral USRP

Los dispositivos *USRP* comprenden el elemento físico en el cual se implementa la tecnología SDR. Se definen como un *hardware* genérico para transmisión y recepción de señales. Se componen principalmente por una tarjeta madre que puede ser conectada con varias tarjetas hijas *(Daughterboards)*, las cuales proveen una variedad de interfaces desde simples filtros análogos hasta conversión compleja de circuitos de modulación y demodulación para varias bandas de frecuencia.

Los dispositivos *USRP* pueden ser configurados para su uso en el desarrollo de varias aplicaciones, las cuales van desde su funcionamiento como sensores distribui-

Tabla 1: Recomendaciones de USRP y Daughterboard para varias aplicaciones.

Área de Aplicación	**Modelo USRP Recomendado**	**Daughterboard Recomendada**
Investigación PHY/MAC	N200/210	SBX
Investigación con Radares	N200/210	SBX
Estaciones OpenBTS	B100	WBX/SBX
Educación	N200/210	WBX/SBX
Comunicaciones en HF	B100	LFRX/LFTX
Inteligencia de señales	N200/210	WBX/SBX
Distribución de sensores RF	E100/E110	WBX/SBX
Radios móviles	E100/E110	WBX/SBX

dos de RF hasta comunicaciones satelitales y de inteligencia militar. Cada una de estas aplicaciones va de la mano con un dispositivo específico. La compañía *Ettus Research* ofrece una gama de modelos *USRP*, sus respectivas placas *Daughterboard* y la aplicación en la que se desarrollará con mayor eficiencia como se describe en la Tabla 1.

Para la elección de un dispositivo *USRP* ha de determinarse el tipo de interfaz que se usará para la comunicación con el dispositivo, las capacidades de ancho de banda, los mecanismos de sincronización que son específicos para cada modelo etc. Las Tablas 2 y 3 tomadas de (EttusResearch, 2012a) ilustran información general acerca de los modelos disponibles en *Ettus Research* y sus características generales.

Antes de elegir un dispositivo *USRP* deben considerarse varios temas. El primero de ellos es la elección de la manera en la cual se desarrollará el procesamiento: si se usará una PC como *host* o se operará con el *USRP* de manera autónoma, este es un tema que diferencia en gran medida el modelo de *USRP* a usar. El presente proyecto busca

Tabla 2: Características USRP por modelo (Parte 1).

Modelo USRP	**Interfaz**	**BW Total** (MSPS 16b/8b)	**Ranuras Daughterb.**	**Resolución** ADC (bits)	**Tasa** ADC (MSPS)
N210	Gig. Eth.	500/100	1	14	100
N200	Gig. Eth.	500/100	1	14	100
B100	USB 2.0	8/16	1	12	64
USRP1	USB 2.0	8/*	2	12	64
E100	Embebido	8/16	1	12	64
E110	Embebido	8/16	1	12	64

Tabla 3: Características USRP por modelo (Parte 2).

Modelo USRP	**Resolución** DAC (bits)	**Tasa** DAC (MSPS)	**Capacidad MIMO**	**Oscilador Programable**	**Entradas** 1PPS/Ref
N210	16	400	Sí	Sí	Sí
N200	16	400	Sí	Sí	Sí
B100	14	128	No	No	Sí
USRP1	14	128	Sí	No	No
E100	14	128	No	Sí	Sí
E110	14	128	No	Sí	Sí

operar un *USRP* como radio sin una PC actuando como *host*, por lo tanto los dispositivos *USRP E110/E100* son los más apropiados ya que ofrecen un entorno de procesamiento embebido dentro del dispositivo. El *USRP E110/E100* también es apropiado para aplicaciones que puedan requerir funciones de *transceivers* móviles o la distribución de sensores de *RF*. Para aplicaciones más complejas el fabricante *Ettus Research* recomienda las series no embebidas, cuyo desarrollo con plataformas basadas en computadores actuando como *host* involucará menos riesgo y requerirá menos esfuerzo para la optimización de los componentes programados en *software*.

En varios casos puede ser de bastante utilidad el desarrollo de aplicaciones con un *USRP N200/210* y una vez probado el sistema dentro de este dispositivo migrarlo hacia un *USRP E100/110*. Esta ventaja de portabilidad es concedida por el *UHD - USRP Hardware Driver*. También deberá considerarse las capacidades de procesamiento en la PC *host* y el procesador *OMAP (Open Multimedia Applications Platform)* en el caso de los modelos *USRP E100/E110*.

Los dispositivos *USRP E100/E110* no son apropiados para aplicaciones que requieran un sistema *MIMO*. Sin embargo, si se busca operar en bajas frecuencias puede ser posible que en conjunto con tarjetas *Daughterboards* que soporten esas frecuencias pueda generarse dos canales en cada tarjeta *Daughterboard*. La Tabla 4 ilustra las capacidades de los distintos modelos *USRP* disponibles por *Ettus Research* en cuanto a características de sincronización.

También han de tomarse en cuenta requerimientos de ancho de banda necesarios para el desarrollo de cada aplicación. Como puede verse en la Tabla 4, el *USRP N200/N210* es capaz de trabajar con un flujo de datos de hasta 50 Msps en cada dirección en un modo de 8 bits, y 25 Msps en un modo de 16 bits lo que lo hace apropiado para la transmisión y recepción de señales con gran ancho de banda como las requeridas en el estándar 802.11 (Group, 2012).

Tabla 4: Características Sincronización USRP.

Modelo USRP	**Capacidad de BW** (MSPS w/16bit)	**Capacidad MIMO**	**Entrada** Ext Ref.	**Entrada** 1PPS	**Osilador GPS** *Interno*
N210	25	X	X	X	X
N200	25	X	X	X	X
B100	8		X	X	
USRP1	8	X			
E100	4		X	X	X
E110	4		X	X	X

Para el caso del *USRP E110* el cual posee una interfaz *FPGA* puede proveer un *throughput* de 40 Mbps. Este ancho de banda puede ser usado para la transmisión y recepción de datos. Por ejemplo, a una tasa de 4 *bytes*/muestra se generan alrededor de 10 Msps. El fabricante hace hincapié en que esta característica no garantiza que el procesador embebido sea capaz de procesar tal cantidad de muestras.

Tómese en cuenta que las limitaciones de estos dispositivos se encuentran relacionadas directamente con su interfaz específica y es importante considerar el desempeño de la plataforma de procesamiento, en especial cuando se trata de dispositivos embebidos, así como el costo computacional de la aplicación que se desarrolla.

En cuanto a la interfaz de trabajo para la comunicación con el dispositivo, el *USRP N200/210* posee una gran ventaja frente a otros modelos, ya que al estar equipado con una interfaz *Gigabit Ethernet* hace posible una comunicación mucho más rápida con el computador *host* que se encuentre conectado a él y puede extender la distancia que separa el dispositivo y el computador a una mayor longitud manteniendo las mismas características de velocidad de comunicación. También puede accederse mediante un *Switch Gigabit Ethernet* para permitir el acceso a varios dispositivos.

Tabla 5: FPGA's en Dispositivos USRP.

	N210	**N200**	**E110**	**E100**	**USRP1**	**B100**
Vendedor	Xilinx	Xilinx	Xilinx	Xilinx	Altera	Xilinx
Serie FPGA	Spartan 3A DSP	Spartan 3A DSP	Spartan 3A DSP	Spartan 3A DSP	Cyclone	Spartan 3A
Compuertas	3200k	1800k	3200k	1800k	-	1400k
Elementos Lógicos	-	-	-	-	12000	-
Celdas Lógicas	53714	37440	53714	37440	-	-
Slices	23872	16640	23872	16640	-	1264
DSP48's	126	84	126	84	0	0
BRAM	373k	260k	373k	260k	234k	576k
DCM's	8	8	8	8	2	8
Free tools	No	Sí	No	Sí	Sí	Sí

El fabricante *Ettus Research* recomienda una red homogénea de trabajo, es decir sin otros dispositivos conectados a ella, como un *router* por ejemplo. Una de las principales motivaciones para el desarrollo de aplicaciones utilizando dispositivos *USRP* es la capacidad de personalización de la *FPGA* a medida de las aplicaciones que se deseen desarrollar. Un resumen de las *FPGAs* usadas en cada modelo *USRP* se muestran en la Tabla 5.

Los *USRP E100/E110* son plataformas genéricas ideales para la experimentación en el desarrollo de aplicaciones manipulando la *FPGA*. La principal importancia entre cada modelo *USRP* es el tamaño de la tarjeta *FPGA*, y los requerimientos para el desarrollo de herramientas *Xilinx*. El *USRP E100* contiene una *FPGA Xilinx Spartan XC3SD1800A*. Esta *FPGA* está optimizada para capacidades de *DSP*, y su lógica puede ser modificada con las herramientas gratuitas de *Xilinx ISE*, disponibles

en: www.xilinx.com.

El USRP E110 incluye una *FPGA Xilinx Spartan XC3SD3400A*, lo que ofrece casi el doble de recursos, pero requiere una licencia de herramientas de desarrollo *Xilinx* para su manipulación. Los modelos *USRP N200/210* utilizan las mismas FPGAs que los modelos USRP E100/110 respectivamente.

Algunas aplicaciones pueden mejorar su rendimiento al ser beneficiadas con una frecuencia de reloj flexible. Los dispositivos *USRP E100/110* permiten establecer frecuencias de muestreo de reloj ideales para ser usadas en concordancia con varios estándares de comunicaciones. Los estándares *GSM*, utilizados para comunicaciones móviles generalmente implementan aplicaciones usando frecuencias de reloj de 52MHz.

El *driver UHD (Universal Hardware Driver)* permite al usuario el desarrollo de una aplicación que sea compatible con todos los modelos *USRP*, por supuesto con ciertas limitaciones a considerar, como por ejemplo la tasa de muestreo, el ancho de banda de la interfaz con el *host* y detalles de sincronización que aseguren la compatibilidad con el equipo.

CAPITULO 3

HARDWARE Y SOFTWARE

3.1 USRP E110 y *Daughterboards*

3.1.1 *Universal Software Radio Peripheral* E110

Para el desarrollo del presente trabajo se ha elegido el dispositivo *USRP E110* que es parte de la familia de productos *USRP* desarrollado por la empresa *Ettus Reseach*. La elección de este dispositivo viene principalmente por la necesidad del diseño e implementación de un sistema de voz digital embebido; porque con este dispositivo es posible el desarrollo de programas de forma nativa dentro del dispositivo además de la posibilidad de migrar el código desde y hacia otras plataformas *SDR* y porque el fabricante recomienda el uso de este dispositivo para el desarrollo de aplicaciones de radios móviles cuyo funcionamiento sea independiente. El *USRP E110* reúne las características detalladas en la Tabla 6 tomadas de (EttusResearch, 2012a) y (EttusResearch, 2012c).

El *USRP E110* al ser un dispositivo embebido posee su propio sistema operativo con una distribución *Linux* que incluye una versión funcional de *GNU Radio*. Además permite dos configuraciones de *FPGA* distintas, las cuales pueden ser extendidas o

Tabla 6: Características USRP E110.

Elemento	Característica
Procesador	ARM Cortex-A8 (obtiene hasta 8MSps)
DSP	C64 (para aplicaciones embebidas)
FPGA	Xilinx Spartan 3A-DSP 3400
ADC	Dual 64MSps
DAC	Dual 128MSps
Frecuencia de operación	DC-6GHz
Memoria RAM	512MB
Conectividad	Ethernet 100Mbit
Reloj	10MHz a 64MHz
Tasa de Baudios	115200
Potencia de Salida	15dBm

personalizadas a las capacidades de procesamiento de señales para aplicaciones de propósito específico.

El procesador *ARM Cortex A8 de 800MHz* embebido en el dispositivo se encuentra conectado a la *FPGA* por medio de una interfaz de memoria directa, lo que permite el alto flujo de información de banda ancha a través del sistema. Esta interfaz soporta el *Universal Hardware Driver UHD* lo que permite la portabilidad de aplicaciones diseñadas en la mayoría de plataformas.

Un esquema general de la arquitectura de un dispositivo *USRP E110* puede verse a continuación en la Figura 4. La arquitectura modular de este dispositivo trabaja con tarjetas hijas *Daughterboards*, las cuales se encargan de realizar las tareas de *Front End* de la señal *RF*. El *USRP E110* posee dos ranuras para la conexión de una tarjeta transmisora y una receptora, o una sola tarjeta que posea la capacidad de *transceiver*

puede ser también conectada al *USRP E110*.

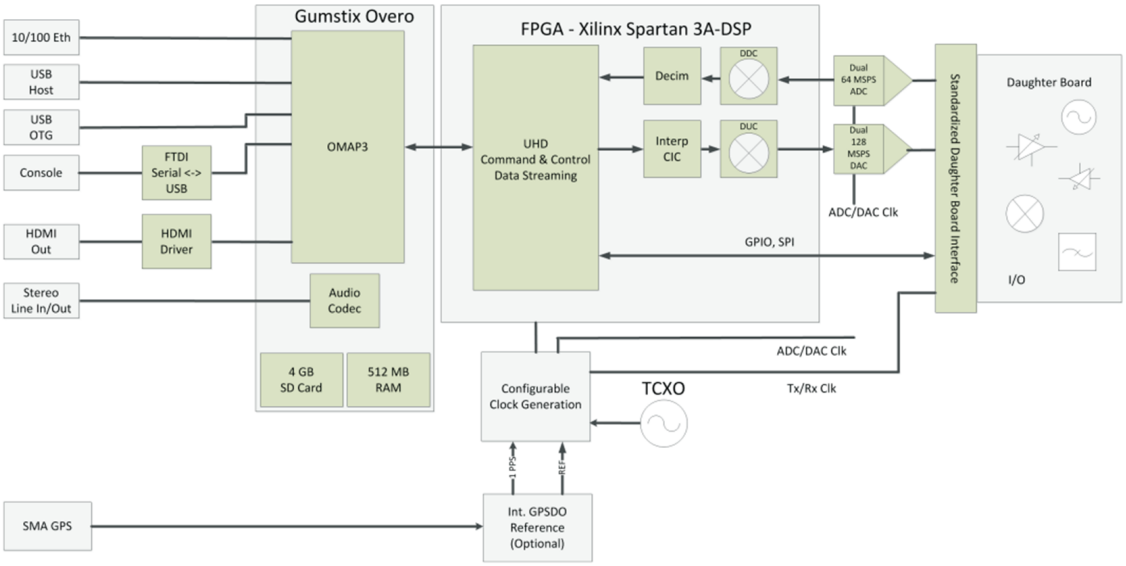


Figura 4: Arquitectura USRP E110.

No debe confundirse la frecuencia de operación del dispositivo con la frecuencia de operación de las tarjetas *Daughterboard*, ya que el rango y banda de operaciones de dichas tarjetas varía dependiendo del modelo. El funcionamiento y la elección de las tarjetas *Daughterboard* se explicará en la sección siguiente.

3.1.1.1 Comunicarse con el *USRP*

Existen varios métodos con los cuales se puede interactuar con el dispositivo:

- Por consola, mediante una interfaz serial.
- Mediante un puerto de red utilizando *SSH*.
- Mediante una interfaz gráfica con *mouse* y teclado.

De todos estos métodos, la interacción por consola es el más simple de todos, y es el que se requiere para la configuración de inicio de los *kernel*, por otro lado, la

interfaz por red es más rápida y es usada para la transferencia de archivos, desde y hacia el dispositivo.

Para el desarrollo del presente trabajo, se ha optado por dos de las tres opciones existentes para la interacción con el dispositivo, el método por consola y la interfaz gráfica.

Para conectar el dispositivo *USRP E110* hacia el computador se debe conectar el aparato a una *PC* (con Sistema Operativo *LINUX*). Los pasos a seguir son:

- Conectar la *PC* por medio de un cable *USB Standard-A* y un conector *Mini USB* hacia el equipo.
- Colocar el conector *Mini USB* en el puerto *"CONSOLE"* del *USRP E110*.
- El sistema operativo *LINUX* asignará al dispositivo una dirección, la cual puede ser visualizada al ejecutar el comando `dmesg`. Podrá observarse dentro de toda la información que se despliegue, una línea que muestre lo siguiente: *"... now attached to ttyUSB"*.
- *ttyUSB0* es ahora la dirección del dispositivo dentro del sistema.
- Asumiendo que el dispositivo se encuentra como *ttyUSB0*, puede usarse el comando siguiente para conectar el aparato y usarlo en modo consola: `sudo screen /dev/ttyUSB0 115200,cs8,-ixon,-ixoff`.
- Hecho esto se iniciará el proceso de arranque del dispositivo, y se verá la pantalla de inicio.

La conexión mediante el puerto *HDMI* del dispositivo permitirá accesar al sistema operativo embebido, y mediante un *HUB USB* puede conectarse un *mouse* y un teclado, se ampliará este método en la sección referente al sistema operativo *Angstrom* (Distribution, 2013).

3.1.1.2 Cambiando la tasa del *Clock Master*

Una de las grandes ventajas del *USRP E110* es su capacidad en la flexibilidad de la frecuencia de oscilación del reloj. El *Clock Master* de los dispositivos embebidos *USRP E110* alimenta tanto a la tarjeta *FPGA* y al *chip* que contiene los *codecs*. La flexibilidad de configuración de frecuencias del reloj permite un gran número de tasas desde 32MHz hasta 64MHz (EttusResearch, 2012b). Algunas de las más comunes son:

- 64 MHz: Tasa máxima del chip
- 61.44 MHz: Para aplicaciones *UMTS/WCDMA*
- 52 MHz: Para aplicaciones *GSM*

Para usar el reloj con una frecuencia de 61.44MHz dentro de un dispositivo *USRP* embebido se hace uso del Oscilador Controlado por Voltaje Externo *VCXO*. Para ello deben moverse dos *jumpers* en el dispositivo.

- J16 Posee dos pines; retirar el jumper, o dejarlo solamente conectado a un *pin*.
- J15 Posee tres pines; mover el jumper de modo que conecten solamente el *pin* 1 y *pin* 2.

Para establecer otras tasas de oscilación del reloj se puede usar el *VCO* interno del dispositivo. Para ello, seguir la siguiente configuración en los *jumpers*:

- J16 Posee dos pines; mover el jumper de modo que conecte (pin1, pin2).
- J15 Posee tres pines; mover el jumper de modo que conecten (pin2, pin3).

Para comunicar al *software UHD* la frecuencia de oscilación deseada en el reloj deberá especificarse al equipo dicha frecuencia mediante una instrucción argumento, la cual es `master_clock_rate` junto con el valor de a frecuencia en Hz. Por ejemplo: `uhd_usrp_probe - - args="master_clock_rate=52e6"`. Este comando establecerá una frecuencia de oscilación de 52MHz.

3.1.1.3 Leds de Status

El dispositivo *USRP E110* posee seis luces *LED* que indican el estado y las operaciones que se encuentra realizando el dispositivo. A continuación se listan los distintos *LEDs* existentes y su estado asociado.

- **LED A**: Transmitiendo
- **LED B**: Señal *Pulse-Per-Second* (para sincronización CLK)
- **LED C**: Recibiendo
- **LED D**: FPGA Cargada
- **LED E**: Bloqueo de referencia
- **LED F**: Placa encendida

3.1.2 Tarjetas *Daughterboard*

Como se ha mencionado anteriormente los dispositivos *USRP* poseen una tecnología modular, que los hace versátiles al momento de desarrollar distintas aplicaciones. En conjunto con la tarjeta principal del dispositivo deberán agregarse tarjetas trabajando como *Front End*, y que permitan realizar la correcta trasmisión entre ambos *USRP*.

Ettus Research ofrece una gran variedad de tarjetas hijas *Daughterboards* dependiendo de la banda de frecuencia, el número de canales en los cuales trabaje la aplicación a desarrollar. Las aplicaciones móviles *GSM*, las cuales hacen uso de modulaciones *GMSK* usualmente trabajan en la banda de 900MHz a 1800MHz, las cuales son bandas de teléfono celular. La Figura 5 ilustra la gama de frecuencias que cubre cada tipo de tarjetas (los códigos de color indican cuando una tarjeta incluye capacidad de transmisión, recepción o ambas).

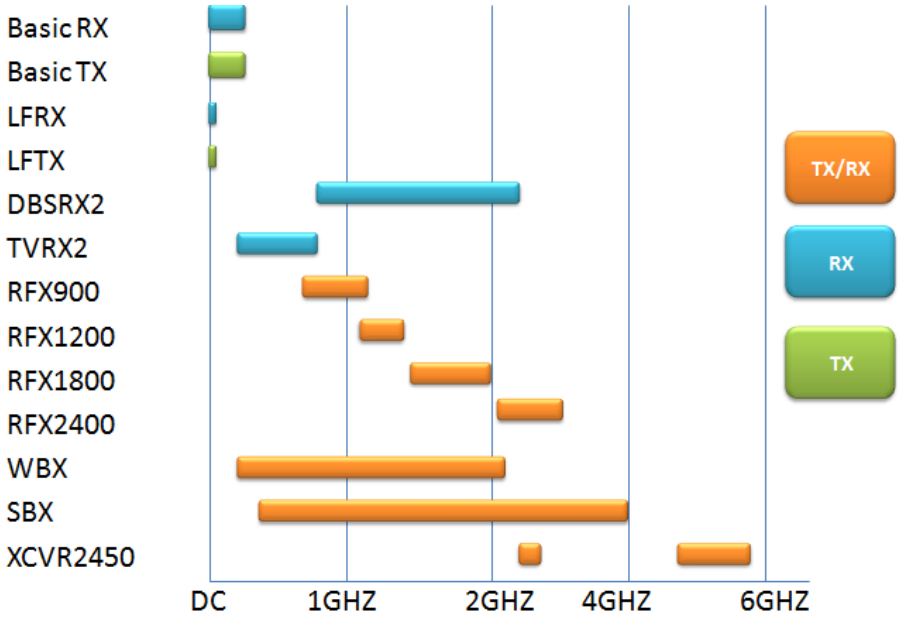


Figura 5: Frecuencias de Trabajo Tarjetas Daughterboard.

La Tabla 7 ilustra algunas de las aplicaciones que pueden realizarse con dispositivos *USRP*. Nótese que el fabricante *Ettus Research* recomienda que para el desarrollo de aplicaciones *GSM* sean utilizadas las tarjetas *WBX*, y *SBX*.

Para la selección de una tarjeta *Daughterboard* basado en el rango de frecuencias que se desea cubrir debemos tener en cuenta el ancho de banda que requiera la aplicación a desarrollarse, en el caso de *GMSK* debido a que es una modulación con un ancho de banda bastante pequeño este aspecto no deberá ser un limitante. La capacidad de usar el ancho de banda total depende del dispositivo *USRP* al cual se encuentra conectada la *Daughterboard* y el emparejamiento de ambos con la resolución de los datos transferidos desde la interfaz del *USRP*. Esto nos lleva a otro punto importante a considerar: la capacidad de ancho de banda del dispositivo *USRP* seleccionado. Puede ser que una placa *Daughterboard* tenga 40MHz de ancho de banda por ejemplo, esto no garantiza que el dispositivo *USRP* pueda transferir un ancho de banda igual hacia la *Daughterboard*.

Tabla 7: Daughterboards Recomendadas para Varios Rangos de Frecuencia y Aplicaciones.

Aplicación	Rango de Frec.	Tx/Rx	Daughterboard
Rx TV	54-806MHz	Rx Only	TVRX2,WBX
Rx GPS	L1-1575.42MHz	Rx	DBSRX2,WBX
	L2-1227.60MHz	Rx	DBSRX2,WBX
OpenBTS &	GSM900-890-960MHz	Tx/Rx	WBX,SBX
Basestation GSM	GSM1800-1850-1989MHz	Tx/Rx	WBX,SBX
WiMAX	2.5GHz	Rx	SBX
Rx FM	88-108MHz	Rx	TVRX2, WBX
Desarrollo 802.11B/G/N	2.4GHz, 5GHz	Tx/Rx	XCVR2450
Comunicaciones HF	3-30MHz	Tx/Rx	LFRX+LFTX
Investigación Radares	2-4GHz	Tx/Rx	SBX

Algunas aplicaciones requerirán elegir un *Front-End* externo para realizar operaciones de *Up-conversion*, *Down-conversion*, amplificación y funciones de filtrado. En estos casos el *Front-End* tendrá como salida una frecuencia intermedia *(IF)*. Es por ello que se ha seleccionado las tarjetas *RFX1800 1.5-2.1 GHz Rx/Tx* las cuales se encuentran diseñadas para su operación en las bandas de *1900* MHz, y tienen las especificaciones técnicas descritas en la Tabla 8.

Tabla 8: Daughterboards Recomendadas para Varios Rangos de Frecuencia y Aplicaciones.

Banda de Frecuencia	1500MHz - 2100MHz
Potencia de Salida	100mW - 20dBm
Figura de Ruido	8dB
Ancho de Banda	40MHz
Especificaciones de canal	1 Tx IQ, 1Rx IQ, FD
Oscilador Local Independiente	Sí
Capacidad MIMO	Sí

3.2 Sistema Operativo *Angstrom*

En el *USRP*, las altas tasas de muestreo y procesamiento se realizan en la *FPGA*, mientras que los procesos con baja tasa de muestreo ocurren en la tarjeta *SD* en la cual se encuentra el sistema operativo *Angstrom*.

Para utilizar el *E110* mediante la interfaz gráfica se requiere de un monitor y un cable *HDMI/DVI-D* para conectarlo al dispositivo *E110* que también posee un puerto de este tipo. Al iniciar el equipo se mostrará la pantalla de inicio del sistema operativo *Angstrom*, en donde el nombre de usuario y la contraseña son las siguientes:

Username: root

Password: usrpe

La Figura 6 ilustra el esquema de conexión del dispositivo *USRP E110* mediante un cable *HDMI* y periféricos *USB*. Debido a las características de multifuncionalidad del equipo pueden agregarse varios periféricos a este, como un *mouse* y un teclado por medio de un *HUB USB* conectado en el puerto *"USB HOST"* usando un conector *USB Mini-A*.

Observación: No conectar el puerto *"USB HOST"* hacia un puerto *downstream* en un PC, puesto que esto puede ocasionar daños al aparato.

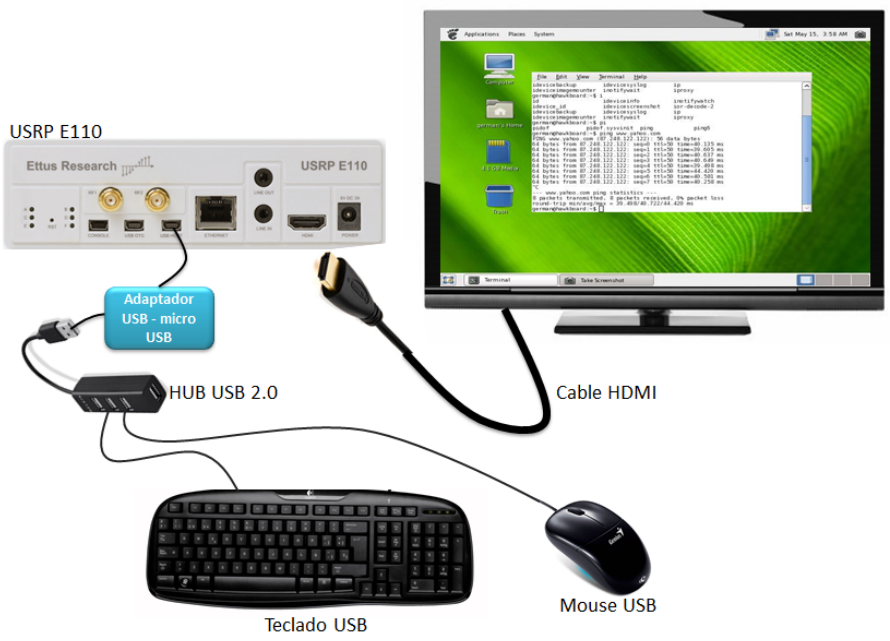


Figura 6: USRP con interfaz HDMI.

3.2.1 Creación y Re-Creación de las Imágenes en la tarjeta *SD*

Como se ha explicado anteriormente, el sistema operativo del *USRP E110* se encuentra dentro la tarjeta *Micro-SD* que contiene. Este sistema operativo puede ser modificado dependiendo de las necesidades del usuario.

Existen dos métodos por los cuales se puede grabar una imagen maestra en las tarjetas *SD* del *USRP E110*

- Mediante un "*build directory*" con un script *make*. Estos archivos se encuentran contenidos en un *tarball: e1xx-xxx-make.tar.bz2*
- Mediante una imagen exacta de la imagen maestra de la tarjeta *SD*. Esta imagen es distribuida como *e1xx-xxx.bin.bz2*

Según el fabricante, la primera es la mejor opción (Research, 2013). La segunda opción requiere una tarjeta *SD* que sea exactamente del mismo tamaño que la tarjeta

SD con la cual se ha grabado la imagen maestra. Esto resulta un problema muy grande, puesto que aunque las tarjetas *SD* del mercado sean del mismo tamaño nominal, siempre variarán entre sí por algunos cientos de *bytes*. Esto afectará la copia *byte-a-byte* de una tarjeta a otra.

3.2.2 Grabar una Imagen Utilizando un *"Build Directory"*

El directorio para la descarga de imágenes puede ser encontrado en:

`http://files.ettus.com/e1xx_images/`

de donde se ha procedido a descargar el archivo `e1xx-xxx-make.tar.bz2` para la versión utilizada en este proyecto.

Una vez descargado, deberán seguirse los pasos siguientes:

- Descomprimir el directorio con el comando:

  `tar jxvf e1xx-xxx-make.tar.bz2`

  Se tendrá un directorio *"e1xx-xxx"* con todos los archivos necesarios para crear una copia de la imagen maestra.

- Colocar la tarjeta *SD* en el computador (Se asume un sistema *LINUX*). Si las particiones se montan automáticamente, desmontarlas, no mediante la extracción segura del dispositivo sino utilizando el comando *"umount"* para cada partición, de esta manera: `sudo umount /ruta/de/la/partición/a/desmontar`.

  Puede ejecutarse el comando `fdisk` para averiguar cual de los discos montados corresponde a la tarjeta *SD* del dispositivo `sudo fdisk -l`. Este comando mostrará las particiones existentes. Deberán tener nombres como `/dev/sdb1` o `/dev/sdb2`.

- Se hará uso del *script 'MakeEttusSDCard.sh'* para crear la imagen. Este *script*

funcionará con cualquier tamaño de tarjeta *SD*.

```
sudo ./MakeEttusSDCard.sh /nombre/partición
```

Verificar en que partición se estará ejecutando el script. Puede darse el caso que en algunas distribuciones de *Linux*, el script *'MakeEttusSDCard.sh'* genere una imagen que no arranque, en este caso puede utilizarse el *script 'MakeEttusSD-Card.legacy.sh'* en su lugar.

El proceso de ejecución y quemado de la imagen tomará un tiempo relativamente largo, al final del cual se informará que el proceso se ha completado con éxito.

3.2.3 Grabar la Imagen Maestra de *Ettus*

Para ello seguir los pasos siguientes:

- Dirigirse a la página `http://files.ettus.com/e1xx_images/` en donde se deberá seleccionar el directorio para la versión de la imagen que desee crearse y descargarla.
- Descomprimir la imagen `bunzip2 e1xx-xxx.bin.bz2`
- Puede restaurarse cualquier imagen en una tarjeta *SD* mediante el comando `'dd'`. Se debe ser muy cuidadoso al usar este comando porque si se le pide al comando `'dd'` su ejecución en un dispositivo equivocado, podría sobre escribir la imagen sobre el disco duro del computador.

  ```
  sudo dd bs=1024 if=sd-backup.bin of=/dev/sdb
  ```

- El comando `'dd'` toma bastante tiempo en completar su ejecución. Utilizará una gran porción de memoria, no debe sorprendernos si el equipo empieza a ralentizarse mientras este proceso se completa.

3.2.4 *Firmware* y *Drivers*

Las características de *"solamente software"* de los dispositivos *USRP* requieren actualizaciones del *kernel*, archivos de arranque y demás módulos instalados para coincidir con versiones más recientes de *software*. También el sistema de archivos *root* que incluye los módulos de *kernel* puede ser actualizado. Los procesos de actualización pueden ser divididos en tres categorías en función de las necesidades del desarrollador:

- Actualizar solamente el *kernel* y archivos de arranque.
- Actualizar los módulos del *kernel* en un sistema de archivos *root* existente.
- Actualizar completamente el sistema de archivos *root*. (incluye los módulos del *kernel*)

Adicional a esto es necesario actualizar el *driver UHD*, el *firmware* de la *FPGA* y el *software GNU Radio*.

Las instrucciones de actualización de *drivers* e imágenes maestras se encuentran disponibles en el siguiente enlace:
`http://code.ettus.com/redmine/ettus/projects/usrpe1xx/wiki/FAQ`

3.3 *GNU Radio* y *GNU Radio Companion*

Los componentes *software* caracterizados en la tecnología *SDR* contemplan dos secciones:

- *Software* de control.
- *Software* de procesamiento de señales.

El *software* de control depende en gran medida del fabricante de la plataforma *hardware*, quien suele recomendar o proveer (como es el caso) las herramientas *software* necesarias para el procesamiento de la señal.

3.3.1 *USRP Hardware Driver UHD*

Es el *driver* requerido para trabajar con dispositivos *USRP*. Es una librería escrita en *C++* pensada para trabajar en plataformas *Linux*, *Windows* y *Mac OS*. Este *driver* es el encargado de proveer control sobre los productos de *Ettus Research*. Puede ser utilizado de manera autónoma e independiente o en conjunto con aplicaciones como:

- **GNU Radio**: Herramienta *software*, libre y de código abierto de desarrollo. Entrega la posibilidad de implementación de sistemas *SDR* mediante el uso de bloques de procesamiento de señales.
- **LabVIEW**: Plataforma de desarrollo y diseño de sistemas *hardware* y *software*, usa un lenguaje de programación gráfico (lenguaje G). Creada por *National Instruments*.
- **Simulink**: Entorno de programación visual de diagramas de bloques integrado en *Matlab*. Usado para simulación de sistemas, puede trabajar con *hardware USRP* para implementar físicamente sistemas *SDR*.
- **OpenBTS**: Aplicación de *Unix* que trata de presentar la interfaz *GSM* mediante el uso de *SDR*.

El presente proyecto hará uso de la herramienta *GNU Radio* como *software* de procesamiento de señales.

3.3.2 *GNU Radio*

El proyecto GNU radio (GNURadio, 2014) se inició en 2001 y fue fundado por Eric Blossom con la finalidad de desarrollar un marco de trabajo para *SDR*. Es una herramienta *software* libre y de código abierto, constituida por un conjunto de archivos y librerías que proporcionan bloques de procesamiento de señales, permitiendo el diseño y simulación de sistemas *SDR*.

GNU Radio puede ser utilizado para simulación, o junto con *hardware* externo adicional para la implementación física de sistemas *SDR*. El funcionamiento de *GNU radio* puede ser entendido como un diagrama de flujo, donde cada nodos simbolizan un bloques de procesamiento de señales, y la interconexión entre ellos determina el camino que seguirá la señal comenzando en una fuente (*Source*) y terminando en un sumidero (*Sink*).

La Figura 7 representa el funcionamiento de *GNU* radio en una aplicación en la que se dispone de una fuente de datos, de un sumidero de datos y de tres bloques que desempeñarán alguna función de procesamiento. Puede entenderse entonces que los tipos de bloques usados en *GNU Radio* pueden ser clasificados en tres grupos:

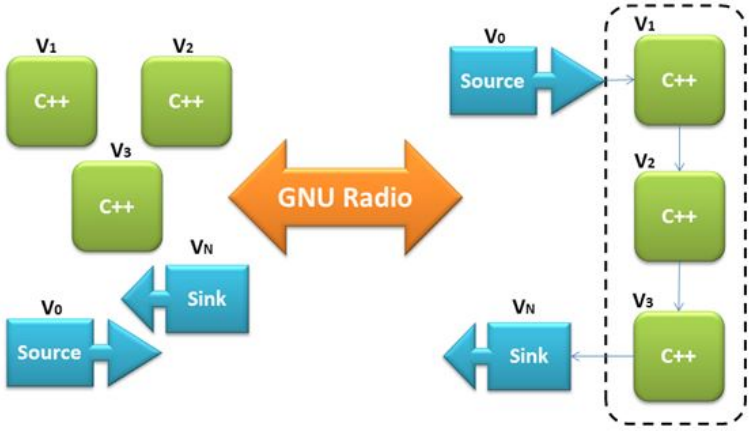


Figura 7: Relación interdependiente en GNU Radio.

- **Fuentes**: Ficheros, otros programas, *hardware* radio, periféricos de entrada (micrófono).
- **Sumideros**: Ficheros, otros programas, *hardware* radio, visualizadores gráficos de forma de onda de la señal, periféricos de salida (parlantes).
- **Bloques de procesamiento de señal**: Amplificadores, moduladores, demoduladores, filtros, operadores lógicos y matemáticos, etc.

La sección de procesamiento de un sistema tradicional que es llevado a cabo por *GNU Radio* se muestra en la Figura 8 en donde se muestran las funciones llevadas a cabo por esta herramienta tanto en la cadena de transmisión como en la de recepción.

Figura 8: Funciones de procesamiento en cadena de transmisión y recepción en GNU Radio.

En *GNU Radio* los bloques procesan los datos de manera continua desde los bloques de entrada hacia los de salida, idealmente cada bloque se encarga de realizar una única función, lo que dota a *GNU Radio* sus características de flexibilidad.

Cada bloque posee determinado número de puertos de entrada y salida, así como un tipo de dato específico. Una fuente posee un puerto de salida, mientras que un sumidero está constituido únicamente por puertos de entrada. Toda aplicación generada en *GNU Radio* deberá constar al menos de una fuente y un sumidero.

Los tipos de datos que maneja *GNU Radio* son:

- **byte**: 1 *byte* de datos
- **short**: 2 *bytes* de datos
- **int**: 4 *bytes* de datos
- **float**: 4 *byte* de datos para números en punto flotante
- **complex**: 8 *bytes* de datos, un *float* para cada componente.

De la misma manera, *GNU Radio* en su interfaz gráfica asigna un color específico en cada puerto para identificar el tipo de dato con el que trabaja el bloque. La Figura 9 muestra los colores asociados por *GNU Radio* a cada tipo de dato.

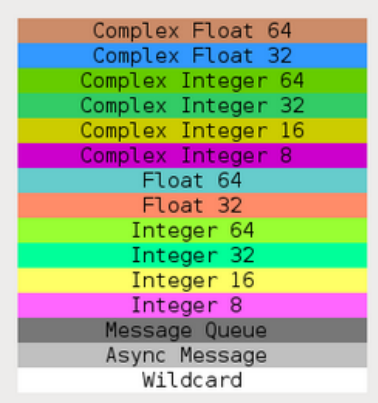


Figura 9: Relación color - tipo de dato en GNU Radio.

El procesamiento de la señal en bajo nivel se encuentra implementado en lenguaje *C++*, mientras que la aplicación se encuentra escrita en lenguaje *Python* quien se encarga de la interconexión de bloques. *Python*, que es un lenguaje de *script* accede a las funciones implementadas en *C++* por medio de la herramienta *software Simplified Wrapper and Interface Generator* SWIG. La creación de la aplicación puede realizarse en dos maneras:

- Programando directamente la aplicación en *Python*.

- Diseñar la aplicación con la herramienta gráfica *GNU Radio-companion*

El diseño contemplado en el presente proyecto se encuentra realizado con la herramienta *GNU Radio-companion* por las facilidades que brinda el establecimiento de un diagrama de flujo con elementos gráficos.

Los bloques de procesado de señal se implementan en cuatro tipos de archivos:

- **Archivos .xml**: Definen parámetros como el tipo de dato, número de puertos de entrada y salida, librerías, etc.
- **Archivos .h**: Bibliotecas de los bloques de procesado de señal.
- **Archivos .cc**: Contienen la función que realizará el bloque de procesado de señal, se escriben en *C++*
- **Archivos .i**: Encargados de la comunicación entre bloques de procesamiento de señales y la interfaz en Python.

Entonces, la arquitectura de *GNU Radio* puede verse como un sistema estratificado con tres niveles de abstracción y una interfaz, tal y como se muestra en la Figura 10. De esta estructura puede obviarse la capa correspondiente a *GNU Radio-companion* ya que es posible diseñar aplicaciones directamente con líneas de comandos en lugar de una interfaz gráfica.

El hacer uso de descriptores `xml` vienen dados por la disponibilidad de bloques en la interfaz gráfica y la interfaz *SWIG* está relacionado con las capacidades de embeber código *C++* en *Python*. El motivo de usar C++ como lenguaje nuclear de la estructura es porque se trata de un lenguaje de programación de nivel intermedio, con buen rendimiento y la capacidad de permitir programación orientada a objetos.

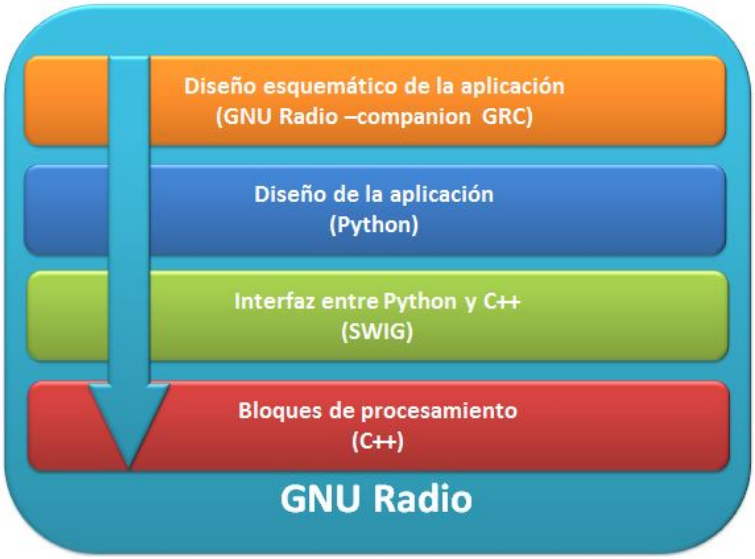


Figura 10: Arquitectura GNU Radio.

En cuanto a los aspectos negativos de *C++* está el no presentar buenas prestaciones al momento de realizar interfaces para interacción con el usuario. *C++* tampoco es un buen lenguaje para la integración estratificada. En este punto entra *Python* como lenguaje de alto nivel de *scripting* que no requiere compilado, esto aporta ventajas para la integración e interacción con el usuario.

Finalmente, *GNU Radio-companion GRC* es la interfaz gráfica con la que el desarrollador puede escribir una aplicación sin preocuparse por el código, esto simplifica el nivel de complejidad para el diseño de aplicaciones ya que no requiere conocimientos en lenguajes de programación.

En cuanto a las librerías. En el Anexo A. Las Tablas 14 y 15 muestran la manera en que se agrupan las distintas librerías para la conformación de *GNU Radio*. Estos archivos se reúnen en módulos con base en la función que desempeñan.

A su vez, los módulos de *GNU Radio* son estructurados en carpetas, las cuales se encargan de reunir las librerías y archivos mencionados. La Figura 11 muestra la

estructura que presenta un módulo *GNU Radio* con sus respectivos componentes. La Tabla 16 en el Anexo A. incluye información detallada acera de la función específica de cada carpeta.

Figura 11: Estructura de un Módulo GNU Radio.

Se debe mencionar que cada carpeta (Incluso la carpeta raíz) posee un archivo *CMakelist.txt* conteniendo la información necesaria para la configuración de los bloques.

Al ser una herramienta de código abierto, en *GNU Radio* es posible realizar modificaciones al *software*, incluso es posible agregar módulos personalizados al proyecto existente. Estos nuevos módulos no forman parte del núcleo de GNU Radio, son instalados fuera del directorio raíz, y por esta razón son denominados *out-of-tree*.

Scripts como *read_tipeofdata_binary.m* permiten leer el contenido de los sumideros de datos para llevar un control del flujo de datos y determinar la función que se encuentra desempeñando cada bloque de procesado. Este *script* trabaja en conjunto con programas como *Octave* y/o *Matlab* dando muestra de la flexibilidad de *GNU Radio* al operar con varias plataformas *software* tanto de código abierto como propietario.

3.3.3 *GNU Radio-companion GRC*

GNU Radio-companion surge como alternativa a la programación por línea de comandos en lenguaje *Python*. *GRC* es una interfaz que integra facultades de programación visual bastante similar a *Simulink* de *Matlab*, genera el código *Python* de la aplicación de forma automática, permitiendo de esta manera modificar directamente el código con herramientas visuales.

El código generado por *GRC* es mucho menos legible que uno escrito a mano, también puede contener algunas líneas extras de poca o ninguna relación con la aplicación desarrollada, esto se debe a que *GRC* al generar el código *Python* desde una interfaz gráfica puede darse el caso de que importe librerías extras que no se utilicen en el proceso en cuestión.

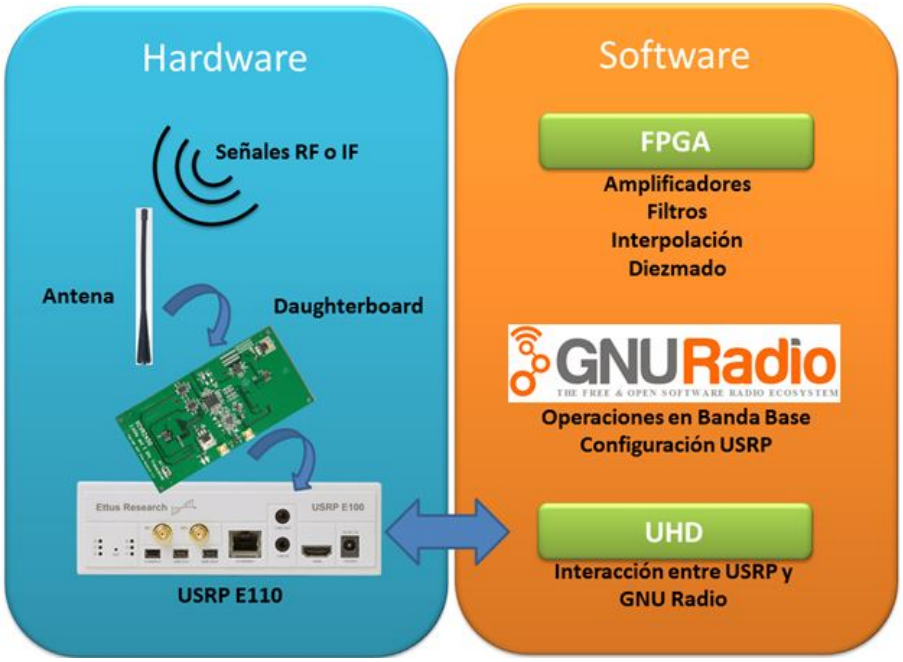


Figura 12: Resumen de interrelación *software* y *hardware* entre elementos de un sistema SDR.

En función del *hardware* utilizado, puede suceder que mientras se está ejecutando

el programa surjan algunas anomalías en cuando a sobreprocesamiento e infraprocesamiento, dichas anomalías se presentarán en la pantalla mientras el programa se ejecuta, y responden a los siguientes códigos de error.

- "u" : USRP
- "a" : audio (tarjeta de sonido)
- "O" : *Overrun* (el dispositivo no puede mantenerse con los datos recibidos desde el *USRP* o la tarjeta de audio)
- "aUaU" : audio *Underrun* (Número de muestras listas insuficientes para ser enviadas al *USRP Sink*)
- "S" : Indica una secuencia de error de números en los paquetes *Ethernet* desde el *USRP* hacia el *PC* (Situación similar a "O")

En conclusión, El USRP junto con la antena y la tarjeta *Daughterboard* se encargan de realizar operaciones de tratamiento de la señal cuando esta se encuentra en *RF* o en *IF*. *GNU Radio* se encarga de realizar operaciones en banda base, configuración y en conjunto con el *driver UHD* establecen la interfaz con el *USRP*. La información tratada por *GNU Radio* es conducida hacia la *FPGA* que se encarga de operaciones como el diezmado, interpolación y configuración de elementos de la cadena de *RF* como filtros y mezcladores. La Figura 3.3.3 ilustra los procesos de los que se encarga cada elemento que compone un sistema *SDR* que utilize *USRP* y *GNU Radio*.

CAPITULO 4

DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

En este Capítulo se detallará en aspectos que conciernen exclusivamente al desarrollo del sistema de comunicaciones utilizando dispositivos *USRP E110*. Físicamente el sistema de comunicaciones se encuentra conformado por los elementos mostrados en la Figura 13.

Figura 13: Componentes físicos del sistema de comunicaciones.

El proceso de diseño e implementación constó de las etapas que se listan a continuación:

- **Caracterización de antenas y tarjetas hijas.**
- **Diseño del transmisor.**
- **Diseño del receptor.**
- **Simulaciones de diseño:** Pruebas de funcionamiento del sistema con el uso de un bloque de modelamiento de canal en *GNU Radio*.
- **Implementación del sistema de comunicaciones**: Proceso de embebido del sistema dentro de los dispositivos *USRP E110*.

4.1 Aspectos Técnicos

Dado que el presente proyecto contempla la transmisión y recepción de voz en tiempo real, hemos de tomar en cuenta algunos aspectos que conllevan el diseño e implementación del sistema en general, así como una descripción a breves rasgos de las operaciones que realiza el sistema.

4.1.1 Descripción de un Sistema de Comunicaciones

En la sección del transmisor, la información original recogida por el micrófono es analógica y debe ser digitalizada mediante técnicas de cuantización para lograr una representación binaria de dicha información. Una vez que la información que contiene las muestras de voz se encuentre en formato binario, el *USRP* transmisor deberá procesar digitalmente esta información y convertirla en formas de onda electromagnéticas que son definidas por sus características físicas, tales como la amplitud de la señal, frecuencia de portadora y fase.

El otro terminal del enlace de comunicaciones es el *USRP* receptor. Este se encuentra encargado de identificar las características físicas de la forma de onda receptada que ha sido transmitida a través de un canal ruidoso y lleno de distorsiones, y por último retornar la señal receptada dentro de la correcta representación binaria.

La Figura 14 muestra un diagrama de bloques que constituye un sistema de comunicaciones digitales. Al hablar de una implementación basada en *SDR* los componentes programables pueden ser manipulados por *software* o con lógica programable.

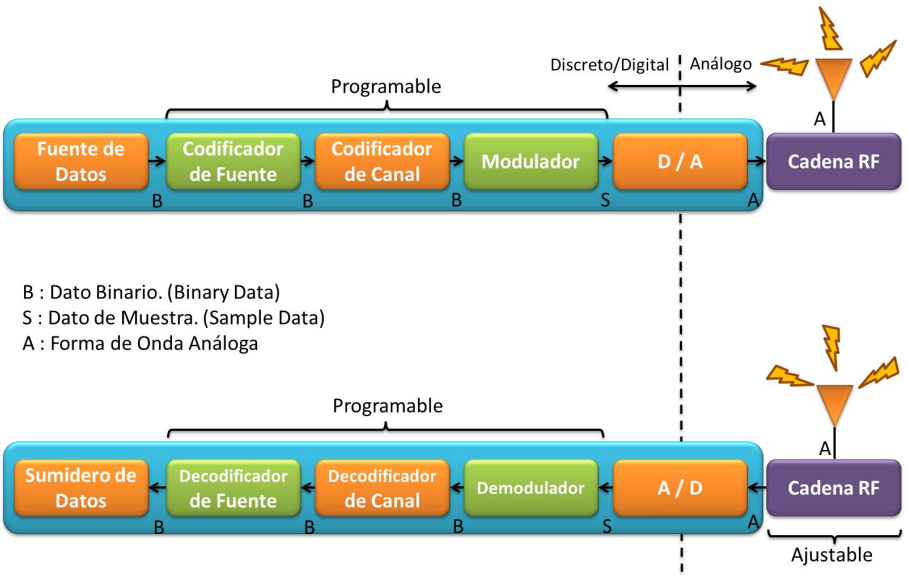


Figura 14: Diagrama de bloques de un sistema de comunicaciones.

En el diagrama de la Figura 14 puede verse que tanto la entrada al transmisor como la salida del receptor se originan desde un bloque *digital source* y son alimentados a un bloque *digital sink* en la sección del receptor. Estos dos bloques representan la fuente y destino de la información digital entre el transmisor y receptor.

Una vez que la información binaria es ingresada hacia el transmisor, la primera tarea es retirar todos los patrones binarios redundantes de la información para elevar la eficiencia de la transmisión, esto se realiza con el bloque de codificación de fuente. En

el receptor, el bloque decodificador de fuente vuelve a introducir la redundancia para regresar la información binaria de vuelta a su forma original.

Cuando la información binaria redundante en el transmisor ha sido removida con el bloque codificador de fuente se procede a usar un bloque de codificación de canal, el cual se emplea para introducir cierta cantidad controlada de información redundante para protegerla de errores potenciales que puedan introducirse durante el proceso de transmisión en un medio inalámbrico, redundancia que luego será retirada para regresar la información a su forma original.

El paso siguiente es convertir la información binaria en formas de onda electromagnéticas con ciertas propiedades como amplitud, frecuencia de portadora y fase. Esto se hace con un bloque de modulación, el cual también tiene su proceso inverso en el receptor.

Finalmente las muestras discretas que salen del bloque de modulación son remuestreadas y convertidas a formas de onda analógicas en banda base usando un convertidor *D/A* (Digital a Analógico) antes de ser procesadas por el *Front-End* de *RF* y ser subidas a la frecuencia de la portadora.

Para la realización del diseño presentado en este documento se trabajó en la implementación de un sistema de comunicación de voz utilizando *codecs* de voz basados en la recomendación de la *ITU G.721* (VoipForo, 2013b), la modulación escogida es *GMSK* debido a que combina cualidades de gran calidad de voz y baja carga computacional.

4.1.2 Consideraciones de Diseño

Cuando se realiza el diseño de un sistema de comunicaciones con base en *SDR* y sus respectivos componentes, es importante entender las limitaciones de cada plataforma

y de qué manera varios parámetros de diseño pueden afectar el desempeño del modelo resultante.

Uno de los objetivos del presente proyecto es lograr capacidades de procesamiento en tiempo real para una transmisión fácil con alto nivel en la calidad de voz y optimizando los recursos ofrecidos por la plataforma embebida elegida. Sin embargo, si el microprocesador empleado por la plataforma *SDR* no es lo suficientemente poderoso como para soportar las operaciones requeridas por el sistema de comunicación digital (dispositivos embebidos), debe reconsiderarse el diseño del modelo completo y los requerimientos del sistema en general, ya que de otra manera la implementación del sistema basado en *SDR* no funcionará del todo, ya sea con un pobre desempeño o mediante el anuncio de alertas y errores correspondientes a los procesos computacionales requeridos.

En resumen, las consideraciones a tomar en cuenta cuando se desarrollan sistemas de comunicaciones basadas en plataformas *SDR* según (*Digital Communication Systems Engineering with Software-Defined Radio*, 2013) incluyen lo siguiente:

- La integración entre la capa física y el *software* implementado en un procesador embebido requiere una óptima interdependencia entre todas las capas cuando se precisa procesamiento en tiempo real.

- Debe asegurarse que exista el suficiente ancho de banda para la transmisión, por lo que es importante conocer las capacidades del *hardware* con respecto a estos atributos físicos.

- Muchos sistemas de comunicación digital poseen una arquitectura centralizada para controlar las operaciones sobre todo el sistema. El conocer la arquitectura de la plataforma *SDR* es importante para determinar que tipo de operaciones deben realizarse para comunicar un dispositivo con otro.

- La habilidad de realizar experimentos controlados en distintos ambientes (multiobstáculos, multicaminos, etc.) es importante para el propósito de demostrar la confiabilidad de una implementación *SDR* en particular.

- Capacidad de reconfiguración y rápida adaptación por medio de un flujo en el diseño de *software* para la descripción de algoritmos y protocolos también es necesario.

Dado que el sistema debe trabajar con un micrófono conectado al puerto de audio de entrada del *USRP* han de tomarse en cuenta ciertas consideraciones con respecto a la amplitud de la señal de entrada, los valores de la amplitud impacta directamente en el voltaje de la señal enviada hacia el conversor *DAC* del *USRP* que trabaja en rangos desde -1V a +1V, por lo tanto un valor que exceda a esta magnitud saturará al *DAC*. Por otro lado, valores con una magnitud inferior (pero próxima) a este rango podría ocasionar que la señal se comprima y trabaje en regiones no lineales.

4.1.3 Requerimientos de *Software*

Para la realización del diseño se hará uso de la herramienta *GNU Radio-companion* cuyas características y relaciones de codependencia con otras plataformas *software* han sido ampliamente revisadas y discutidas en el Capítulo 3. del presente documento. Se recomienda utilizar *GNU Radio*, versión estable 3.6, la cual contiene bibliotecas de bloques de procesamiento de señales actualizados y libres de *bugs*.

Como sistema operativo ha sido usado *Angstrom Distribution* debido a que posee el mejor soporte de dependencias (requerimientos, capacidades y relación entre elementos de *software*) para poder posteriormente configurar, compilar e instalar *GNU radio*. Esta distribución de *Linux* se encuentra embebida en la tarjeta *SD* de los equipos *USRP E110*.

4.1.4 Requerimientos de *Hardware*

Los modelos *USRP E110* cuentan con un procesador de 800 MHz *ARM Cortex A8* y 512MB *RAM*. Son dispositivos *SDR* embebidos, los cuales pueden establecer comunicación vía *SSH* o haciendo uso del puerto de consola integrado en el dispositivo. Las características técnicas de estos dispositivos han sido detallados en el Capítulo 3. del presente documento.

En su mayoría, el diseño de este sistema de comunicaciones fue desarrollado conectando los dispositivos a un televisor mediante los puertos *HDMI* que integran los *USRP*, para la conexión de dispositivos periféricos se utilizó un teclado y *mouse USB* conectados mediante un *HUB* al puerto *host* de los *USRP*. Cuando fue necesario la comunicación por consola se necesitó como mínimo un computador portátil con procesador *Intel Atom* de 1.5 GHz, 2 GB de memoria *RAM* y puerto *USB*-2.0. La conexión por consola fue establecida con un sistema operativo nativo, no se experimentó realizar la comunicación mediante máquinas virtuales.

Tanto el subsistema de transmisión como el de recepción requerirán el uso de una antena *Dual VERT900* la cual funciona en las bandas de 824 MHz a 960 MHz y 1710 Mhz a 1990 MHz, se trata de una antena vertical omnidireccional con 3 dBi de ganancia. El subsistema de transmisión tendrá conectado al puerto de entrada de audio un micrófono activo para la toma de señales de voz, mientras que el subsistema de recepción estará dotado de un par de parlantes estándar de computador o en su defecto un set de auriculares de casco.

4.2 Códec de voz

Un códec de audio se conforma de un conjunto de algoritmos que permiten codificar y decodificar los datos sonoros, lo cual significa reducir la cantidad de bits que ocupa

el fichero de audio. Sirve para comprimir señales de audio con el propósito de ocupar el menor espacio posible, consiguiendo una buena calidad final, y descomprimiéndolos para reproducirlos o manipularlos en un formato más apropiado. Las funciones realizadas por el codificador de voz corresponden a los procesos de codificación de fuente.

Los vocoders implementan algoritmos paramétricos específicos para la codificación de la voz. Estos analizan la señal de voz correspondiente a un segmento temporal considerado estacionario para extraer los parámetros del modelo y la excitación. Esta información es la que se codifica. En el proceso de decodificación, el decodificador sintetiza los parámetros a través de un modelo de producción de voz (Valero, 2013).

4.2.1 Teoría del Vocoder

Como ya se ha comentado la comunicación de voz es analógica, mientras que el procesamiento de datos es digital. La transformación de la señal analógica a una señal digital se realiza mediante una conversión analógico-digital. Este proceso de conversión analógico digital o modulación por pulsos codificados (PCM) se realiza en tres etapas:

- Muestreo (*sampling*)
- Cuantificación (*quantization*)
- Codificación (*codification*)

4.2.2 Muestreo

El proceso de muestreo consiste en tomar valores instantáneos de una señal analógica, a intervalos de tiempo iguales. Los valores instantáneos obtenidos son llamados muestras. Este proceso es explicado en la Figura 15:

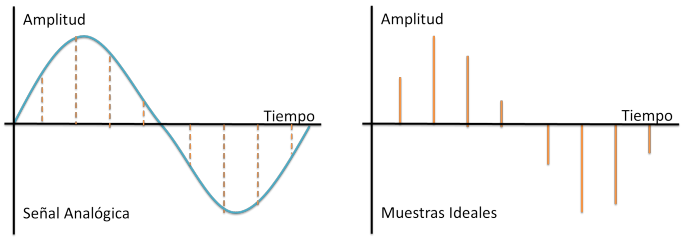


Figura 15: Proceso de muestreo de una señal analógica.

El muestreo se efectúa siempre a un ritmo uniforme, que viene dado por la frecuencia de muestreo o *sampling rate*. La condición que debe cumplir la frecuencia de muestreo viene dada por el teorema del muestreo:

"Si la frecuencia más alta contenida en una señal analógica $x_a(t)$ *es* $F_{max} = B$ *y la señal se muestrea a una velocidad* $Fs > 2F_{max}$*, entonces* $x_a(t)$ *se puede recuperar totalmente de sus muestras mediante la siguiente función de interpolación:.*

$$g(t) = \frac{sin(2\pi Bt)}{2\pi Bt} \tag{4.1}$$

Así, $x_a(t)$ *se puede expresar como:*

$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a\left(\frac{n}{Fs}\right) g\left(t - \frac{n}{Fs}\right) \tag{4.2}$$

donde $x_a\left(\frac{n}{Fs}\right) = x_a(nT) \equiv x(n)$ *son las muestras de* $x_a(t)$".

Entonces, una señal que contiene únicamente frecuencias inferiores a F, queda completamente determinada por muestras tomadas a una velocidad igual o superior al doble de esta frecuencia, esta tasa de muestreo se denomina teorema de *Nyquist* (LPI, 2014).

De acuerdo con el teorema del muestreo, las señales que se encuentran dentro del rango de frecuencia vocal, las cuales ocupan la banda de 500 Hz a 3500 Hz, han de ser muestreadas a una frecuencia igual o superior a 7000 Hz (2×3500). En la práctica, sin

embargo, se suele tomar una frecuencia de muestreo (Fm) de 8000 Hz en adelante. Es decir, se toman 8000 muestras por segundo que corresponden a una separación entre muestras de:

$$T = \frac{1}{Fm} = \frac{1}{8000} = 0.000125s. = 125\mu s, \tag{4.3}$$

Donde T es la duración de una muestra de la señal de voz. Por lo tanto, dos muestras consecutivas de una misma señal están separadas 125 μs que es el período de muestreo. En este punto cabe aclarar que si bien es cierto las frecuencias producidas por el aparato fonador humano van desde los 500 Hz a los 3500 Hz el espectro audible para un aparato auditivo joven y sano abarca frecuencias que van desde los 20 Hz hasta los 20 kHz, es por ello que el ser humano está en capacidades de escuchar frecuencias mucho más altas o bajas provenientes de fuentes sonoras distintas al aparato fonador humano, como los instrumentos musicales (Herriko, 2003).

4.2.3 Cuantificación

La cuantificación es el proceso mediante el cual se asignan valores discretos, a las amplitudes de las muestras obtenidas en el proceso de muestreo. Existen varias formas de cuantificación en función de su complejidad.

- Cuantificación uniforme.
- Cuantificación no uniforme (Adrián de Pérez, 2003).

El diseño realizado hace uso de una cuantificación uniforme en la etapa de muestreo de voz y una cuantificación no uniforme mediante un códec (compresor - decompresor), conforme a lo que se muestra en la Figura 16.

El proceso de cuantificación no uniforme responde a una característica determinada llamada ley de Codificación o de compresión. Existen dos tipos de leyes de

codificación: las continuas y las de segmentos.

En la ley de codificación continua, los intervalos de cuantificación son todos de amplitud distinta, creciendo ordenadamente desde valores muy pequeños, correspondientes a las señales de nivel bajo, a valores grandes, correspondientes a las señales de nivel alto.

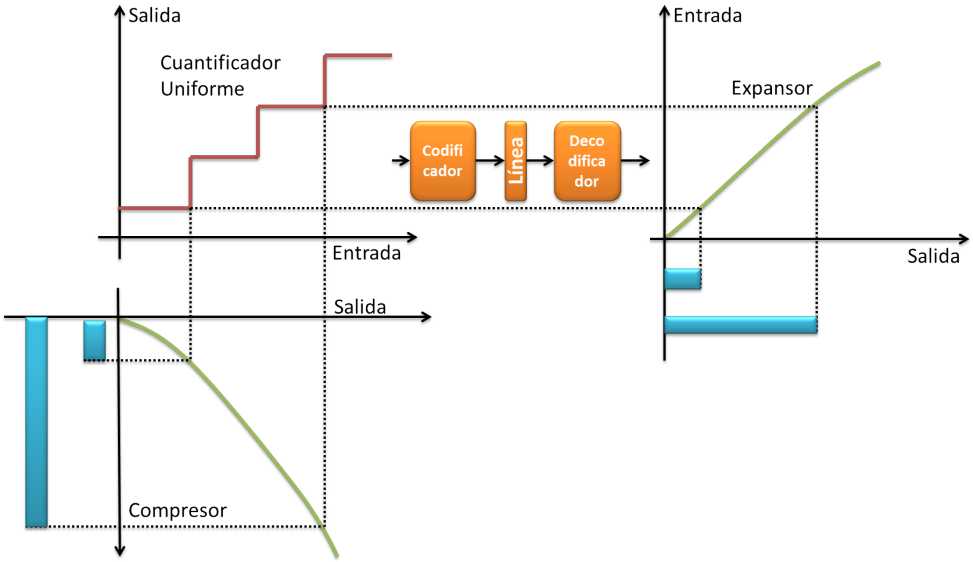


Figura 16: Cuantificación no uniforme utilizando códec. (VoipForo, 2013a)

En la ley de codificación de segmentos la gama de funcionamiento se divide en un número determinado de grupos y dentro de cada grupo los intervalos de cuantificación tienen la misma amplitud, siendo distinta de unos grupos a otros. En un proceso de cuantificación no uniforme generalmente se utilizan las leyes de codificación de segmentos.

4.2.4 Funcionamiento de un vocoder

Un vocoder comprime ciertos valores tomados de las muestras de voz en la frecuencia fundamental utilizando filtros, estos valores entonces se modifican con el conocimiento de que el habla humana varía típicamente entre ciertos valores de frecuencias (500 a

3500 Mhz). El resultado es habla inteligible, aunque con ciertas características de voz mecánica o metalizada. Los vocoders a menudo incluyen también un sistema para generar sonidos no sonoros, usando un segundo sistema consistente en un generador de ruido en lugar de la frecuencia fundamental de la voz.

Las técnicas de compresión de audio tienden a restar calidad respecto a la señal de audio original no comprimida, en general si la compresión es buena la percepción de esta disminución de calidad es poco perceptible. Cuando se habla de técnicas de compresión de voz debe tenerse en cuenta que a mayor compresión la calidad se ve perjudicada.

Cada algoritmo de compresión difiere de los demás; las técnicas más complejas obtienen una mayor calidad en la señal, pero a su vez hacen uso de una mayor cantidad de recursos de procesamiento.

Al realizar el diseño de comunicaciones presentado en este documento hemos tenido en cuenta que la comunicación es muy sensible a los retardos y al *jitter* (variabilidad temporal durante el envío de señales digitales), por lo tanto, los algoritmos de compresión diseñados deben introducir un retardo muy pequeño ya que de lo contrario la característica de tiempo real se perdería, además podrían aparecer problemas de ecos y baja calidad de audio. Esta restricción limita las posibilidades de los algoritmos empleados para el diseño. Existen otros algoritmos de compresión de audio que al no tomar en cuenta estas restricciones permiten unas tasas de compresión más elevadas, pudiendo analizar muestras de audio más extensas y emplear un mayor tiempo en su análisis, introduciendo por tanto mayor retardo y *jitter* además de requerir una capacidad de procesamiento mayor que otros algoritmos de compresión.

Se ha seleccionado el códec G.721 como algoritmo de compresión de la señal de voz. Este códec, también conocido como *ADPCM* (*Adaptive Differential Pulse Code*

Modulation), es un códec de forma de onda, por lo que en teoría sirve para todo tipo de señales, aunque su comportamiento es notoriamente mejor para señales de voz. Se muestrea a 8 kHz, por lo que es un códec de voz de banda estrecha (*narrowband*).

En algoritmos de compresión de voz la técnica popularmente usada es predecir el valor de la siguiente muestra con el valor de las muestras anteriores, en esto se basan los esquemas de *DPCM* (*PCM* Diferencial), en los que se cuantifica la diferencia entre la señal original y la predicha. Esto es posible gracias a las correlaciones en las muestras de señales de voz debido a los efectos de la cavidad vocal y las vibraciones de las cuerdas vocales. Si las predicciones son efectivas entonces la señal de error entre las muestras predichas y las actuales muestras de la señal de voz tendrán menor variación (varianza) con relación a las muestras originales de la señal de voz. El objetivo principal se reduce en cuantificar esta señal de error con menos bits que la señal original de voz (Araseki & Ozawa, 1982).

Los resultados de los algoritmos de compresión utilizados en estos códecs se pueden mejorar si el predictor y el cuantificador se hacen adaptativos, de tal forma que puedan cambiar para adaptarse a las características de la señal que se está codificando tal y como ocurre con el códec G.721 que cuantifica estas diferencia con 4 bits, dando lugar a una tasa binaria de 32 Kbps.

La Tabla 9 ilustra algunos formatos de compresión digital de audio estandarizados por la *ITU-T* que introducen bajo retardo como resultado de los algoritmos de compresión que utilizan.

Otras razones por las que se eligió G.721 como algoritmo de compresión de voz son su frecuencia de muestreo, la cual coincide con la frecuencia que el bloque de audio toma para el muestreo de la voz desde el micrófono y por su caudal de canal de 32Kbps, que ofrece una velocidad que permite al *USRP* adaptarse a la cantidad

Tabla 9: Formatos de compresión digital de bajo retardo. (VoipForo, 2013a)

Formato	**F. muestreo** (Hz)	**Canales**	**Caudal por canal** (Kbps)	**Uso**
PCM (G.721)	8000	1	64	Telefonía
ADPCM (G.721)	8000	1	32	Telefonía
SB-ADPCM (G.722)	16000	1	48/56/64	Videoconf.
MP-MLQ (G.723.1)	8000	1	6.3/5.3	Telef/Internet
ADPCM (G.726)	8000	1	16/24/32/40	Telefonía
E-ADPCM (G.727)	8000	1	16/24/32/40	Telefonía
LD-CELP (G.728)	8000	1	16	Telef/Videoconf.
CS-ACELP (G.729)	8000	1	8	Telef/Internet
RPE-LTP (GSM 06.10)	8000	1	3.24	Telefonía GSM
CELP (FS 1016)	8000	1	4.8	
LPC-10E (FS 1015	8000	1	2.4	

de muestras que se generan como resultado de la compresión de voz con una buena calidad de sonido.

4.3 Codificación de Canal

La codificación de canal se usa con el fin de proteger la información para dotarla de inmunidad frente a errores ocasionados por el ruido. Esta codificación consiste básicamente en introducir información controlada de redundancia, de forma que sea posible reconstruir la secuencia de datos original de la forma más fiable posible cuando sucedan errores.

GNU Radio dispone de bloques para establecer codificación de canal, como por ejemplo el bloque de codificación *Trellis*. El diseño contemplado en el presente documento no incluye codificación de canal debido a la alta carga computacional que representa la inclusión de este tipo de algoritmos de redundancia y la capacidad de procesamiento del sistema embebido en el *USRP*. Al momento de incluir este bloque en el diseño, el *USRP* no fue capaz de establecer la comunicación ya que el número de muestras que se requerían para el envío de datos no eran soportadas por el *hardware*. Es por esta razón que la codificación de canal fue suprimida al momento de realizar la implementación.

4.4 Modulación

La técnica de modulación *GMSK* (*Gaussian Minimum Shift Keying*), resulta ser una modulación muy útil, ya que optimiza el uso del ancho de banda. En este esquema cada símbolo representa un bit, que se genera a cada cambio de fase, como se muestra en la Figura 17.

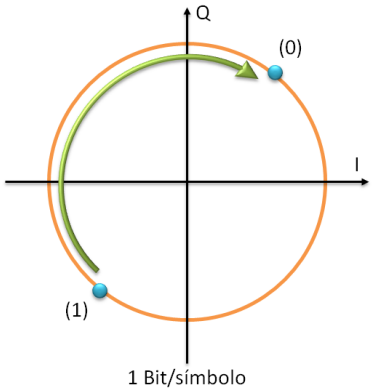


Figura 17: Esquema de modulación GMSK. (Torres Nova & Paz Penagos, 2008)

GMSK es un esquema de modulación binaria simple que se puede ver como derivado de *MSK*. En *GMSK*, los lóbulos laterales del espectro de una señal *MSK* se reducen pasando los datos a través de un filtro Gaussiano de premodulación. El filtro Gaussiano aplana la trayectoria de fase de la señal *MSK* y por lo tanto, estabiliza las variaciones de la frecuencia instantánea a través del tiempo. En la práctica, *GMSK* es muy atractiva por su excelente eficiencia de espectral y de potencia.

El filtrado convierte a cada dato modulante que ocupa en banda base un periodo de tiempo T, en un símbolo que ocupa varios periodos. Sin embargo, dado que esta conformación de pulsos no cambia el modelo de la trayectoria de la fase, *GMSK* se puede detectar coherentemente como una señal *MSK*, o no coherente como una señal simple *FSK* (Tomislav & Marijan, 2008).

Una de las principales características de la modulación *GMSK* es el parámetro *BT*, el cual se refiere al producto entre el tiempo de bit *T* y el ancho de banda del filtro a 3dB. El filtro de premodulación introduce interferencia intersimbólica en la señal transmitida, pero esta degradación no es grave si el parámetro *BT* del filtro es mayor a 0.3, lo dicho anteriormente es más sencillo de entender al ver la Figura 18.

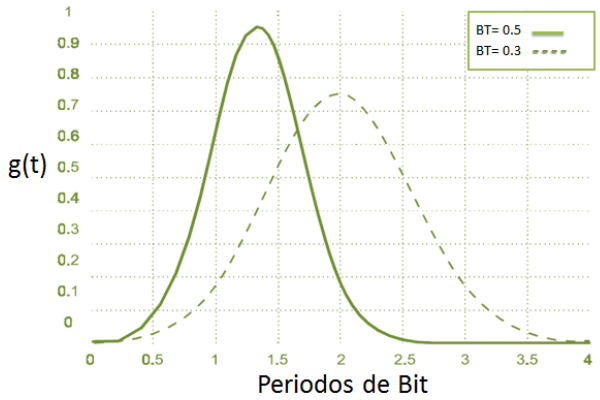


Figura 18: Periodo de bit para valores de BT de 0.3 y 0.5. (Torres Nova & Paz Penagos, 2008)

El filtro Gaussiano de pre modulación tiene una respuesta impulsiva dada por:

$$h_{G(t)} = \frac{\sqrt{\pi}}{\alpha} exp\left(-\frac{\pi^2}{\alpha^2}t^2\right) \tag{4.4}$$

Y su respuesta en frecuencia viene dada por:

$$H_G(f) = exp(-\alpha^2 f^2), \tag{4.5}$$

donde el parámetro α está relacionado con el ancho de banda del filtro B por la siguiente expresión:

$$\alpha = \frac{\sqrt{2ln(2)}}{B} \tag{4.6}$$

El filtro *GMSK* se puede definir completamente por B y por la duración de un símbolo en banda base T o equivalentemente por su producto *BT*. La Figura 4.4 muestra la *Power Spectral Density PSD* de una señal *GMSK* para varios valores de *BT* y para una señal *MSK* , que es equivalente a *GMSK* con *BT* infinito. En la Figura 4.4 se observa como a medida que se reduce el parámetro *BT*, los niveles de los lóbulos laterales se atenúan rápidamente. Por ejemplo, para *BT*=0.5, el pico del segundo lóbulo esta en más de 30 dB por debajo del principal, mientras que para *MSK* el segundo lóbulo está solo 20 dB por debajo del principal. Sin embargo, la reducción de *BT* incrementa la

ISI, y por lo tanto se incrementa la tasa de errores de bit (*Bit Error Rate*), pero a pesar de este efecto el rendimiento global del sistema mejora.

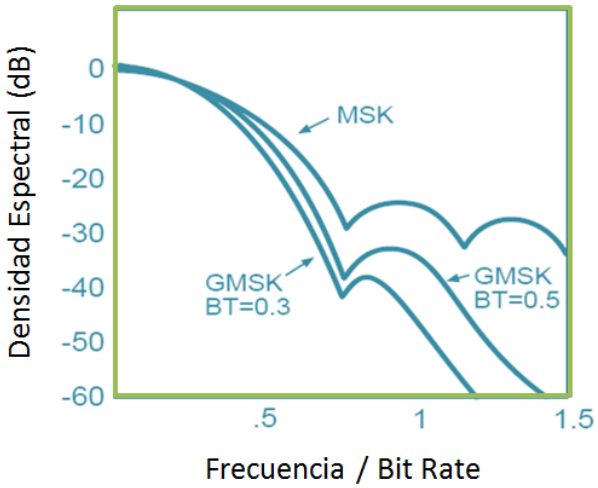


Figura 19: Densidad de Potencia Espectral de una señal GMSK para varios valores de BT. (Tomislav & Marijan, 2008)

4.5 Transmisor

En esta sección se explicará con detalle el diseño y configuración de cada bloque de procesamiento que conforma el subsistema de transmisión, el cual está compuesto por los elementos mostrados en la Figura 20.

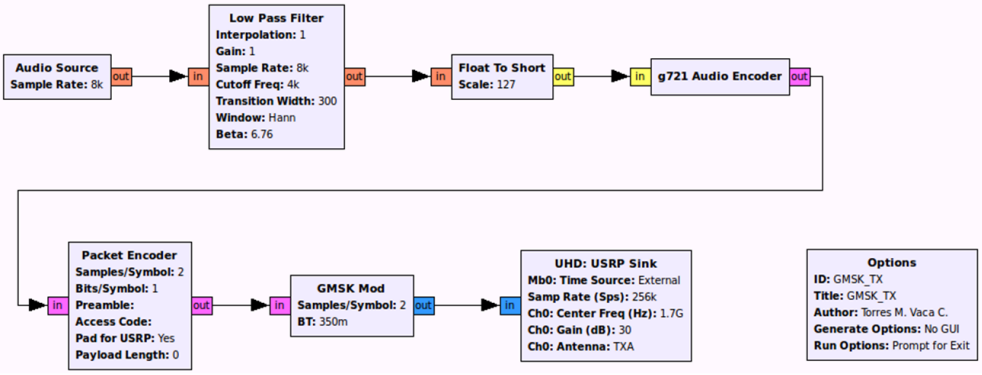


Figura 20: Subsistema de transmisión.

4.5.1 Fuente de Audio

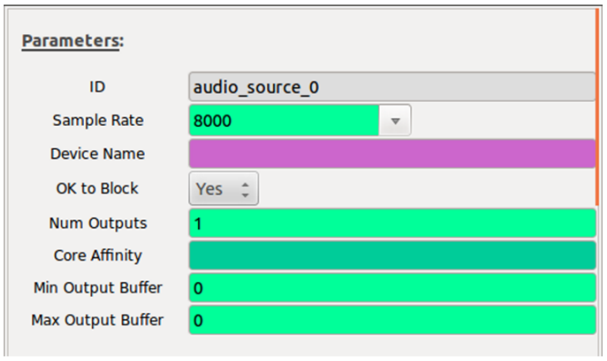


Figura 21: Bloque Audio Source en GNU Radio.

El bloque de audio es capaz de capturar señales de audio desde la tarjeta de sonido del dispositivo *USRP* específicamente desde la entrada del micrófono, a distintas tasas de muestreo desde 8 kHz hasta 44.1 kHz. Para el diseño realizado se ha escogido una tasa de muestreo de 8 kHz debido a que esta tasa de muestreo combina una resolución aceptable para la cuantificación de la señal y una carga computacional para el *USRP*. La salida de este bloque son muestras de punto flotante de 8 bits cuyos valores se encuentran en el rango de -1 a 1. La mayoría de convertidores A/D de las tarjetas de sonido tienen salidas enteras de estas características, la razón por la que este bloque entrega valores de punto flotante es por compatibilidad con otros bloques de *GNU Radio*.

La duración de cada muestra es 125 uS, tal y como puede verse en la ecuación 5.12. Por lo que la tasa de muestreo del bloque de audio viene dado por:

$$Rs = \frac{bits}{tiempo} = \frac{8bits}{125us} = 64kbps \tag{4.7}$$

4.5.2 Filtro Pasa-Bajos

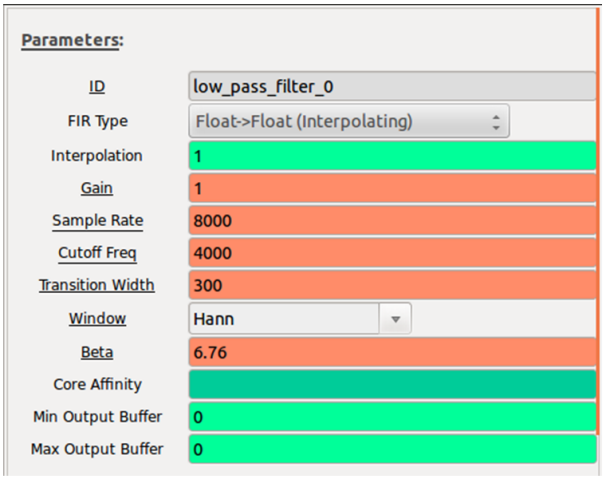


Figura 22: Bloque Filtro Pasa-Bajos en GNU Radio.

El bloque representa un filtro pasa bajos, la tasa de muestreo ha sido fijada en 8 kHz al igual que en el bloque de audio. Debido a que el propósito del sistema de comunicaciones es obtener su mejor desempeño con señales de voz humana la frecuencia de corte del filtro se fija en 4 kHz con un ancho de transición de 300 Hz. La ventana escogida es la de *Hann* debido a su excelente discriminación de señales en filtros pasa bajos (Wurzburg, 2013).

4.5.3 Conversor de Tipo de Dato *Float-to-Short*

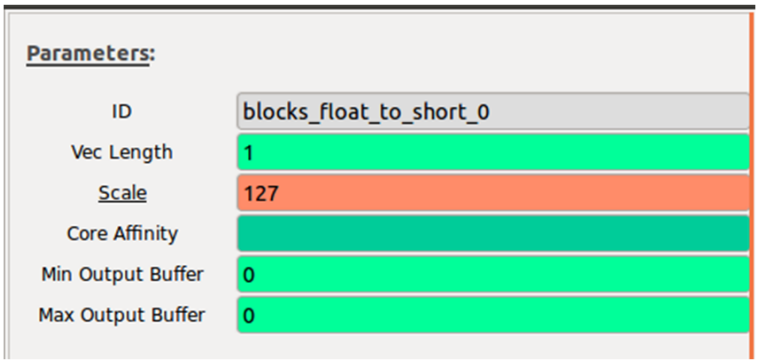


Figura 23: Bloque Float-to-Short en GNU Radio.

En esta sección se agrega un bloque que permite realizar la conversión de un dato tipo punto flotante a un dato de tipo *short* útil al momento de emplearlo junto a un *vocoder*. Se ha escogido una escala de 127 niveles de cuantización que representan 8 bits (7 bits para nivel más 1 bit de signo, recordemos que la señal es cuantizada en valores desde -1 a 1).

4.5.4 Codificador G.721

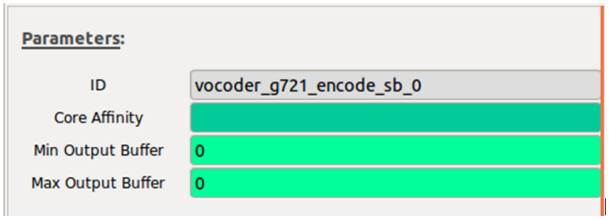


Figura 24: Bloque Codificador G.721 en GNU Radio.

Una vez convertida la señal de tipo punto flotante a tipo *short*, se procede a enviar esta información para ser tratada con algoritmos de compresión de voz realizados por el bloque *vocoder G.721* obteniendo a su salida un dato de tipo *byte*.

La tasa de salida de este bloque es de 32 kbps en función del estándar G.721 utilizado (VoipForo, 2013b), y dado que la tasa de muestreo a la entrada de este bloque es 64 kbps podemos afirmar que el factor de compresión utilizado por el bloque *vocoder* es $1/2$.

4.5.5 *Packet Encoder*

Este bloque permite realizar el empaquetamiento de los datos a fin de que sean correctamente ingresados al bloque modulador. Se ha fijado los valores en 2 muestras por símbolo y 1 bit por símbolo para la modulación *GMSK*.

El bloque *Packet Encoder* requiere un cierto valor de bits por símbolo en función de la modulación que se desee realizar, por ejemplo, si se deseara implementar una modulación 16 QAM harían falta 4 bits por símbolo, dado que $2^4 = 16$. La Tabla 10 muestra el valor de bits por símbolo a configurar en función del esquema de modulación que se desee implementar.

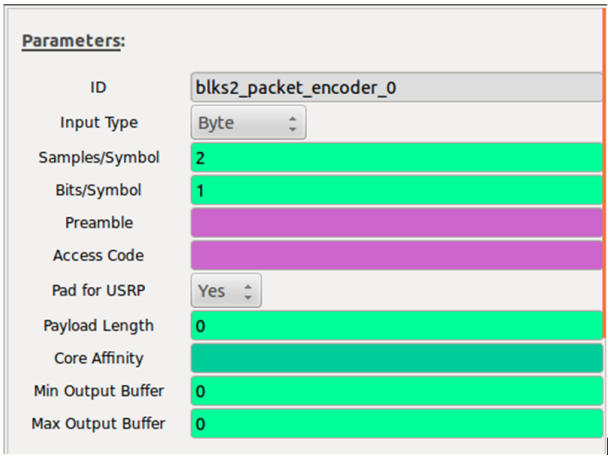


Figura 25: Bloque Packet Encoder en GNU Radio.

El codificador de paquetes convierte el flujo entrante de muestras en paquetes de bits también llamados *Chunks*. El número de bits en cada bloque se puede ajustar para que coincida con el siguiente bloque; el modulador. Un modulador *GMSK*, por ejemplo, modula un bit en cada símbolo. Esto significa que cada bit en las muestras entrantes de ocho bits deben ser enviados un bit a la vez. Algo importante a tomar en cuenta es que este proceso cambia la frecuencia de muestreo desde la entrada del bloque hacia la salida.

Tabla 10: Formatos de compresión digital de bajo retardo.

	GMSK	DBPSK	DQPSK	D8PSK	8QAM	16QAM	64QAM	256QAM
1bit	X	X						
2bit			X					
3bit				X	X			
4bit						X		
6bit							X	
8bit								X

4.5.6 Modulador GMSK

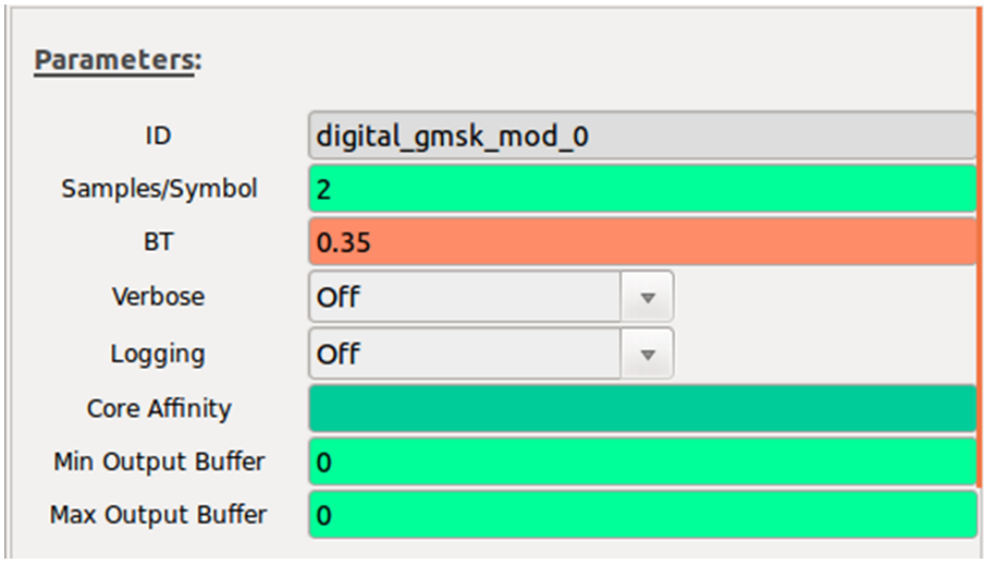


Figura 26: Bloque Modulador GMSK en GNU Radio.

El bloque correspondiente al modulador *GMSK* permite configurar sus valores de acuerdo al empaquetamiento realizado anteriormente, esto quiere decir que debe existir concordancia en el número de muestras por símbolo del bloque *Packet Encoder* con el número de muestras por símbolo del bloque modulador, que para el caso de este diseño es de dos. De no concordar ambos números no existe un mensaje de error o advertencia que nos informe sobre este acontecimiento sin embargo la transmisión de

la señal no se realizará de forma correcta. El valor de *BT* elegido es 0.35, las razones de la elección de este valor se detallan en la sección referente a la modulación del presente Capítulo.

4.5.7 Interfaz UHD USRP *Sink*

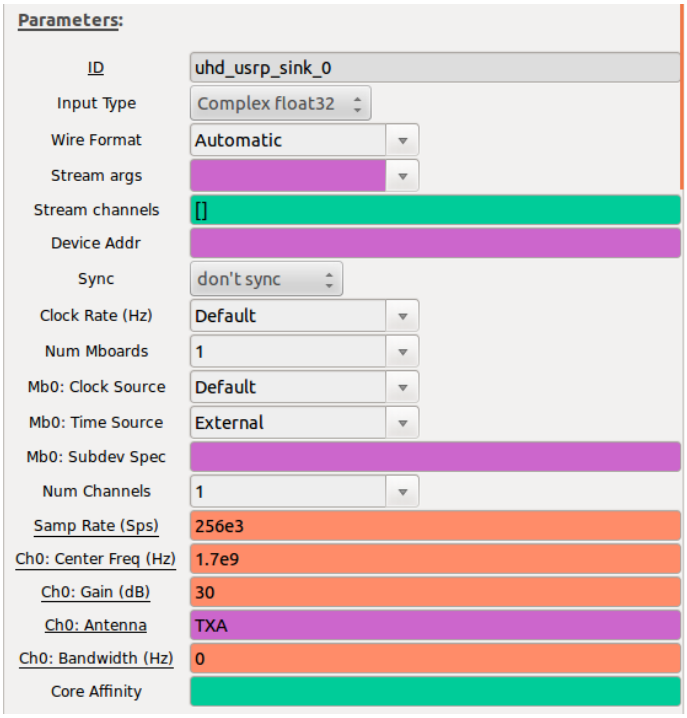


Figura 27: Bloque USRP UHD Sink en GNU Radio.

Para finalizar el diseño referente al subsistema de transmisión se hablará del bloque de interfaz provisto por *GNU Radio*, el *UHD USRP Sink*. Este bloque permite establecer el vínculo del *software* con el dispositivo embebido. La tasa de muestreo ha sido fijada en 256 kHz, la frecuencia central en 1.7 GHz, una ganancia de 30 dB, la explicación acerca de los valores establecidos para la ganancia se detallarán en el Capítulo 5. del presente documento, en la sección correspondiente a la potencia de transmisión y recepción. La antena se encuentra colocada en la *Daughterboard* en el puerto Tx/Rx, por ello se ha escrito TX/RX en la elección del canal de la antena. La tasa de muestreo

T_m para este bloque se calcula con la expresión:

$$T_m = Tasa_{Fuente\ audio} \times Float_{to}Short \times GMSK_{Mod(muestra)} \times GMSK_{Mod(símbolo)} \quad (4.8)$$

$$T_m = 8kHz \times 8\frac{bit}{muestra} \times 2\frac{muestra}{símbolo} \times 1\frac{símbolo}{bit} \quad (4.9)$$

$$T_m = 128kHz \quad (4.10)$$

Para asegurar el cumplimiento de la frecuencia de *Nyquist* se ha establecido un valor de frecuencia de muestreo equivalente al doble del valor obtenido, es decir 256 kHz.

4.6 Receptor

El subsistema receptor consiste básicamente en el proceso inverso llevado a cabo en el transmisor, y se compone de los bloques de procesamiento mostrados en la Figura 28

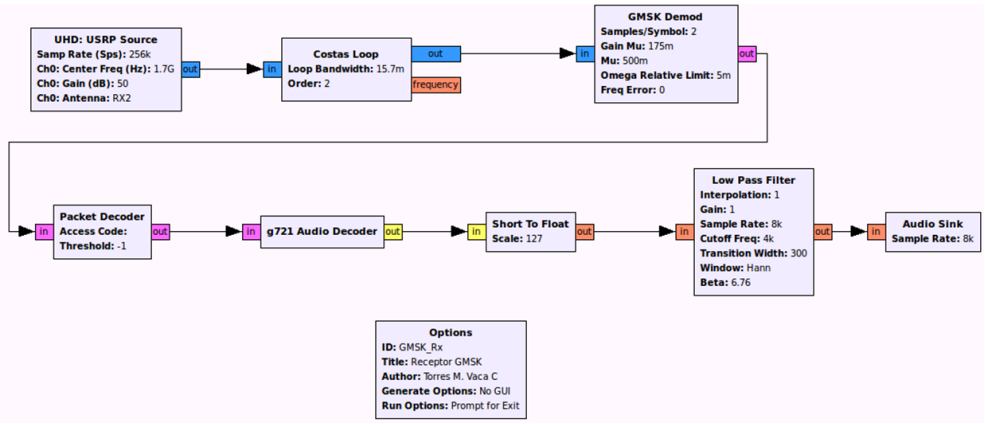


Figura 28: Subsistema de recepción.

4.6.1 Interfaz UHD USRP *Source*

Este bloque al igual que su contraparte sumidero establece una interfaz con el *USRP*, pero esta vez realiza funciones de recepción de muestras, este bloque actúa como fuente de datos del subsistema de recepción. Se ha configurado el bloque con los mismos parámetros que en el transmisor, con la única diferencia de ganancia establecida en 50 dB y el puerto de antena será el RX2.

Para la realización de pruebas de simulación tanto este bloque como el bloque *UHD USRP Sink* son omitidos, puesto que en un entorno de simulación no se establece comunicación con los elementos físicos del *USRP*.

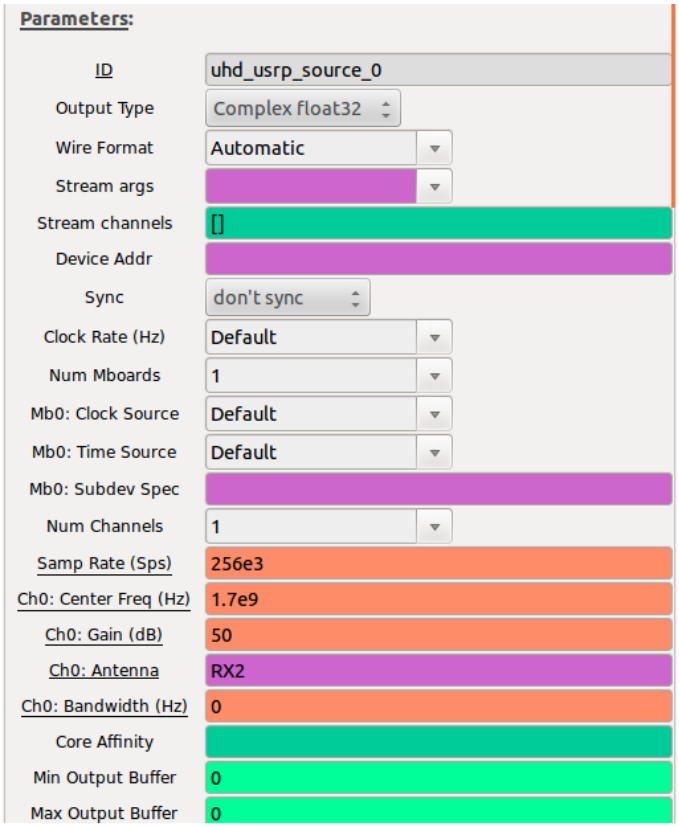


Figura 29: Bloque USRP UHD Source en GNU Radio.

4.6.2 *Costas Loop*

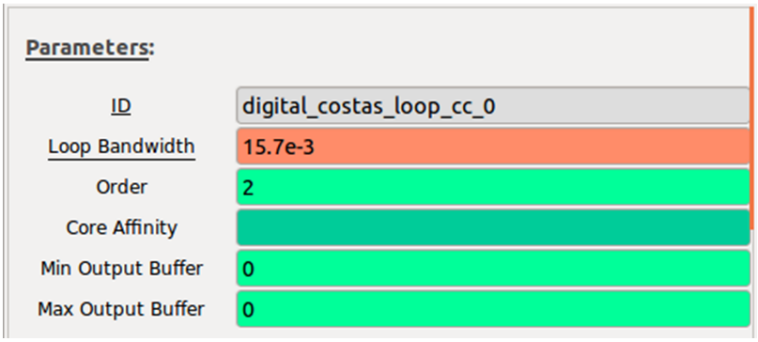


Figura 30: Bloque Costas Loop en GNU Radio.

Este bloque permite la detección de señales en recepción cuando la portadora no es recibida, a fin de enganchar el receptor al transmisor. Este proceso es realizado para la recuperación de portadora. Para el diseño se ha fijado un lazo de segundo orden para modulaciones digitales *GMSK* con un ancho del lazo recomendado de $\frac{\pi/2}{100}$ (Jeff, 2002).

4.6.3 Demodulador GMSK

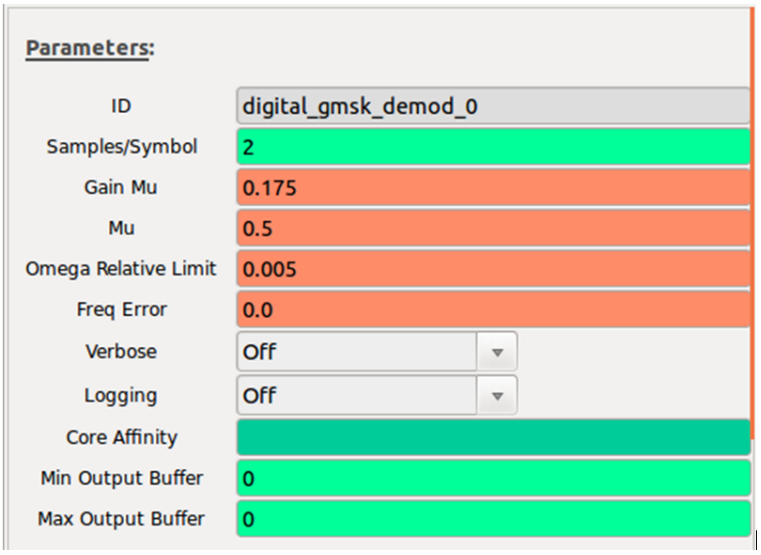


Figura 31: Bloque Demodulador GMSK en GNU Radio.

Se encarga de la demodulación de la señal GMSK. Se trata del proceso inverso llevado a cabo en el subsistema transmisor. Este bloque constituye un receptor coherente puesto que tiene conocimiento de la frecuencia y el esquema de modulación empleada por el Tx.

4.6.4 *Packet Decoder*

Se encarga de realizar exactamente las operaciones inversas al bloque *Packet Encoder* descrito en la sección del transmisor.

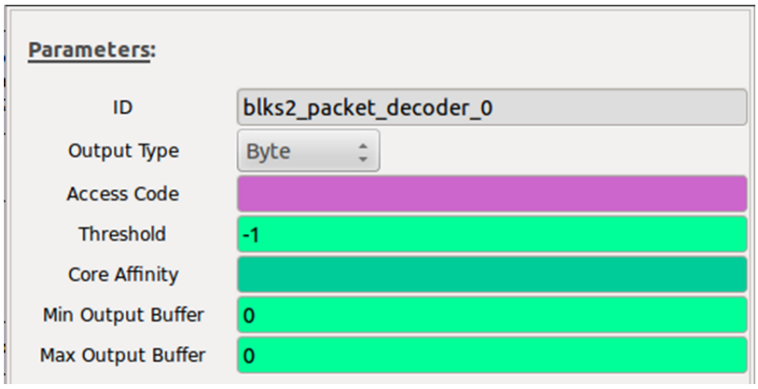


Figura 32: Bloque USRP Packet Decoder en GNU Radio.

4.6.5 Decodificador G.721

Decodifica los datos de acuerdo al estándar G.721. La salida de este bloque entrega datos de tipo *short*.

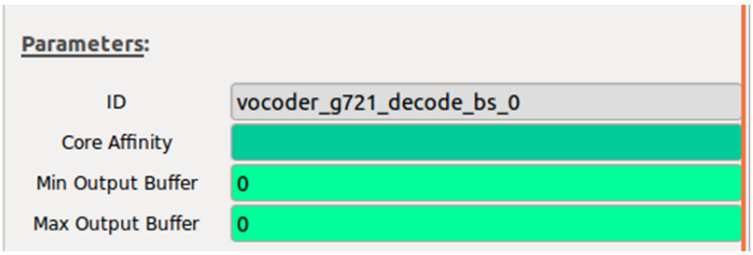


Figura 33: Bloque Decodificador G.721 en GNU Radio.

4.6.6 Conversor de tipo de dato *Short-to-Float*

Este bloque realiza la conversión de un dato tipo *short* a un dato de tipo punto flotante. Para este bloque se ha elegido una escala de 127 niveles de cuantización al igual que su contraparte en el transmisor.

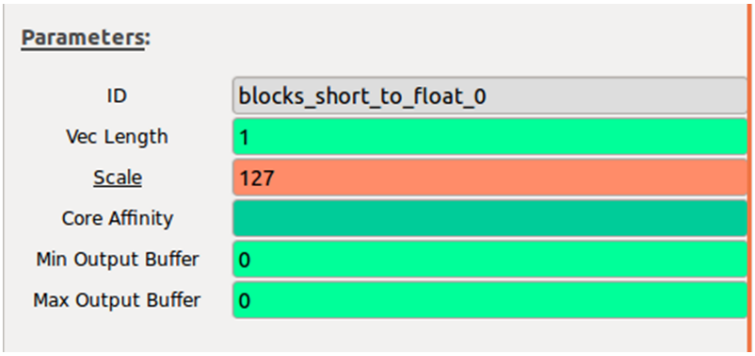


Figura 34: Bloque Short-to-Float en GNU Radio.

4.6.7 Filtro Pasa-Bajos

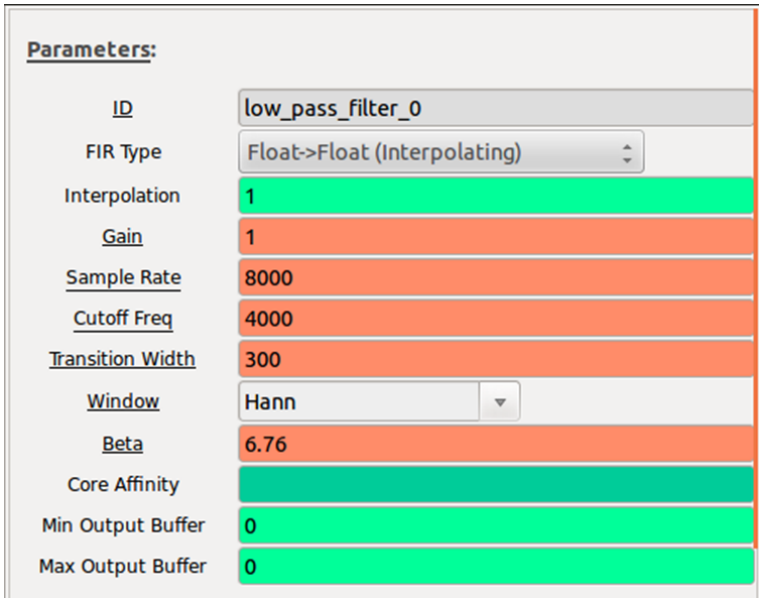


Figura 35: Bloque Filtro Pasa-Bajos en GNU Radio.

La tasa de muestreo ha sido establecida en 8 kHz para guardar concordancia con el bloque de audio. La frecuencia de corte del filtro ha sido fijada en 4 kHz con un ancho de transición de 300 Hz. La ventana escogida es la de *Hann* debido a su excelente discriminación de señales en filtros pasa bajos (Wurzburg, 2013).

4.6.8 Sumidero de Audio

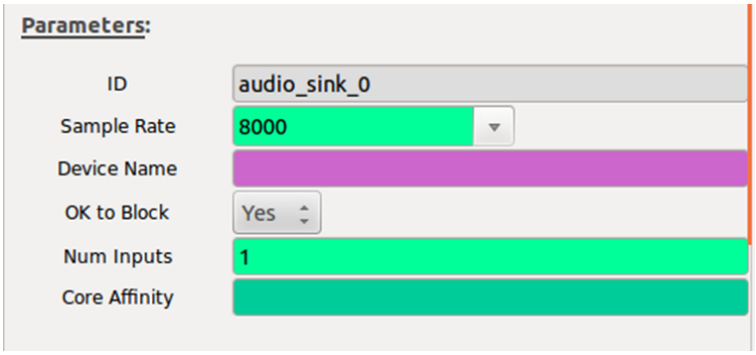


Figura 36: Bloque Audio Sink en GNU Radio.

El bloque de sumidero de audio es capaz de reproducir audio desde la tarjeta de sonido del dispositivo, específicamente desde la salida de audio, a distintas tasas de muestreo desde 8 kHz hasta 44.1 kHz. Al igual que su análogo en el transmisor, la tasa de muestreo ha sido establecida en 8 kHz.

4.7 Entorno de simulación

Adicional a los bloques descritos tanto en el subsistema transmisor como en el subsistema receptor existe un bloque de simulación de canal llamado *Channel Model AWGN*.

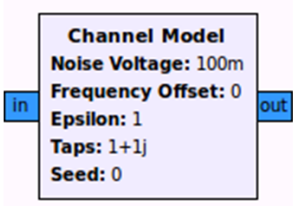


Figura 37: Bloque Channel Model AWGN en GNU Radio.

El bloque *Channel Model AWGN (Additive white Gaussian noise)* es esencialmente un canal de ruido aditivo gaussiano (AWGN) con algunas opciones extra. Este modelo simula un canal AWGN, así como desplazamientos de frecuencia y sincronización entre el transmisor y el receptor, y un entorno de multitrayectos simple. Los parámetros configurables de este bloque son:

- **Noise Voltage**: Configura el nivel de ruido AWGN como una tensión (a calcular externamente para satisfacer, por ejemplo, una SNR deseada).
- **Frequency Offset**: Desplazamiento de la frecuencia normalizada. Un valor de 0 indica que no se presenta offset; 0.25 establece un cuarto de la tasa de símbolo.
- **Épsilon**: Configura la sincronización de la muestra para emular las diferentes tasas entre los relojes de muestreo del transmisor y el receptor. El valor de 1.0 es lo normal.
- **Taps**: Derivaciones de un filtro FIR para emular un perfil de retardo multitrayecto.
- **Noise Seed**: Generador de números aleatorios para la fuente de ruido.

La Figura 71 disponible en los Anexos del presente trabajo muestra el sistema de comunicaciones con el bloque de modelamiento de canal incluido.

Además, *GRC (GNU-Radio Companion)* también incluye un bloque gráfico de interfaz: el *QT GUI Sink* de la Figura 38 cuyas funciones se describen a continuación:

- **fftsize**: tamaño inicial FFT
- **wintype**: tipo de ventana FFT inicial, pueden ser:
  - WIN_BLACKMAN.
  - WIN_BLACKMAN_HARRIS
  - WIN_HAMMING
  - WIN_HANN
  - WIN_KAISER
  - WIN_RECTANGULAR.

  se recomienda WIN_BLACKMAN_HARRIS si no se tiene necesidades específicas del entorno de ventanas.
- **fc**: frecuencia central de la pantalla del eje X.
- **Ancho de banda**: establece el rango del eje X en torno de fc.
- **Nombre**: El título del objeto de interfaz gráfica de usuario en la barra de título.
- **plotfreq**: Ventana de visualización de frecuencia.
- **plotwaterfall**: Visor cascada.
- **plottime**: Visor de tiempo.
- **plotconst**: Ventana de visualización de constelación.

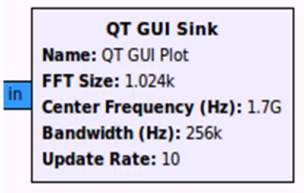
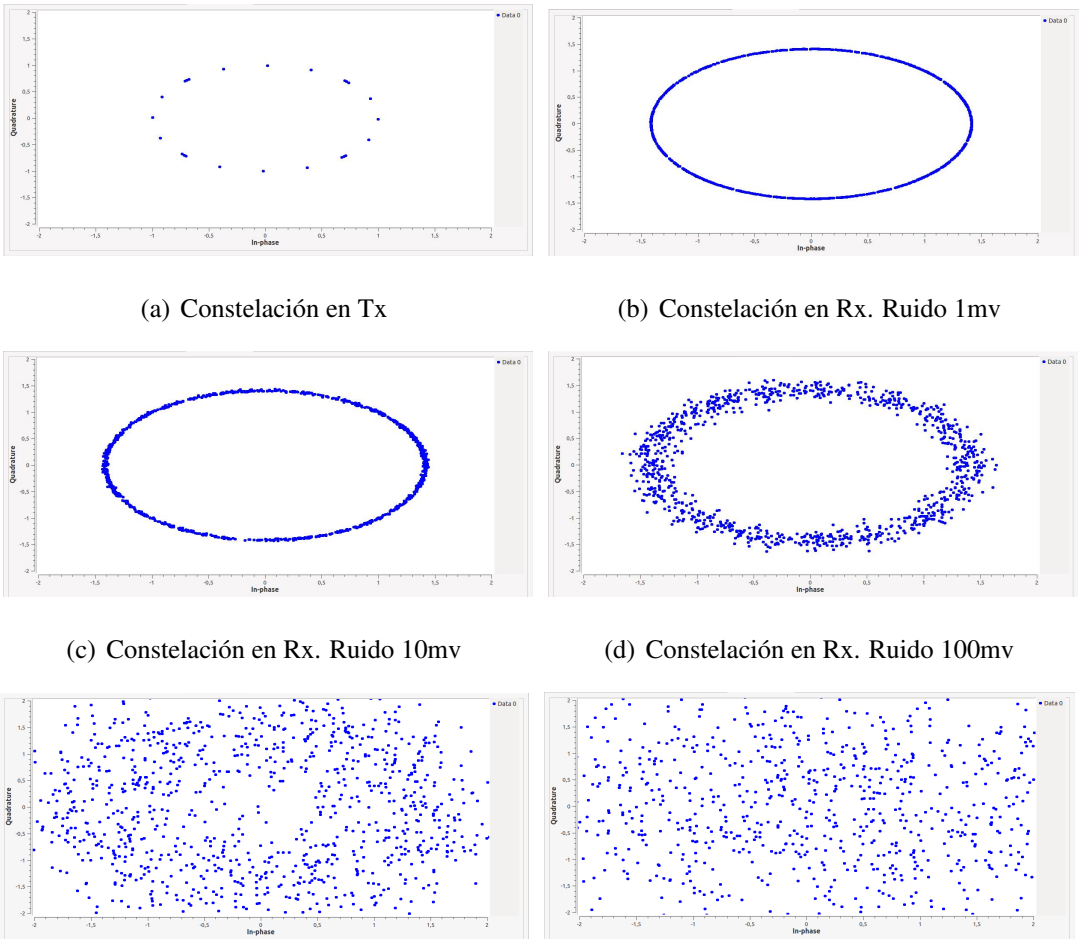


Figura 38: Bloque de interfaz gráfica QT GUI Sink.

Mediante el uso de estos dos bloques se ha logrado simular varios entornos de canal para tener una idea aproximada cuando la implementación sea realizada, la Figura 4.7 muestra el comportamiento de la constelación *GMSK* recibida frente a variaciones en el ruido del canal. Se han manipulado los factores *Noise voltage* y *taps*; como puede observarse, entre mayor sea el factor de ruido existente en el canal, la comunicación será cada vez peor existiendo una tasa de error mucho mayor.



(a) Constelación en Tx

(b) Constelación en Rx. Ruido 1mv

(c) Constelación en Rx. Ruido 10mv

(d) Constelación en Rx. Ruido 100mv

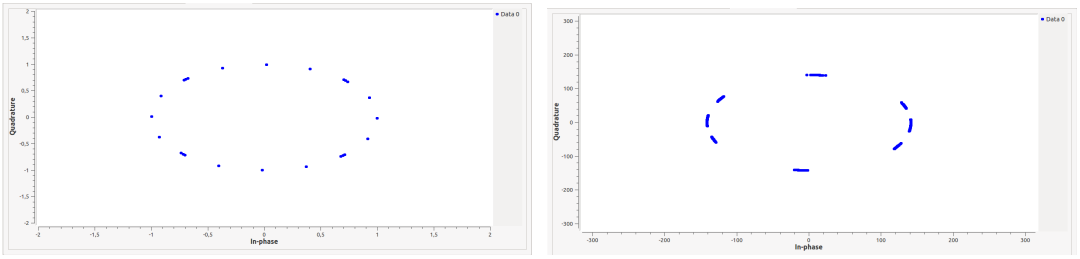
(e) Constelación en Rx. Ruido 500mv

(f) Constelación en Rx. Ruido 1V

Figura 39: Constelación GMSK Tx y Rx con simulación en la variación en el ruido del canal.

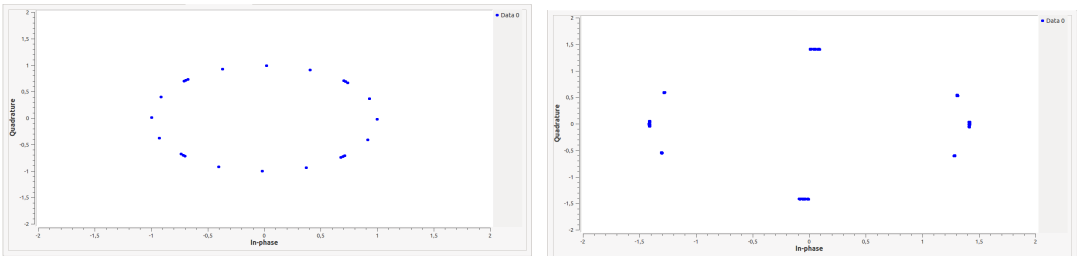
También se han simulado condiciones de retardo de multitrayecto manipulando el parámetro *taps*, obteniéndose los resultados mostrados en la Figura 4.7. La Figura 4.7 ilustra el resultado de la constelación cuando se agrega un *Offset* en frecuencia, como

resultado pueden observarse ciertos desfases en la señal, los cuales tienden a agrupar la constelación en ciertas regiones deformándola.



(a) Constelación en Rx sin multitrayecto (b) Constelación en Rx con multitrayecto

Figura 40: Constelación GMSK con simulación de efectos multitrayecto.



(a) Constelación en Rx sin offset (b) Constelación en Rx con offset

Figura 41: Constelación GMSK con simulación de offset en frecuencia.

GNU Radio ofrece varios archivos de configuración de *drivers* de audio, los cuales son modificables en función de la aplicación a desarrollar, estos archivos se encuentran contenidos en: `/usr/etc/gnuradio/conf.d`. La manipulación de estos archivos resultará en una mayor fidelidad en la transmisión, menor retardo y latencia adecuada dependiendo de las características del medio inalámbrico en el cual se encuentren los dispositivos. Para el desarrollo del presente proyecto se ha elegido la configuración del *driver* de audio *ALSA (Advanced Linux Sound Architecture)*, y se ha manipulado los valores correspondientes al tiempo y número de periodo para lograr un cambio en el tiempo de *buffering* total que se obtiene como el producto de los dos factores mencionados anteriormente.

4.8 Proceso de Embebido

El objetivo de esta sección es mostrar el proceso de migración del diseño del sistema de comunicaciones desde el entorno de simulación a la implementación embebida en el equipo por medio de un *Script* de tal forma que el manejo de la plataforma *software* de diseño *GNU Radio-companion* pase a un segundo plano logrando un funcionamiento *Plug & Play*.

Cuando un diagrama de bloques diseñado en GRC es compilado, se generan varios archivos, entre los cuales están: un archivo `.sh` ejecutable por consola y un archivo con extensión `.py` el cual es un fichero ejecutable multiplataforma similar a `.exe` en *Windows*. Este archivo corresponde al *Script* que será ejecutado al iniciar el equipo.

La ejecución del archivo `.py` desde un terminal o consola se realiza con: `./nombre del archivo` una vez que se ha localizado su ubicación en el sistema mediante el comando: `cd rutadelarchivo`.

Los sistemas con base en *Linux* por defecto requieren de un nombre de usuario y contraseña para realizar el inicio de sesión, deberá desactivarse esta característica para lograr que el *Script* pueda ejecutarse con normalidad.

4.8.1 Inicio Automático de Sesión en Linux

Es posible configurar un inicio automático evitando la petición de usuario y contraseña para la ejecución automática del *script* ya sea de manera inmediata o con un cierto intervalo de espera antes de proceder.

4.8.1.1 Método 1

Para configurar esta opción, en la interfaz gráfica del *USRP* dirigirse a: *Administración » Preferencias de Ventana de Inicio » Seguridad*, donde las opciones mostradas en la

Figura 42 serán desplegadas:

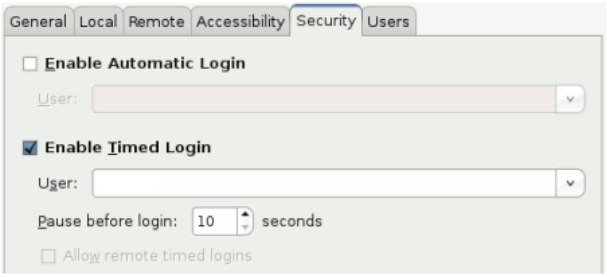


Figura 42: Preferencias de Ventana de Inicio Linux.

Podemos elegir la ejecución inmediata o con un intervalo de espera antes del inicio. Una vez realizados estos pasos el inicio de usuario y contraseña no será necesario.

4.8.1.2 Método 2

Otro método para configurar el inicio automático de sesión es realizando la edición del fichero `lightdm.conf`, para ello ejecutaremos el siguiente comando por consola:

```
$ /etc/lightdm/lightdm.conf
```

Una vez abierto el archivo, procederemos a editarlo incluyendo: `autologin-user= "nombre de usuario"` al final del archivo. Recordar que el nombre de usuario es el que aparece despues del caracter @ en el título de la ventana de consola, así: `nombredeusuario@xxxx`.

En este caso sería: `autologin-user=root`. Si por el contrario no se desea que el sistema autoarranque, quedaría de la siguiente manera: `autologin-user=`.

De esta forma el inicio automático de sesión en *Linux* puede omitirse.

4.8.2 Ejecución automática de un programa al iniciar Linux

A continuación se mostrarán los pasos a seguir para conseguir la ejecución automática del archivo `.py` que contiene el sistema diseñado

4.8.2.1 Método 1

Dirigirse a: *Sistema » Preferencias » Aplicaciones al inicio*. En la pestaña *"Programas de inicio"* clicar en *"Añadir"* como puede verse en la Figura 43.

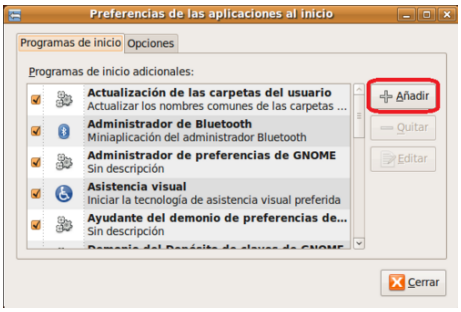


Figura 43: Ventana Programas de Inicio Linux.

Se abrirá una nueva ventana, en la cual deberá ingresarse el nombre del archivo `.py` y el comando habitualmente utilizado para ejecutar el archivo, por ejemplo: `/usr/bin/gnome-terminal/GMSK_TX`.

En adelante el programa será ejecutado al inicio de sesión, esto puede verse en la Figura 44.

Figura 44: Ventana de Opciones Programas de Inicio de Linux.

4.8.2.2 Método 2

El segundo método permite la ejecución de un *Script* localizado en cualquier directorio del sistema cuando este inicie sesión. Suponiendo que tenemos el *script bash* llamado `GMSK_TX` producto de la compilación en GRC se deberá otorgársele permisos de ejecución con el comando `sudo chmod +x GMSK_TX.sh`.

A continuación se deberá editar el fichero `rc.local` y añadir la instrucción `sh`, que es la encargada de ejecutarlo después de las secciones comentadas, pero antes de la línea `'exit 0'`. En el archivo agregar: `sudo nano/etc/rc.local` y añadir las líneas correspondientes, quedando de la siguiente forma:

```
#!/bin/sh -e

# rc.local

# This script is executed at the end of each multiuser runlevel.

# Make sure that the script will "exit 0" on success or any other

# value on error.

# In order to enable or disable this script just change the execution

# bits.

# By default this script does nothing.

sh /home/root/GMSK_TX.py

exit 0
```

Nótese que se ha añadido el comando `sh` para indicar que se desea su ejecución. Para desinstalarlo, basta con comentar la línea añadida o bien eliminarla.

Puede darse el caso en el que se requiera un tiempo de espera antes de la ejecución del *Script*, por ejemplo un *sleep* de 10 segundos antes ya que `rc.local` suele ejecutarse antes que otros servicios, de esta manera se evita el conflicto en el inicio de varios

servicios.

Otra opción ejecutar el *Script* junto con el resto de servicios del sistema, para ello el *Script* debe ser movido a la carpeta *init.d*, deberá otorgársele permisos de ejecución y actualizar *rc.d* con configuración por defecto, con los siguientes comandos:

```
sudo mv /home/root/GMSK_TX.sh /etc/init.d/
sudo chmod +x /etc/init.d/GMSK_TX.sh
sudo update-rc.d GMSK_TX .sh defaults
```

Para desinstalar lo realizado (en el caso que se requiera), ejecutar el comando: `sudo update-rc.d -f GMSK_TX.sh remove` y eliminar manualmente el *Script* de `init.d`, así: `sudo rm /etc/init.d/GMSK_TX.sh`. Tomado de (Vilchez, 2011), (Vijamaro, 2009), (D., 2012) y (Koolwal, 2012).

CAPITULO 5

PRUEBAS Y RESULTADOS

5.1 Protocolo de pruebas

Una vez realizado el proceso de embebido en los *USRPs* y tras haber comprobado que ambos dispositivos son capaces de sostener una correcta transmisión y recepción, se inició un proceso de pruebas tanto técnicas como subjetivas para evaluar la calidad del sistema.

El desempeño integral del sistema puede entenderse como un árbol del cual partirán dos ramas principales: Las mediciones de carácter objetivo, y las mediciones de carácter subjetivo. Las mediciones de carácter objetivo implican la evaluación y el contraste de factores netamente técnicos, como son: ancho de banda utilizado para la transmisión, potencia de transmisión, potencia de recepción, medición de retrasos, factores y problemas de propagación inalámbrica etc. Por otro lado, El sistema diseñado fue planeado para ser usado con señales de voz, por lo que se requerirán mediciones de naturaleza subjetiva que de acuerdo a la opinión del usuario generen una calificación que permita evaluar el desempeño del sistema.

Las etapas de pruebas a las que se sometió el sistema se detallan a continuación:

- **Respuesta en frecuencia:** Análisis de la efectividad de los filtros digitales usados en el diseño.
- **Potencia de transmisión y recepción:** Para determinar alcance máximo de Recepción
- **Pruebas en varios ambientes de propagación:** Analizar el comportamiento y respuesta del sistema en varios ambientes y con obstáculos.
- **Ancho de Banda y Eficiencia Espectral:** Para verificar frecuencias de resonancia y Ancho de Banda ocupado.
- **Retardos de transmisión**: Evaluación de las muestras de señales recibidas en contraste con la señal enviada detallando aspectos de diferencia en tiempo.
- **Análisis de correlación**: Contraste de las muestras de señales recibidas en con la señal original enviada.
- **Encuesta *Mean Opinion Score*:** Percepción subjetiva del sistema.
- **Test de intelegibilidad segmental:** Detalle de combinaciones silábicas en las que el sistema presentó falencias.

Para finalizar se realizó pruebas de transmisión de voz bajo otros esquemas de modulación más complejos de *GMSK*, pero dada la complejidad computacional y la capacidad de procesamiento que requieren estos procesos estas pruebas fueron realizadas sustituyendo un dispositivo embebido por otro que utilice procesamiento en una PC *host* como el *USRP N210*.

5.2 Análisis de la Eficiencia del Sistema

5.2.1 Caracterización de Antenas

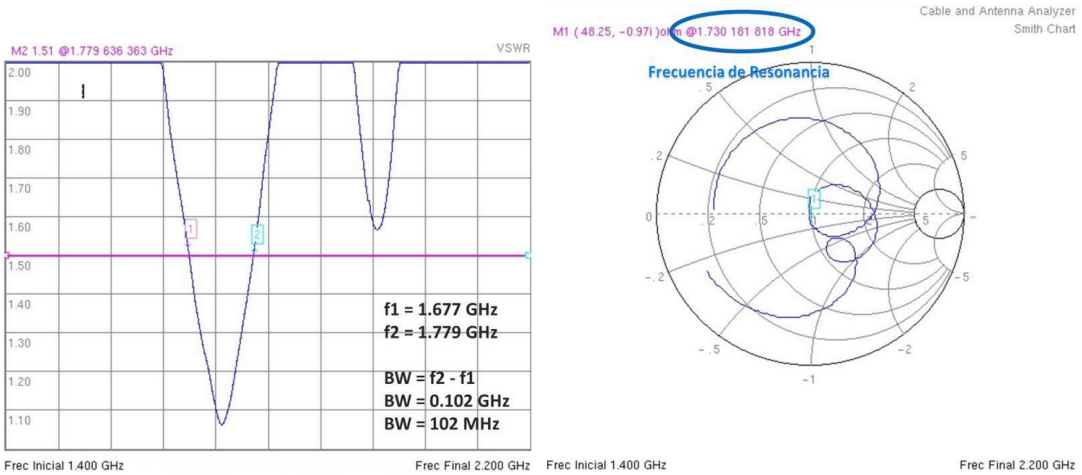
Los parámetros técnicos de las antenas utilizadas para el presente diseño se encuentran detallados en el Capítulo 4. del presente documento, en la subsección *"Requerimientos de Hardware"*. La Tabla 11 resume los parámetros de VSWR (*Voltage Standing Wave Ratio*) de las antenas utilizadas; el intervalo entre la frecuencia inicial y final indica el rango en Hz en el cual se midió el VSWR de la antena.

Tabla 11: Parámetros VSWR antenas.

Parámetros	
Frecuencia Inicial	1.4GHz
Frecuencia Final	2.2GHz
VSWR Superior	2
VSWR Inferior	1
Límite	1.5

La configuración física de los elementos para la realización de este experimento se muestra en la Figura 45. La Figura 5.2.1 muestra la frecuencia de resonancia de la antena en donde el valor del VSWR es más próximo a 1, este valor es 1.7 GHz marcando un VSWR de 1.04, por este motivo la frecuencia de 1.7 GHz fue usada para el diseño del sistema de comunicaciones. El ancho de banda obtenido con un VSWR de 1.5 para las antenas es de 102 MHz aproximadamente.

Figura 45: Configuración Física Caracterización de Antenas.



(a) VSWR (b) Carta de Smith

Figura 46: Frecuencia de resonancia, parámetros VSWR y ancho de banda de las antenas utilizadas.

5.2.2 Rango de Frecuencia de las Tarjetas Hijas

Antes de realizar pruebas de transmisión del sistema de comunicaciones se ha creído pertinente realizar experimentos de transmisión que corroboren la frecuencia de operación de las tarjetas *Daughterboard* a usarse. Las especificaciones técnicas referentes a estas tarjetas se encuentran en el Capítulo 4. del presente documento, en la subsección "Tarjetas *Daughterboard*". En la Figura 47 se muestra la configuración física de los elementos utilizados para esta tarea, el analizador de espectros ha sido configurado con una atenuación de 50 dB para evitar el uso de atenuadores físicos conectados al cable. La Figura 5.2.2 muestra el comportamiento de las tarjetas hijas frente a varios rangos de frecuencia.

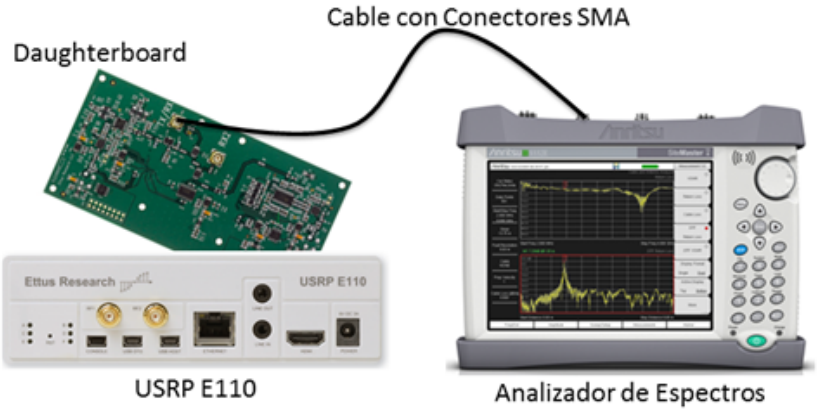


Figura 47: Configuración Física Pruebas Rango de Frecuencia.

Como se ha podido comprobar, el rango efectivo de las tarjetas *Daughterboard* es de 1.5 GHz a 2 GHz, 0.1 GHz menos en rango de funcionamiento que el especificado por el fabricante. La Figura 48 ilustra el comportamiento experimental obtenido por las tarjetas.

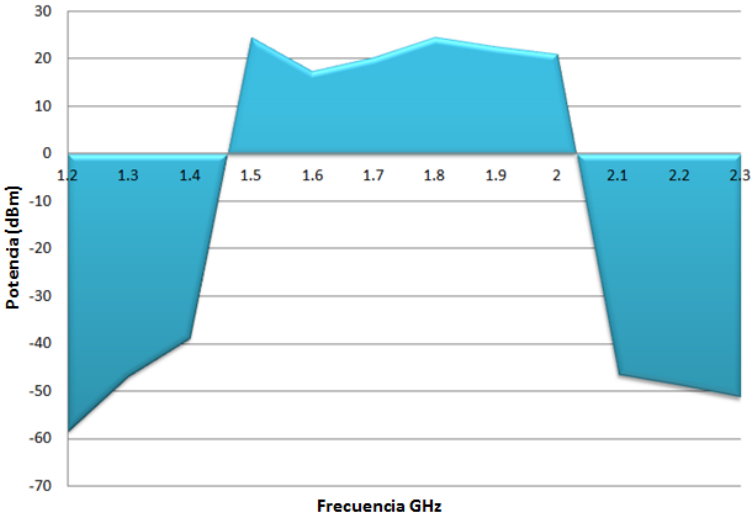
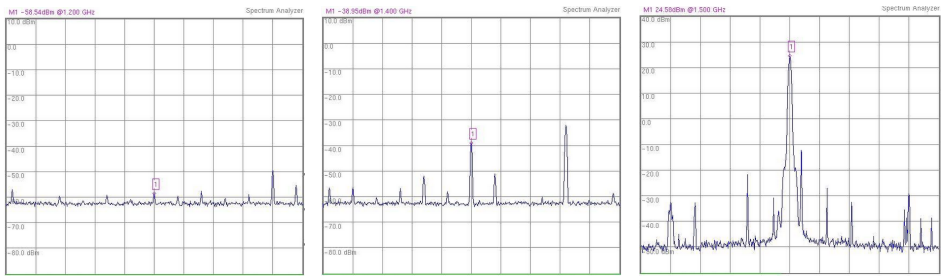
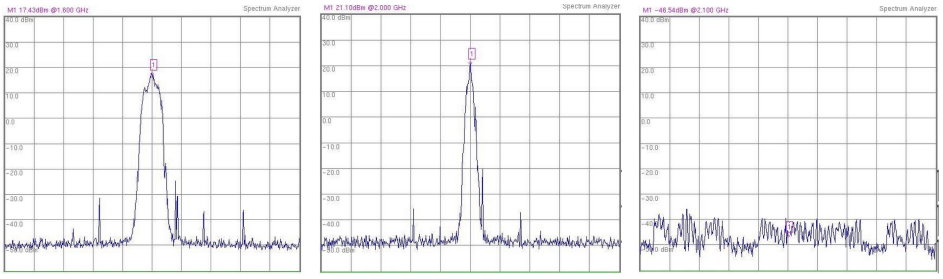


Figura 48: Región de Trabajo Daughterboards RFX1800.



(a) Frecuencia 1.2GHz (b) Frecuencia 1.4GHz (c) Frecuencia 1.5GHz



(d) Frecuencia 1.6GHz (e) Frecuencia 2GHz (f) Frecuencia 2.1GHz

Figura 49: Frecuencia de resonancia de las tarjetas daughterboard utilizadas.

5.2.3 Respuesta en Frecuencia

El sistema de comunicaciones diseñado implementa bloques *Software* pasa-bajos de filtrado de señal, esta subsección estará dedicada a analizar la respuesta en frecuencia de los filtros, cuyos parámetros se detallan en del Capítulo 4 del presente documento. Para la realización de este experimento se introdujo una señal senoidal con amplitud 0.5 Vpp al puerto de audio del *USRP* transmisor y se comparó el voltaje de entrada de la señal con el de salida, tomado en la salida de la antena de la placa *Daughterboard*, la configuración de los dispositivos se muestra en la Figura 50. Los resultados obtenidos se muestran en la Figura 51.

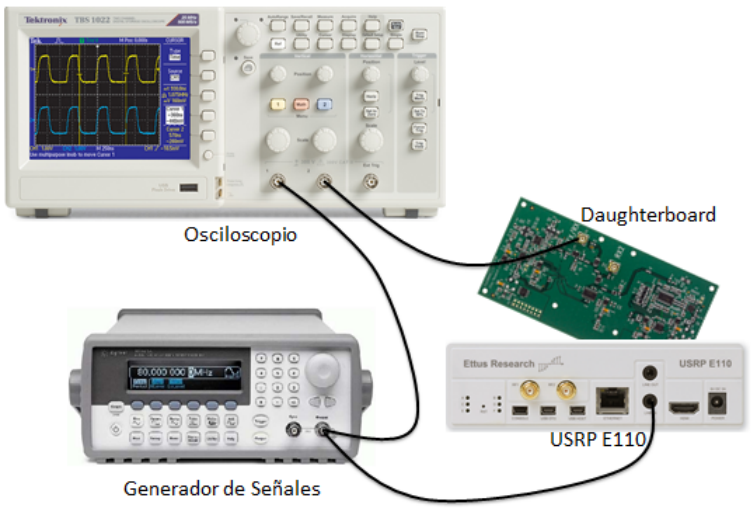


Figura 50: Configuración Física Respuesta en Frecuencia Filtro.

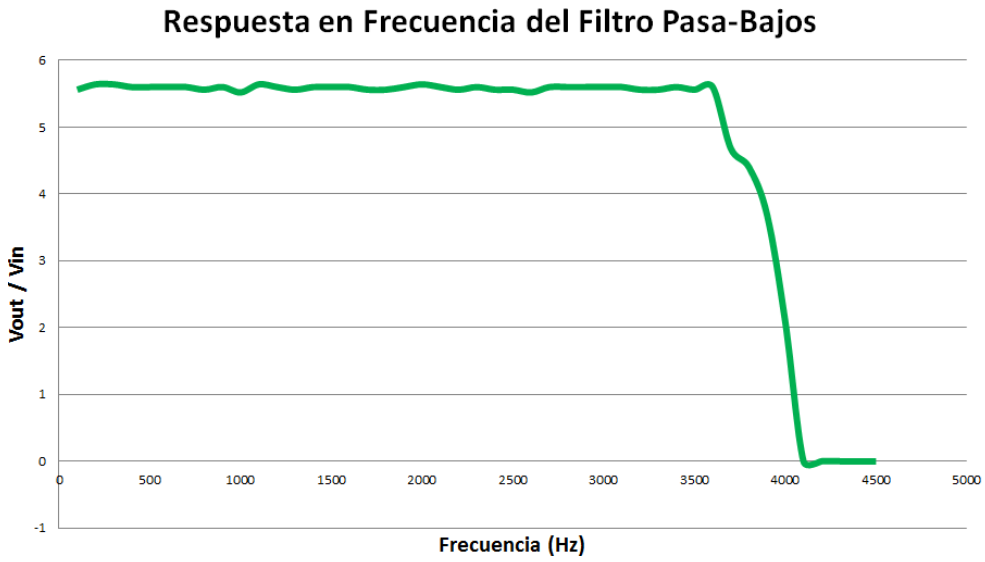


Figura 51: Respuesta en frecuencia del filtro software pasa-bajos implementado.

Como puede verse en la Figura 51, el filtro diseñado resulta ser bastante efectivo con respecto a las frecuencias a las cuales se lo ha configurado ya que presenta una transición muy marcada entre las frecuencias pasantes y las frecuencias discriminadas. La relación $\frac{Vout}{Vin}$ resulta mayor que 1 debido a que el propio filtro puede ser configurado de manera que agregue una ganancia a las frecuencias pasantes, y como resultado la señal de Rx posee una mayor amplitud que la señal muestreada por el bloque *Audio Source*.

5.2.4 Potencia de Transmisión y Recepción

Las pruebas de potencia de transmisión del sistema de comunicaciones fueron realizadas tomando valores extraídos desde la interfaz gráfica de *GNU Radio* como con el uso de dispositivos de medición externos (Analizador de espectros).

Como se ha mencionado en el Capítulo 4. del presente documento, los bloques *UHD USRP Source* y *UHR USRP Sink* contienen un parámetro para configurar el valor de la ganancia de la antena en dB. El procedimiento para determinar el comportamiento

en la variación de este parámetro siguió los siguientes pasos:

- Conectar el *USRP* con su respectiva tarjeta *Daughterboard*, y mediante un cable *SMA* conectar el puerto de salida de la tarjeta hija con el analizador de espectros.
- Variar los valores de la ganancia de la antena en el bloque *UHD USRP Sink* en pasos de 1, desde -10dB hasta que el analizador de espectros entregue valores constantes.
- El analizador de espectros entrega resultados en dBm, por lo que ha de transformarse este valor en mW con la fórmula:

$$P[mW] = 10^{P[dBm]/10} \tag{5.1}$$

- Graficar los datos obtenidos.

La configuración física del experimento es exactamente igual a la que se muestra en la Figura 47. Mediante experimentación en la variación del valor de la ganancia de la antena se ha podido determinar que este parámetro puede funcionar también como atenuador de la señal y posee un comportamiento lineal hasta llegar a la potencia máxima de transmisión permitida por la tarjeta *Daughterboard* que son 100 mW. La Figura 5.2.4 muestra los valores de ganancia obtenidos conforme se ha ido variando el parámetro de ganancia de la antena; el eje x representa la ganancia establecida en dB en *software* y el eje y corresponde a la ganancia medida en la *Daughterboard* en mW.

Para lograr la visualización de la relación señal ruido *GNU Radio* ofrece una biblioteca que contiene varios bloques de interfaz. Estos bloques se encuentran localizados en el módulo `gnuradio.qtgui` y deben ser importados dentro de *Python* con el comando `from gnuradio.qtgui import qtgui`.

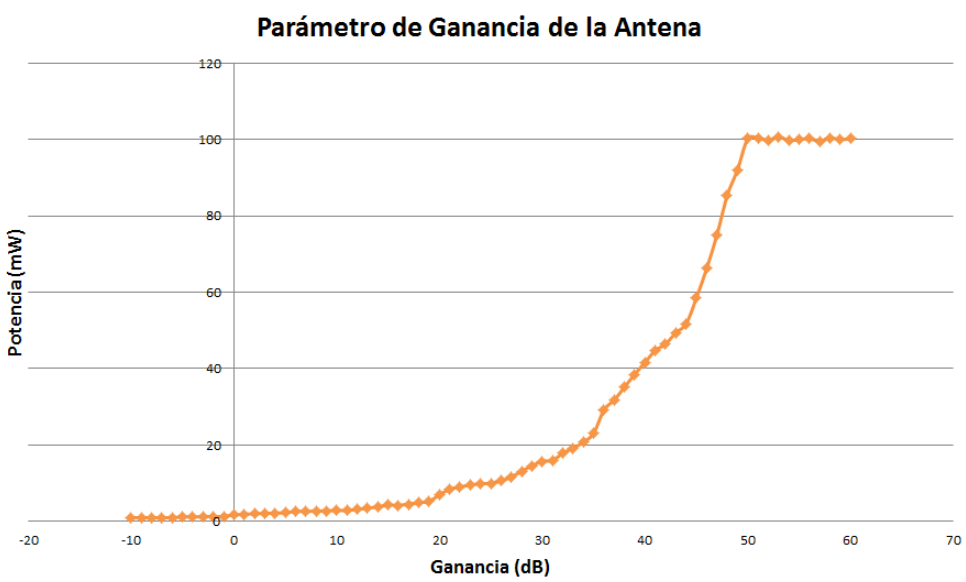


Figura 52: Respuesta de la antena con la variación del parámetro Antena Gain en el bloque UHD USRP Sink/Source.

La Figura 53 ilustra el valor de potencia de transmisión obtenido por la interfaz del *Software GNU Radio* y la Figura 54 muestra la portadora de la señal *GMSK*. La Figura 55 muestra la portadora de la señal recibida por el analizador de espectros a cierta distancia del transmisor y con un spam mayor para mejor apreciación. Debido a la periodicidad introducida por el procesamiento discreto y a una imperfección en el filtraje de las *Daughtherboard* se observa un armónico de la señal de interés, estas réplicas corresponden a armónicos de baja potencia de las señales periódicas. La señal Rx tendrá más atenuación a medida que se aleje el receptor o se presenten obstáculos entre ambos dispositivos.

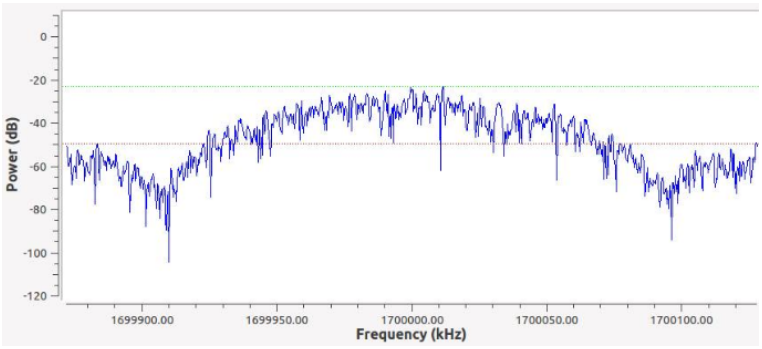


Figura 53: Espectro GMSK transmitido. Obtenido con bloques de interfaz gráfica GNU Radio.

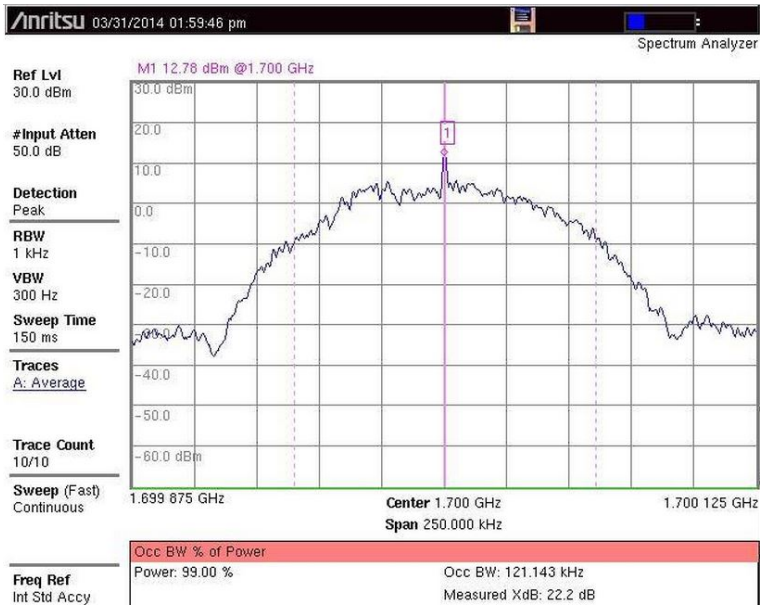


Figura 54: Espectro GMSK transmitido. Obtenido con analizador de espectros externo.

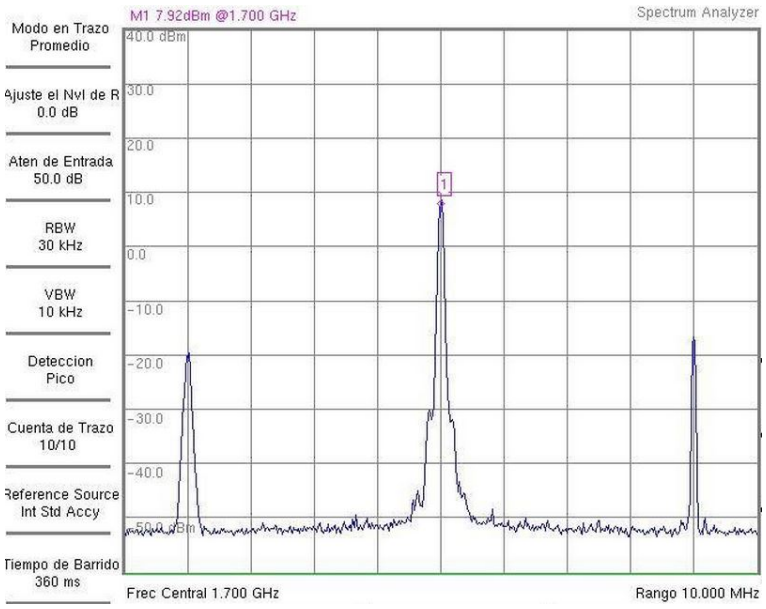


Figura 55: Espectro GMSK recibido. Obtenido con analizador de espectros externo.

5.2.5 Pruebas en Varios Ambientes de Propagación

Se realizaron pruebas de transmisión de voz en tres ambientes de propagación:

1. Ambiente de laboratorio sin obstáculos.

2. Ambiente de laboratorio con obstáculos.

3. Campo Abierto.

El esquema físico de la configuración de los equipos se describe en la Figura 13. Como resultados: En el Ambiente 1. el sistema fue capaz de mantener la comunicación con una buena calidad de voz en el receptor a distancias de hasta 8 m con ligeras atenuaciones en la potencia de recepción a medida que la distancia del transmisor era mayor, lamentablemente no se dispuso de un espacio cerrado de mayores dimensiones para constatar el comportamiento del sistema más allá de esa distancia. La Figura 56 ilustra la variación de la potencia de Rx del sistema en función de la distancia y una

potencia de Tx constante para el Ambiente 1; como puede observarse, la potencia de Rx es directamente proporcional a la distancia entre dispositivos.

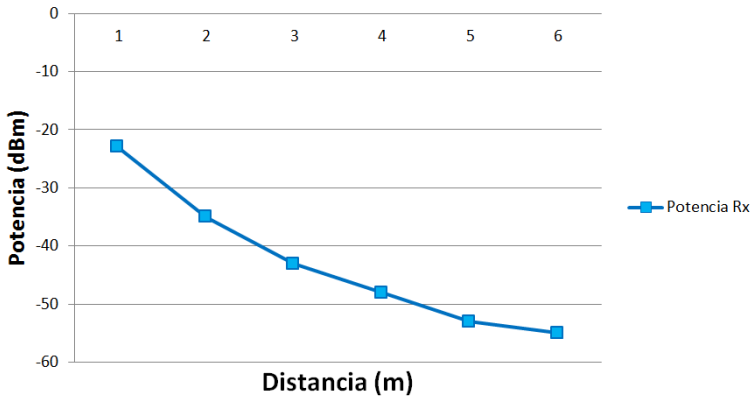


Figura 56: Potencia de Rx para el Ambiente 1 (PTx=-16dB).

En el Ambiente 2. se realizó mediciones del sistema con varios obstáculos entre el transmisor y receptor a varias distancias. Estos obstáculos incluyeron cristal, madera y paredes de concreto. Pudo constatarse que la potencia de recepción fue mucho menor al tener como obstáculo una pared de concreto, los resultados en el receptor mostraron una señal de voz de baja calidad y en ciertos momentos ininteligible.

El Ambiente 3. resultó ser el peor de todos. Recordemos que la potencia de salida de la tarjeta *Daughterboard* es de 100mW, por lo que en un ambiente totalmente abierto la comunicación no pudo lograrse a distancias mayores a 1m sin que la señal de voz se transformara en ruido hasta atenuarse completamente.

La distancia de comunicación experimentada en el Ambiente 3. guarda estrecha relación con la primera zona de Fresnel y el campo cercano radiado por las antenas. Durante la primera mitad del ciclo, la potencia se irradia desde la antena, en donde parte de la potencia se guarda temporalmente en el campo cercano. Durante la segunda mitad del ciclo, la potencia que está en el campo cercano regresa a la antena. Esta acción es similar a la forma en que un inductor guarda y suelta energía, por esta razón

el campo cercano se llama en ocasiones “campo de inducción” (Cordoba, 2003) y se expresa con la siguiente fórmula.

$$0.62\sqrt{\frac{D^3}{\lambda}} \leq r < \frac{2D^2}{\lambda} \tag{5.2}$$

El campo cercano se considera el área dentro de la distancia $\frac{D}{\lambda}$ de la antena, donde λ es la longitud de onda, D es la longitud de la antena (23.8cm) y r una distancia desde un punto de radiación arbitrario hasta un punto de medición arbitrario.

La longitud de onda λ viene dada por la fórmula: $\lambda = \frac{c}{f}$ donde c es la velocidad de la luz (3e8 m/s) y f la frecuencia de transmisión (1.7 GHz).

$$\lambda = \frac{c}{f} = \frac{3e8}{1.7e9} = 0.176m \tag{5.3}$$

Sustituyendo los valores obtenidos en la fórmula del campo cercano tenemos:

$$0.62\sqrt{\frac{0.238^3}{0.176}} \leq r < \frac{2 * 0.238^2}{0.176} \tag{5.4}$$

$$0.351m \leq r < 0.643m \tag{5.5}$$

La fórmula de campo cercano no toma en cuenta la potencia del dispositivo ya que muestra la distancia a partir de la cual la comunicación puede darse. Estas distancias son bastante similares a las que el sistema era capaz de mantener la comunicación en el Ambiente 3. por lo que queda comprobado que la zona de campo cercano en un ambiente de campo abierto para el sistema de comunicaciones diseñado va desde 0.35 m a 0.64 m sin amplificadores como ha sido el caso. Esta distancia será mayor en función de la potencia de amplificadores externos acoplados al *USRP*.

La razón por la cual se obtuvo una distancia mucho mayor en el Ambiente 1 y 2 puede deberse a que el dispositivo receptor aprovecha los multitrayectos del entorno en el que se encontraba.

El ancho de banda de la transmisión es de aproximadamente 125 kHz, lo que es un ancho de banda aceptable para transmisiones de voz con esquemas de modulación *narrowband* como es el caso de *GMSK*. El ancho de banda no presenta cambios frente a variaciones de distancia o potencia de Tx/Rx.

5.2.6 Probabilidad de Error, Ancho de Banda y Eficiencia Espectral

La probabilidad de error para *GMSK* viene dado por la expresión siguiente:

$$P_e = \frac{1}{2} erfc\left(\sqrt{\frac{\alpha E_b}{2N_0}}\right) \tag{5.6}$$

Donde α es un valor constante que depende del valor BT de filtro gaussiano. Cuando α tiene un valor de 2, corresponde a un BT igual a infinito, y cuando BT=0.3, el valor de α es de 0.9. Este concepto puede ser mejor entendido en función de la degradación del sistema mostrada en la Figura 57, y viene expresada por:

$$Degradacion(dB) = 10log\left(\frac{\alpha}{2}\right) \tag{5.7}$$

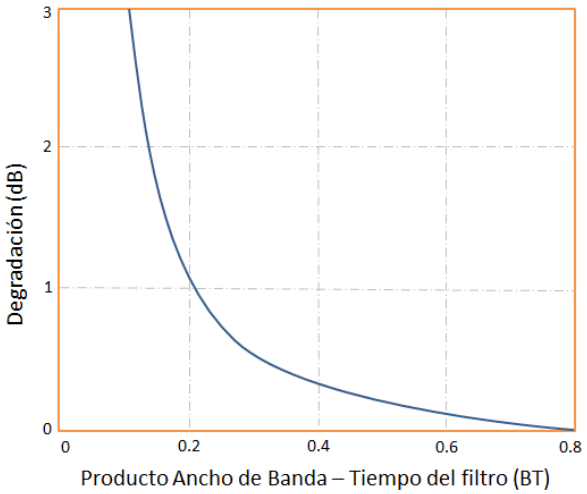


Figura 57: Parámetro BT vs Degradación de la señal. (Manandayam, 2012)

Para valores de BT inferiores a 0.3 la degradación de la señal es muy elevada, por tanto su recuperación en el receptor se vuelve muy difícil. La Figura 58 relaciona la tasa de error del sistema *GMSK* con la relación señal-ruido para algunos valores de BT.

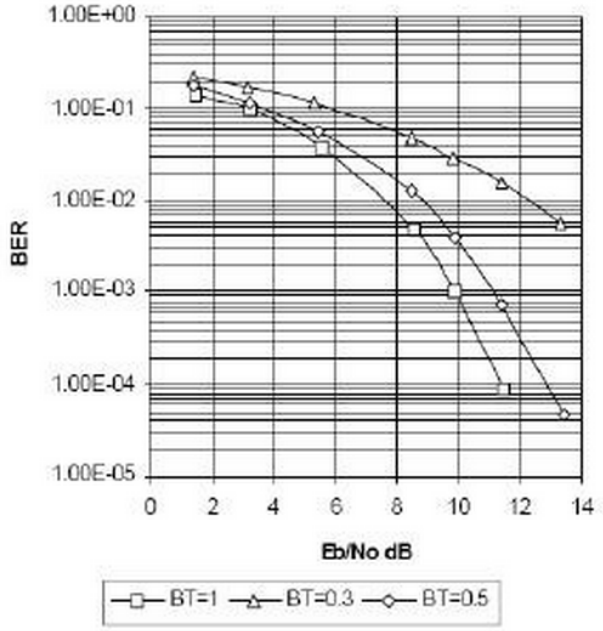


Figura 58: BER para varios valores de BT. (Manandayam, 2012)

Se ha procedido a obtener la probabilidad de error P_e con α=0.9 con la ecuación 5.6, obteniéndose los valores presentados en la Tabla 12 en donde puede observarse que a medida que la relación señal-ruido crece, la probabilidad de error es menor, esto se observa gráficamente en la Figura 59.

Tabla 12: BER para α=0.9.

SNR	**BER**(GMSK)	**SNR**	**BER**(GMSK)
1	0.1715	9	0.002215
2	0.09	10	0.00135
3	0.05	11	0.000825
4	0.0289	12	0.00051
5	0.01695	13	0.0003125
6	0.01005	14	0.000193
7	0.00605	15	0.0001195
8	0.003645	16	0.000074

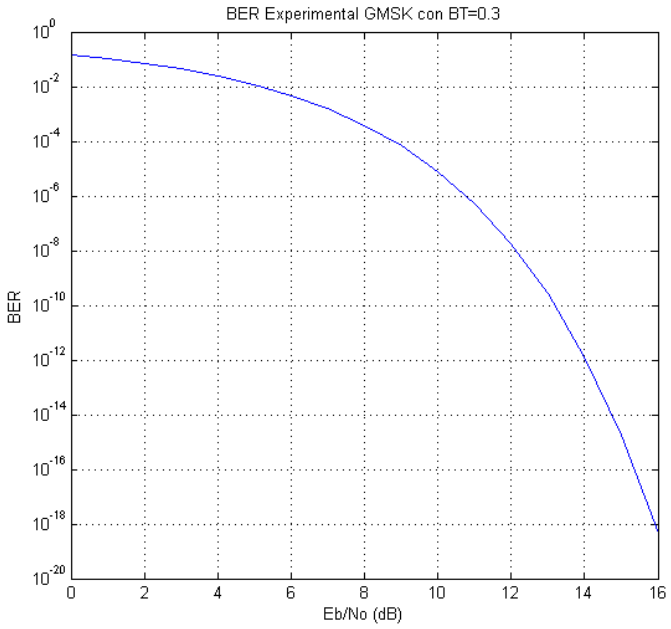


Figura 59: Comportamiento del BER para GMSK con α=0.9.

GMSK posee una eficiencia espectral menor a 0.7 bps/Hz, que viene dado por la

expresión (Manandayam, 2012) (Torres & Paz, 2012):

$$\nu = \frac{R_b}{W} = \frac{log_2(A)}{1+\alpha}, \tag{5.8}$$

En donde A representa el alfabeto de la constelación, que para el caso de *GMSK* es dos, puesto que los símbolos correspondientes para este alfabeto son 1 y 0, y α corresponde al factor asociado al BT.

$$\nu = \frac{R_b}{W} = \frac{log_2(2)}{1+0.9} = 0.5263 bps/Hz \tag{5.9}$$

El ancho de banda en GMSK puede hallarse a partir de la expresión:

$$\nu = \frac{R_b}{W}, \tag{5.10}$$

En donde R_b es la tasa de transmisión o tasa de bits (bps) entrante al bloque de modulación. En el caso del presente diseño es 32kbps ya que es la tasa de bits entregada por el vocoder G.721, y W es el ancho de banda de la modulación *GMSK* en banda base. Entonces:

$$W = \frac{R_b}{\nu} = \frac{32 kbps}{0.5263 bps/Hz} = 60.80 kHz \tag{5.11}$$

Recordemos que en banda base W corresponde a la mitad del ancho de banda en pasa banda, que es valor observado en el analizador de espectros. Por tanto, el ancho de banda total para este diseño es:

$$B = 2W = 2 \times 60.80 kHz = 121.6 kHz \tag{5.12}$$

5.2.7 Retardos de Transmisión

La configuración física para este experimento es similar a la mostrada en la Figura 13. con la única diferencia que en lugar de micrófono se envió señales de audio me-

diante un cable auxiliar. Los experimentos para el análisis de retardos de transmisión siguieron los pasos listados a continuación:

- Generar un archivo con un tono de audio con 0.5Vpp de amplitud.
- Situar los *USRP* a una distancia de 2m.
- Enviar el archivo por el puerto de audio de entrada el *USRP* Tx mediante un cable auxiliar. Este proceso también pudo ser realizado ingresando directamente el tono desde la interfaz de *GRC* mediante el bloque *Wav File Source*, sin embargo se perdería el sentido de experimentar el sistema de comunicaciones en su totalidad, ya que tendría que omitirse el bloque *Audio Source*.
- Grabar y guardar la señal de audio recibida en un archivo de *.wav* utilizando el bloque *Wav File Sink* de *GRC*.
- Comparar la diferencia en número de muestras de ambas señales y calcular esta diferencia en tiempo.

Los experimentos fueron realizados en el Ambiente 1.

Los resultados obtenidos reflejaron una diferencia de 330 muestras entre la señal enviada y recibida, como puede observarse en la imagen 60.

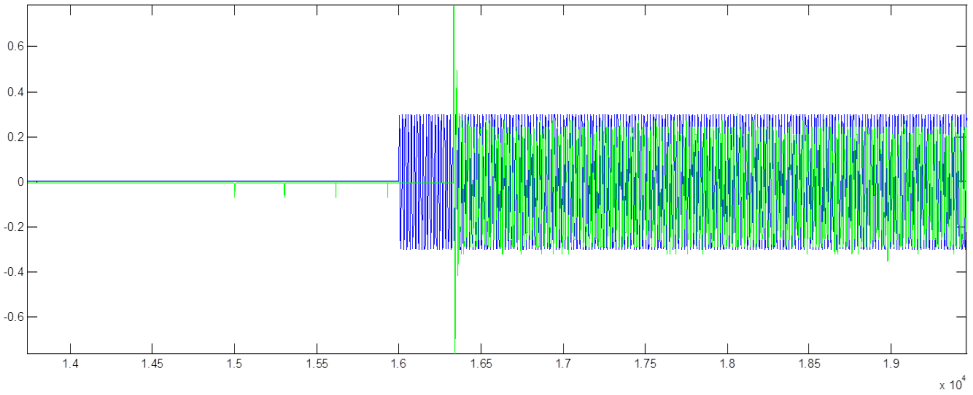


Figura 60: Retardo existente entre las señales Tx y Rx.

El tiempo de retardo de estas muestras incluye:

- Procesamiento del subsistema transmisor.
- Retardo en propagación.
- Procesamiento del subsistema receptor.

La frecuencia de muestreo como ya se ha dicho es 8kHz, lo que implica un total de 8000 muestras por segundo. Cada muestra tiene una duración D de:

$$D = \frac{1}{f_{muestreo}} = \frac{1}{8000} = 0.125ms \tag{5.13}$$

Entonces, si cada muestra dura 0.125ms y existen 330 muestras de retraso, el retardo total será la multiplicación entre ambos: 41.25 ms.

El retardo en tiempo de propagación para una distancia de 2m es prácticamente nulo dado que:

$$t_{propag} = c \times d, \tag{5.14}$$

donde c es la velocidad de la luz, y d la distancia entre dispositivos. Para 2 m el tiempo de propagación es 6.67 ns, que no llega a ser ni la duración de una muestra de la señal.

Si bien es cierto el retraso obtenido de 41.25 ms no es lo bastante grande como para ser percibido por el oído humano, sí representa una indicación de la carga computacional que pone el sistema sobre los *USRP*, dando muestras de que el procesador podría estar operando en los umbrales máximos de su capacidad.

Adicionalmente, pudo comprobarse este retardo mediante un osciloscopio al cual se le introdujo una señal senoidal a varias frecuencias con 0.5 Vpp de amplitud y se

contrastó con la salida del puerto de audio del *USRP* Receptor, los resultados obtenidos se muestran en la Figura 61.

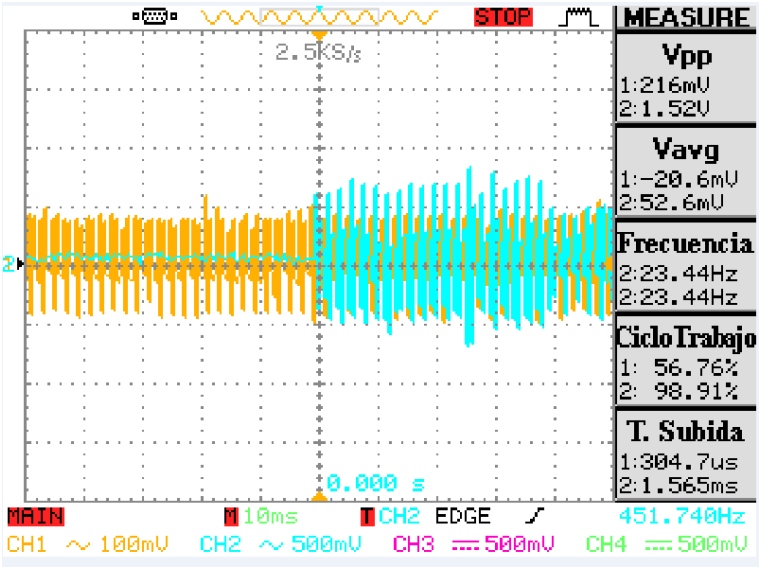


Figura 61: Retardo existente entre las señales Tx y Rx, tomado con osciloscopio.

En la Figura 61, el eje x se encuentra en una escala de 10ms por división, y puede verse que existe una diferencia de 4.8 divisiones, lo que representa alrededor de 48ms de diferencia entre ambas ondas; este valor es bastante similar al obtenido mediante los cálculos anteriores.

También se realizaron pruebas de transmisión utilizando dispositivos no embebidos *USRP N210* tanto como transmisor y receptor, lo experimentos abarcaron esquemas de modulación *GMSK*, *DBPSK* y *DQPSK*. Los resultados de estos experimentos se resumen en la sección de Anexos C.

5.2.8 Análisis de Desempeño MBSD

MBSD (Modified Bark Spectral Distortion) (Yang & Yantorno, 2000) es una medida de desempeño que estima la distorsión del habla teniendo en cuenta el umbral de enmascaramiento de ruido, con el fin de incluir sólo distorsiones en su cálculo.

El algoritmo presenta tres etapas en el tratamiento de las muestras: cálculo de sonoridad, cálculo de umbral de enmascaramiento de ruido, y el cálculo de *MBSD*.

Para el cálculo de sonoridad la señal de voz se procesa en varios pasos: análisis de banda crítica, igual intensidad de pre-énfasis e intensidad de volumen. El umbral de enmascaramiento de ruido se compara con el tono original y el de la voz codificada para determinar si la distorsión es perceptible. Finalmente, el cálculo de *MBSD* se define entonces como la media de la diferencia de sonoridad estimada que es perceptible.

Para el análisis de desempeño *MBSD* se ha hecho uso de funciones de correlación, no como medidas de calidad del sistema, sino para determinar el punto de mayor correspondencia entre las muestras de las señales Tx y Rx, con el fin de aplicar posteriormente el algoritmo *MBSD* para analizar el desempeño del sistema en función de la distorsión de la señal Rx con respecto a la Tx y la distancia entre dispositivos.

Las funciones de correlación permitirán determinar el punto en el cual existe una mayor correspondencia entre cada muestra de las señales Tx y Rx, para ello los siguientes pasos fueron ejecutados:

- Generar un archivo con un tono de audio con 0.5 Vpp de amplitud.
- Situar los *USRP* a una distancia de 0.5 m.
- Enviar el archivo por el puerto de audio de entrada el *USRP* Tx mediante un cable auxiliar. Este proceso también pudo ser realizado ingresando directamente el tono desde la interfaz de *GRC* mediante el bloque *Wav File Source*, sin embargo se perdería la naturaleza del experimento, ya que tendría que omitirse el bloque *Audio Source*.
- Grabar y guardar la señal de audio recibida en un archivo de *.wav* utilizando el bloque *Wav File Sink* de *GRC*.

- Aumentar la distancia entre dispositivos en pasos de 0.5 m.
- Analizar la correlación existente entre ambas señales para hallar el punto en el que las señales Tx y Rx poseen mayor correspondencia en sus muestras.

Los experimentos fueron realizados en el Ambiente 1.

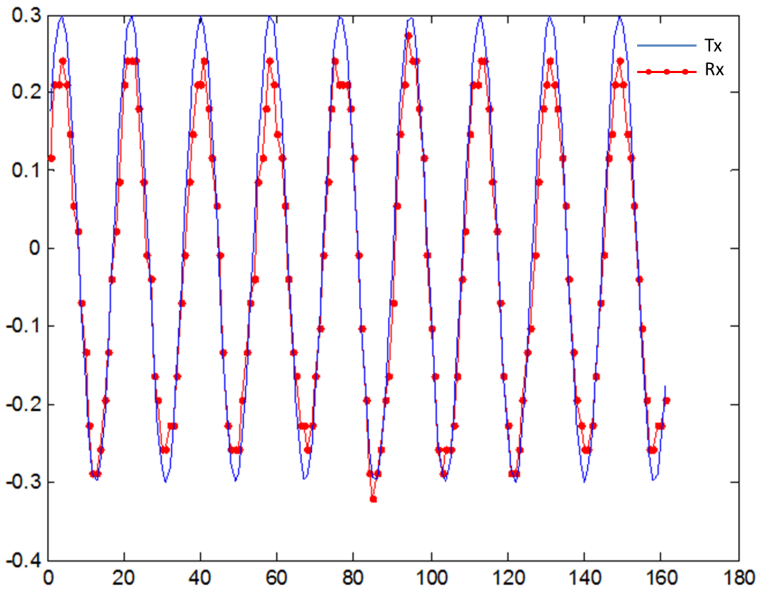


Figura 62: Desfase con tono de frecuencia y amplitud constante.

La Figura 62 muestra el comportamiento del sistema cuando se introduce un segmento de audio de frecuencia constante, como puede observarse, no existen mayores distorsiones en la señal Rx, siendo esta muy similar a la señal Tx. Por otro lado, al sustituir el tono por un segmento de voz los resultados son los observados en la Figura 63, aquí puede verse un segmento de voz de 20 ms en donde existen segmentos en los cuales la señal es bastante distorsionada con respecto a la original, y en otros segmentos ambas señales son similares.

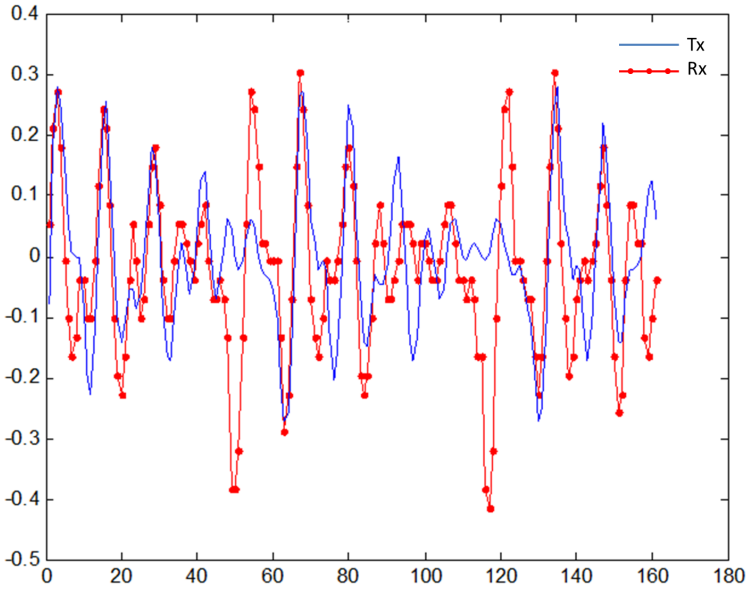


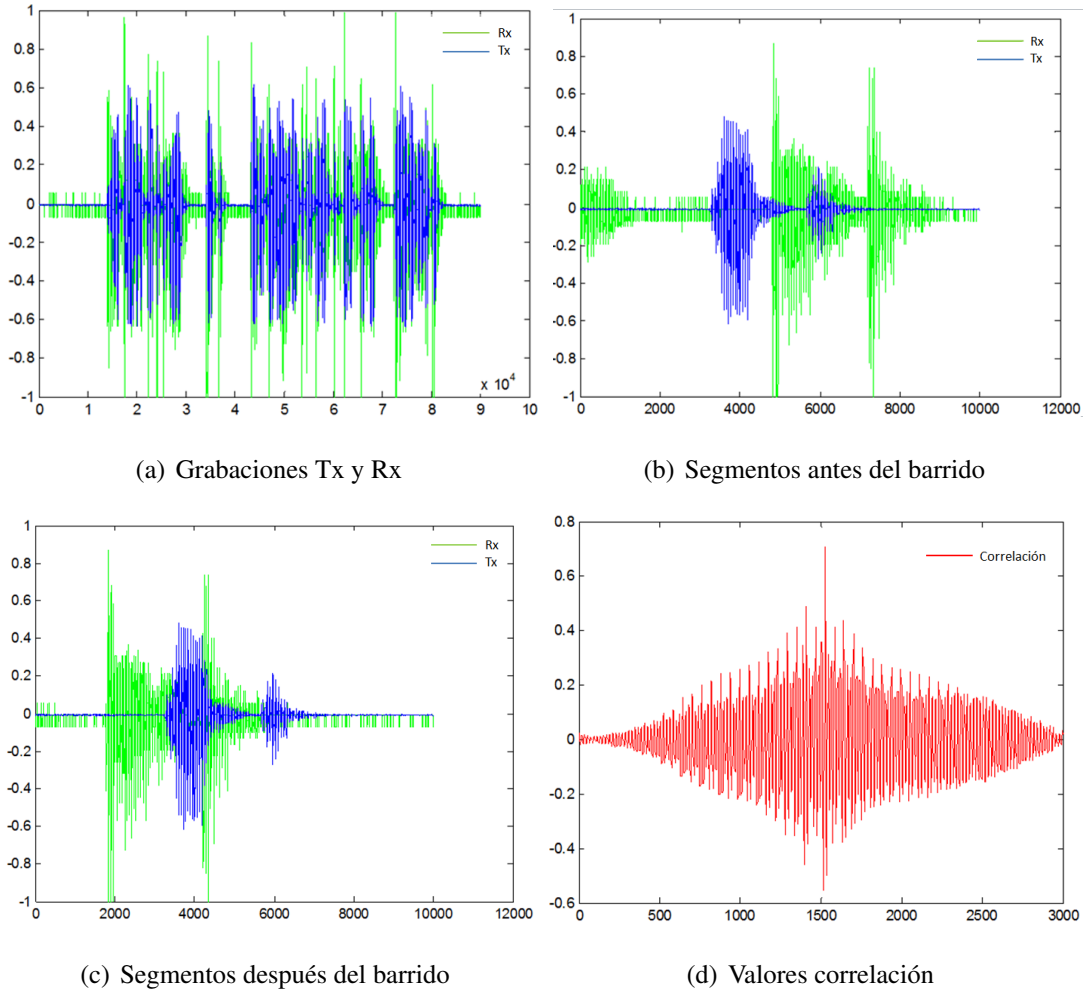
Figura 63: Desfase con muestras de voz.

Por estos motivos de distorsión de la señal Rx al momento de usar un segmento de voz se buscó una manera para determinar un punto en el cual ambos segmentos se encuentren lo más sincronizados posible (utilizando funciones de correlación), y a partir de ahí analizar el desempeño del sistema en función de las señales Tx y Rx.

Es posible que estos eventos de distorsión aleatorios encontrados en las muestras de voz recibidas sean producto de los procesos de cuantización, empaquetamiento y reconstrucción de los segmentos de datos manipulados por el sistema. Un análisis más detallado de las señales mostró que estos desfases se encontraban principalmente en regiones en donde la amplitud decaía a niveles muy bajos (silencios), comprobando lo mencionado con respecto al *DAC* del dispositivo en la sección consideraciones de diseño del Capítulo 4 del presente documento.

El proceso para hallar el mayor punto de correlación se realizó mediante un barrido muestra a muestra de una señal con respecto a otra estática y determinando la correlación entre ambas señales en cada paso. Se procedió a la realización del barrido

con cada par de muestras de audio grabadas a distintas distancias y se determinó la máxima correlación existente entre cada una. El proceso realizado se describe en la Figura 5.2.8.



(a) Grabaciones Tx y Rx

(b) Segmentos antes del barrido

(c) Segmentos después del barrido

(d) Valores correlación

Figura 64: Proceso de barrido para hallar la máxima correlación entre segmentos de audio Tx y Rx.

Una vez determinados los puntos de mayor correspondencia entre las señales de Tx y Rx a distintas distancias se aplicó el algoritmo *MBSD*, obteniéndose los resultados mostrados en la Figura 65.

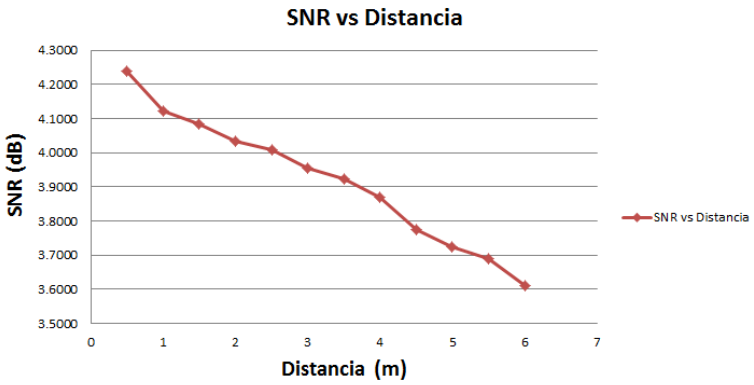


Figura 65: Resultados MBSD a varias distancias.

Los resultados obtenidos del algoritmo *MBSD* indican que el desempeño del sistema disminuirá en tanto la distancia entre dispositivos *USRP* se incremente, hasta eventualmente en algún punto determinado perder la comunicación.

5.3 Evaluación de Parámetros Subjetivos

5.3.1 Encuesta *Mean Opinion Score*

Dentro de los factores subjetivos que se utilizan para medir la calidad de un sistema de comunicaciones se encuentra el *Mean Opinion Score*. El *MOS* es un factor que puede trasladar la percepción subjetiva del usuario a una expresión numérica cuantificable basada en la calidad de lo que escucha en el sistema luego de que la señal ha sido transmitida y eventualmente comprimida usado códecs.

El uso del *MOS* como tipo de evaluación nace de la naturaleza propia de la voz humana y su percepción sensorial por el aparato auditivo. Al realizar procesos de filtrado, modulación, y sobre todo compresión de la voz es imprescindible pensar en un método de evaluación de la señal recibida. Una encuesta *MOS* proporciona información suficiente para cualificar la calidad de la señal de voz que escucha el usuario (Nadeem, 2013).

El *MOS* puede ser expresado en un número entero, que va desde 1 a 5, siendo 1 el peor y 5 la mejor calificación a la calidad. La Tabla 13 detalla la calificación y la descripción de calidad asociada a esta.

Tabla 13: Calificación MOS. (Nadeem, 2013)

	Calificación	**Descripción**
1	Excelente	Similar a la comunicación en persona.
2	Bueno	Puede ser que se reciba alguna interferencia, pero en general el sonido es claro.
3	Aceptable	Puede escucharse la comunicación, pero existen factores que generan interferencia.
4	Malo	Casi imposible escuchar la comunicación.
5	Pésimo	Imposible escuchar la comunicación.

La subjetividad que conlleva el uso de *MOS* como elemento de evaluación depende del aparato auditivo de cada persona sometida a la encuesta, y de esto desemboca el pensar en varios criterios particulares a considerar para elegir al banco de participantes.

Los factores que influyen en las opiniones sobre la calidad del sistema incluyen elementos subjetivos que podrían tener una influencia directa sobre el resultado de la evaluación, como por ejemplo:

- La afinidad del usuario sometido a la encuesta con la muestra de audio y la voz del locutor que se encuentre en la sección de Tx.

- El género de la persona que recibe los mensajes sintetizados. Estudios indican que por razones evolutivas las mujeres tienden a escuchar con mayor facilidad sonidos más agudos, diferenciar tonalidades en el volumen de la voz mejor que los hombres, y son capaces de discernir los sonidos para categorizarlos en su cerebro (Pérez F., 2013).

- La edad del encuestado. Este parámetro influirá en gran medida en los resultados puesto que el aparato auditivo merma su capacidad conforme la edad del individuo, quien podría no discernir con claridad la señal de audio que percibe.

- El estado de ánimo del locutor, ya que por características de entonación y por la arquitectura del sistema equipado con un códec de voz podría producir una voz ligeramente robotizada o de un acento particular.

- La apertura del oyente al entendimiento y no a la crítica destructiva.

Normalmente el resultado definitivo se basa en la democracia, y se centra en la opinión de un grupo de personas sometidas a la encuesta, una vez obtenidos los resultados de las opiniones recogidas, estos serán promediados para obtener un estimado de la calidad del sistema. Para la realización de este experimento se procedió de la siguiente manera:

- Elegir un ambiente cerrado y tranquilo para la realización de la encuesta de *Mean Opinion Score*.

- Situar los dispositivos *USRP E110* a una distancia de 2 metros.

- Pedir al individuo escuchar la siguiente oración en el transmisor: *"Universidad Politécnica de las Fuerzas Armadas, transmisión de voz utilizando dispositivos periféricos USRP E110"*.

- Una vez que el encuestado haya escuchado y se haya familiarizado con la versión original de la frase que será transmitida se le pedirá escuchar nuevamente el segmento de audio pero esta vez en el receptor.

- Pedir al encuestado calificar la calidad del segmento de audio escuchado en función del original.

El universo de encuestados constó de 30 de personas de distintos sexos y edades varias, los resultados de la evaluación del sistema basado en la calificación asignada por los encuestados se muestran en la Figura 5.3.1. La Figura 5.3.1 agrupa la valoración del sistema en formato de porcentajes. Cabe resaltar que ninguno de los encuestados ha valorado la transmisión escuchada como pésima.

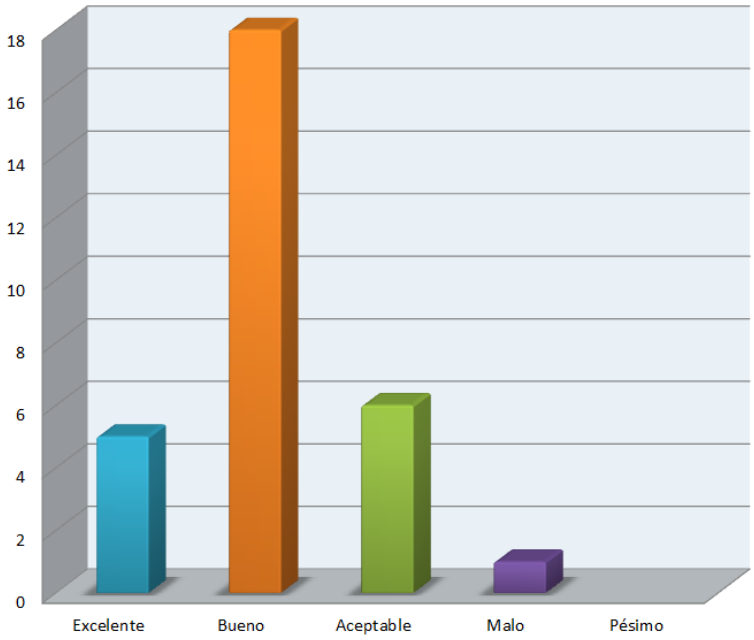


Figura 66: Evaluación MOS. El eje Y corresponde al número de personas y el eje x a la calificación asignada al sistema.

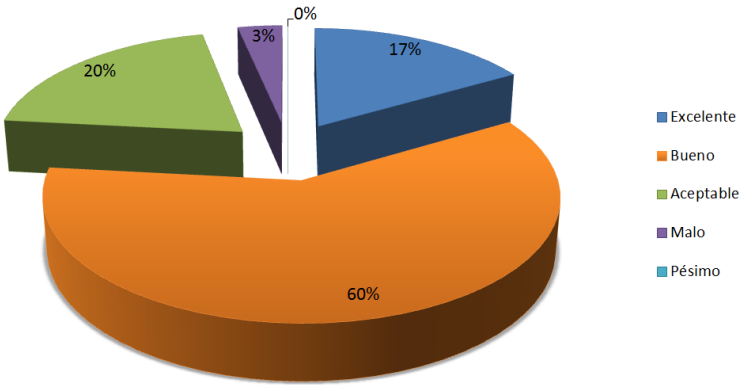


Figura 67: Evaluación MOS en porcentaje.

En general, el sistema de transmisión de voz mediante dispositivos *USRP E110* valorado por un universo de treinta personas obtuvo una calificación en promedio de 3.9/5, lo cual bastante es aceptable para un sistema de transmisión de voz que hace uso de un algoritmo de compresión de voz como es el vocoder G.721, el cual posee una valoración de 3.85/5 según estándares de telefonía *VoIP* (VoipForo, 2013b).

5.3.2 Test de Intelegibilidad Segmental

Se ha creído conveniente que para una mejor evaluación de la calidad de audio recibida en el sistema se agregue otros tipos de pruebas subjetivas que permitan detectar errores y evaluarlos comparativamente, para ello se ha recurrido a pruebas de intelegibilidad segmental e intelegibilidad en contexto orientadas al idioma español.

Las pruebas de intelegibilidad segmental requieren de una mayor información fonética y se basan en dos puntos fundamentales para la evaluación:

- Inteligibilidad de los elementos segmentales.
- Inteligibilidad de las combinaciones vocálicas.

Estos dos puntos en conjunto se conocen también como Test de Rimas ó *RT, RhymeTest* que en su versión más actual y utilizada se conoce como Test de Rimas Modificado *MRT, Modified Rhyme Test*; fue diseñado por Fairbanks en 1958 y modificado por House en 1965. Se trata de un test formado por estímulos consistentes en palabras monosilábicas con la estructura consonante-vocal-consonante, en el que los oyentes deben elegir una palabra entre seis alternativas (Llisterri J., 1991) . Actualmente, el test de rimas modificadas se utiliza para la medición subjetiva de sistemas telefónicos y de generación de voz artificial (Briceño E, 2010).

Las palabras difieren en un único segmento, que se encuentra o en posición inicial o en posición final; una de las ventajas de este test es que ha sido utilizado en el

habla natural, y existen por lo tanto medidas estandarizadas que pueden utilizarse como punto de referencia.

El requisito de la monosilabicidad (una sílaba) introduce problemas importantes en cuanto a la percepción de sonidos por el aparato auditivo puesto que al adaptar este test al idioma español se ha tenido que recurrir a palabras poco familiares o reducirse el número de alternativas ante la imposibilidad de encontrar seis palabras que sólo difieran en la consonante inicial o en la final. Por este motivo, el número de alternativas en la respuesta ha sido reducido a 5.

Para la selección de los monosílabos, deberá tomarse muy en cuenta la mayor o menor frecuencia de aparición de las consonantes, tratando de que exista una relación proporcional a la que se encuentra en el idioma castellano. La sección de Anexos C. muestra el modelo de encuesta realizado en esta investigación.

La realización de este experimento se realizó de la siguiente manera:

- Elegir un ambiente cerrado y tranquilo para realizar el test de inteligibilidad segmental.
- Situar a los dispositivos *USRP E110* a una distancia de 2 metros.
- El encuestado recibirá una hoja de test conteniendo una lista de monosílabos, existiendo cinco opciones distintas con variaciones consonánticas tanto iniciales como finales para cada monosílabo.
- Situar al individuo en la sección del receptor. Se reproducirá tres archivos de audio distintos, conteniendo veinte monosílabos cada uno.
- El primer audio evaluará sílabas con un cambio en la consonante final, el segundo trabajará con cambios en la consonante inicial y finalmente el tercero lo hará con grupos consonánticos.

- El encuestado deberá seleccionar de entre las cinco opciones la que considere se acerque más a la que escuchó en el receptor y marcarlo en la hoja de respuestas del test.

Como resultados, de un universo de veinte personas de ambos sexos y distintas edades sometidas al test de inteligibilidad segmental pudo determinarse el comportamiento del sistema frente a estímulos segmentales y la respuesta del oído del encuestado a ciertas variaciones consonánticas similares a las usadas en el idioma español. Para la evaluación, los resultados serán divididos con base en los segmentos de audio escuchados.

El audio correspondiente a la variación de la consonante final fue el primero a ser reproducido dentro del test realizado, los resultados arrojaron que las personas encuestadas no presentaron ningún problema en comprender segmentos silábicos con una gran carga energética de voz en la consonante final, como son los casos de sílabas como “car”, “sul” y “jal” cuyo porcentaje de aciertos es del 100%. Por otro lado, se encuentran sílabas con consonantes cuya carga energética es mucho menor en comparación con las anteriores, como es el caso de “fed”, “cad” y “ref” las cuales lograron porcentaje de aciertos de entre 35% a 45%, a menudo el oído humano tiende a confundir consonantes con baja energía como las letras ‘D’, ‘T’, ‘F’ o ‘S’. Las demás sílabas obtuvieron un porcentaje de aciertos del 60% en adelante, lo que significa que los encuestados no tuvieron mayores dificultades en entender estas composiciones silábicas. La Figura 5.3.2 muestra los porcentajes de aciertos y la sílaba respectiva asociada a este porcentaje.

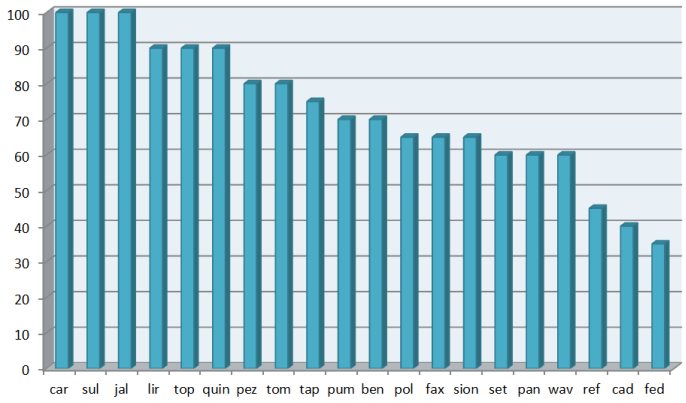


Figura 68: Variación de la consonante final.

Los resultados correspondientes al segmento de audio en el cual se varían las consonantes iniciales son mucho mejores. Se ha obtenido un mayor número de sílabas con un porcentaje de acierto del 100% incluso en combinaciones silábicas cuya consonante inicial es una letra con baja energía de voz, como por ejemplo la letra 'F'. La Figura 5.3.2 ilustra los resultados porcentuales del acierto de cada combinación silábica.

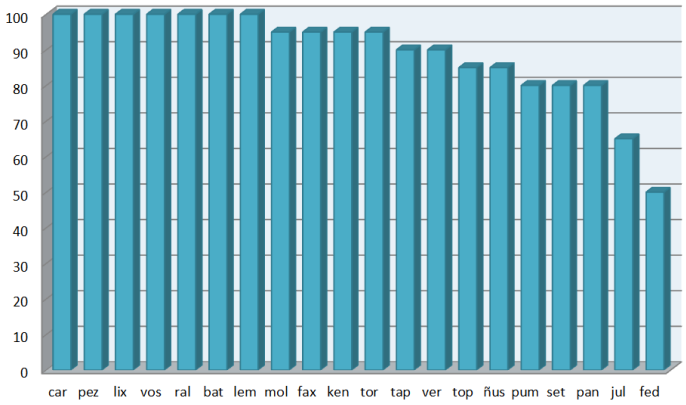


Figura 69: Variación de la consonante inicial.

En lo referente a la evaluación de grupos consonánticos pudo determinarse que el procentaje de aciertos disminuye en sílabas que contienen consonantes de baja energía como la letras 'B' o 'F'. Un resultado que resalta de este grupo es que el menor porcentaje de aciertos corresponde a la sílaba "prac" la cual fue confundida en su gran

mayoría por “brac”, podemos atribuir este error de comprensión al hecho de que al momento de producir sonidos como la letra ‘P’ se genera un silbido de baja energía que se encuentra en el rango de frecuencias que el filtro no discrimina, por tanto este sonido es amplificado por el sistema de radio y ha causado una ligera distorsión al momento de su transmisión. La Figura 5.3.2 muestra el porcentaje de aciertos en la evaluación de grupos consonánticos.

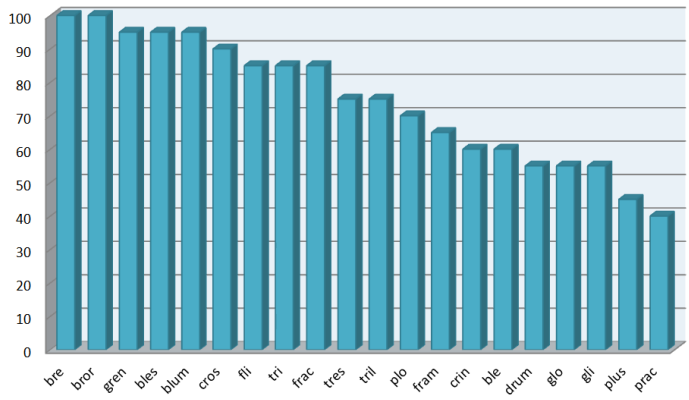


Figura 70: Grupos consonánticos.

Las pruebas de inteligibilidad segmental han ofrecido una indicación de la comprensión en la recepción silábica de palabras aisladas, pero aun así no se tiene información más detallada cuando se generan frases con un contexto específico. Para la evaluación de este punto se ha recurrido a la realización del mismo experimento utilizando oraciones y frases en contexto, el propósito de este experimento será evaluar si existe dificultad al interpretar frases que contengan sílabas con baja energía de voz. Recordemos que los resultados de las pruebas de intelegibilidad segmental arrojaron porcentajes de acierto buenos pero no excelentes en la comprensión de estas sílabas.

Para la realización de este experimento se hizo uso de frases psicoacústicas de *Hardward*, desarrolladas por Egan en 1948 para aplicaciones audiométricas (Linguistics & Faculty, 2010). Su característica principal es que las frases generadas tienen sentido, poseen una sintaxis correcta y son fonéticamente equilibradas. Tomaremos

estas frases adaptadas al idioma español para la realización de las pruebas pertinentes.

Para la realización de este experimento se procedió de la siguiente manera:

- Ha de elegirse un ambiente cerrado y tranquilo para la realización del test de inteligibilidad segmental.
- Se situará los dispositivos *USRP E110* a una distancia de 2 metros.
- Se pedirá al individuo escuchar la frase en la sección del receptor y escribirla para posteriormente analizar si la frase ha sido comprendida en su totalidad.

Se procuró hacer uso de frases que contengan fonemas de baja energía. Las oraciones usadas fueron:

- "Nunca debí dinero, ni lo tomé prestado".
- "No des patadas a los rosales".
- "Me diste la paga de este mes".

Del banco de individuos encuestados, ninguno de ellos tuvo dificultades para entender el contexto de la oración, aun cuando la frase contenía varias sílabas con consonantes de baja energía. Lo que demuestra que cuando se formulan oraciones en contexto el oyente es capaz de entender completamente el mensaje transmitido. Además, ninguno de los participantes de la encuesta afirmó haber escuchado el mensaje con un tono de voz metalizado que podría ser producto del compresor G.721 y ninguno de ellos reportó haber tenido que realizar esfuerzos para entender las oraciones.

CONCLUSIONES

La frecuencia de la voz humana posee un rango de frecuencias de entre 500Hz a 3500Hz. Se eligió una frecuencia de muestreo de 8000Hz para cumplir requerimientos de la frecuencia de Nyquist. A esta frecuencia de muestreo la señal de voz presenta una resolución lo suficientemente buena sin comprometer las capacidades de procesamiento del *Hardware*.

La inclusión de un bloque *vocoder* permitió reducir significativamente la carga computacional de las muestras sobre el sistema ya que mediante este proceso la voz es sintetizada para trabajar con su frecuencia fundamental.

El esquema de modulación digital *GMSK* introduce *ISI* en la señal de entrada debido a que ensancha el pulso a varios tiempos de bit, y a su vez los solapa unos con otros. Se concluye que si el BT del filtro gaussiano se encuentra entre 0.3 y 1, esta interferencia intersimbólica no es tan grave para la recuperación de la información. En cambio, la eficiencia espectral que se consigue al implementar el filtro gaussiano es mayor que la posible pérdida de información al contener errores. Es por ello que se ha utilizado un BT=0.35 para la transmisión, considerando que dadas las características de las bandas de frecuencia de las *Daughterboards* y las características de la señal a transmitir (voz) se prefirió sacrificar el ancho de banda a fin de no tener pérdidas en la información.

La inclusión de un bloque de codificación de canal procedente de la librería de

GNU Radio incrementó la cantidad de muestras procesadas, lo que imposibilitó a los equipos mantener la comunicación. Al final estos bloques fueron omitidos en el diseño final del sistema.

El tiempo de procesamiento de la señal desde que es muestreada en el Tx hasta su salida por el puerto de audio en el Rx es de 41.25ms. Este tiempo es aceptable para comunicaciones en tiempo real ya que el oído humano es capaz de percibir como retardo una diferencia temporal mayor a 150ms (fiwiki, 2012).

El parámetro que configura la ganancia de la antena en los bloques de interfaz *UHD USRP Sink/Source* permite regular la salida en potencia de la tarjeta *Daughterboard*, funcionando como amplificador o atenuador lineal.

En general, el sistema fue calificado como "Bueno" por la encuesta MOS. Se presentaron bajos porcentajes de acierto para los test de intelegibilidad segmental en combinaciones vocálicas de baja potencia, sin embargo, esto no fue un inconveniente al momento de entender un mensaje en contexto.

La capacidad de procesamiento del dispositivo embebido *USRP E110* no hizo posible la implementación de diseños utilizando esquemas de modulación más complejos que GMSK.

6.1 Recomendaciones

Dada la potencia de salida de las *Daughterboards* no se han logrado distancias significativas para la transmisión, por lo que se recomienda el uso de amplificadores de señal para evaluar el desempeño del sistema en ambientes distantes.

Se recomienda la inclusión de etapas de filtrado de la señal tanto en la sección Tx como en Rx ya que los mismos factores de propagación en el medio, además de

los aparatos introducen ruido térmico, de procesamiento y de propagación, que resta calidad a las muestras de audio transmitidas.

En la sección Tx se recomienda el uso de un micrófono activo para la captura de la señal. Tanto en Rx como Tx ha de tenerse especial cuidado en la adaptación de conectores a los puertos, ya que pueden incrementar significativamente la cantidad de ruido introducido en el sistema.

6.2 Trabajo Futuro

El proyecto presentado ha abierto paso a varias otras líneas de investigación con dispositivos de radio definido por *Software*, como son la generación de bloques personalizados para la implementación de codificación de fuente y codificación de canal, el diseño de amplificadores de potencia exclusivos para dispositivos SDR, etc.

En conclusión, el campo de *SDR* aún no ha sido explorado lo suficiente, queda aún la implementación de una red distribuida de sensores de RF, investigación en el campo de radares, *Open BTS* y una gran cantidad de proyectos relacionados con esta línea de investigación.

BIBLIOGRAFÍA

Adrián de Pérez, T. (2003). *Cuantificación.* (Extraído Abril 2014)

Angulo Hugo, P. D. (2011). *Diseño y desarrollo de un radio definido por software, para el ejército ecuatoriano, mediante la utilización de una tarjeta usrp y la herramienta simulink de matlab* (Doctoral dissertation, Universidad de las Fuerzas Armadas - Espe). Retrieved from http://repositorio.espe.edu.ec/handle/21000/4526

Araseki, T., & Ozawa, K. (1982). *Adpcm system for speech or like signals.* Google Patents. Retrieved from http://www.google.com/patents/US4354273 (US Patent 4,354,273)

Blossom, E. (2009). *Exploring gnu radio.* Retrieved from http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html (Extraído Marzo 2013)

Briceño E, C., Carrión. (2010). *Implementación hardware en una fpga de una red neuronal artificial para el reconocimiento de patrones de voz específicos* (Unpublished doctoral dissertation). Universidad Técnica Particular de Loja.

Cordoba, U. (2003). *Ondas electromagnéticas, conceptos básicos.* (Extraído Marzo 2014)

D., D. (2012). *Ejecutar aplicaciones automáticamente al iniciar sesión.* Retrieved from http://www.tuxylinux.com/ejecutar-aplicaciones-automaticamente-al-iniciar-sesion/ (Extraído Abril 2014)

Digital communication systems engineering with software-defined radio. (2013). Artech House.

Distribution, A. (2013). *Building anstrom.* Retrieved from http://www.angstrom-distribution.org/ (Extraído Marzo 2014)

EA1DDO. (2012). *Sdr - software defined radio.* (Extraído Octubre 2013)

EttusResearch. (2012a). *Selecting a usrp device.* Retrieved from http://www.ettus.com/content/files/kb/application_note_selecting_a_usrp.pdf (Extraído Febrero 2014)

EttusResearch. (2012b). *Uhd - usrp-e1x0 series device manual.* (Extraído Enero 2014)

EttusResearch. (2012c). *Usrp e110 datasheet.* (Extraído Febrero 2014)

fiwiki. (2012). *Aplicaciones multimedia en tiempo real.* (Extraído Marzo 2014)

Forum, S. (2007). *Sdrf cognitive radio definitions working document.* Retrieved from http://www.wirelessinnovation.org/IntroductiontoSDR (Extraído Marzo 2013)

Forum, S. (2009). *Introduction to sdr.* Retrieved from http://www.wirelessinnovation.org/IntroductiontoSDR (Extraído Marzo 2013)

Gerald Youngblood, A. (2002). *A software defined radio for the masses.* Retrieved from http://www.exradio.com/Data/Doc/qex1.pdf (Extraído Marzo 2013)

GNURadio. (2014). *Gnu radio, the free and open software radio ecosystem.* (Extraído Febrero 2014)

Group, I. W. (2012). *Ieee standard for information technology.* Retrieved from http://www.ieee802.org/11/ (Extraído Marzo 2014)

Herriko, U. E. (2003). *La voz humana.* (Extraído Marzo 2014)

Jeff, F. (2002). Practical costas loop design. *RF Signal Processing*, 1459-1460. Retrieved from http://defenseelectronicsmag.com/sitefiles/defenseelectronicsmag.com/files/archive/rfdesign.com/images/

archive/0102Feigin20.pdf

Koolwal, K. (2012). *How to lauch programs scripts on lxde startup.* Retrieved from http://linux.koolsolutions.com/2009/09/01/howto-auto-launch-programsscripts-on-lxde-startup/ (Extraído Abril 2014)

Lee, J.-S. (2004). *Implementation of dsp-based digital receiver for the sdr application* (Unpublished doctoral dissertation). Dept. of Electron. Eng., Chung-Buk Nat. Univ.

Linguistics, C., & Faculty, S. L. (2010). *Harvard psychoacoustic sentences.* (Unpublished doctoral dissertation). University of Bielefeld.

Llisterri J., D., Poch. (1991). *Caracterización fonética del bilingüismo, análisis acústico del habla espontánea y evaluación de sistemas de síntesis del habla* (Unpublished doctoral dissertation). Universidad Autónoma de Barcelona.

LPI. (2014). *Teorema del muestreo.* (Extraído Marzo 2014)

Manandayam, N. B. (2012). *Advanced topics in communication engineering.* (Extraído Abril 2014)

Mitola, J. (2000, 8 de Mayo). *Cognitive radio: An integrated agent architecture for software defined radio.* (Royal Institute of Technology (KTH), Estocolmo, Suecia)

Nadeem, U. (2013). *Mean opinion score (mos) a measure of voice quality.* (Extraído Marzo 2014)

Pérez F., B. C. (2013). *Fisiología sensorial ii. audición, gusto y olfato.* (Extraído Marzo 2014)

Python, L. (2014, Abril). *Python.* (Sitio Web)

Quiroz, J. (2010). *Emulador de un sistema de comunicaciones utilizando tecnología sdr* (Unpublished doctoral dissertation). Universidad de las Fuerzas Armadas - Espe.

Radio, G. (2014, Abril). *Gnu radio.* (Sitio Web)

Research, E. (2013). *Users faq.* Retrieved from http://code.ettus.com/redmine/ettus/projects/usrpe1xx/wiki/FAQ (Extraído Marzo 2014)

R. I. Lackey, D. W. U. (1995). *Speakeasy: The military software radio.* (Extraído Marzo 2013)

Schiphorst, R. (2000). *Demonstration of the software-radio concept* (Unpublished doctoral dissertation). Universidad de Twente.

Tech, W. (2002). *Software defined radio: A technology overview.* (Extraído Enero 2014)

Tomislav, S., & Marijan, H. (2008, 12). A simple signal shaper for gmsk/gfsk and msk modulator based on sigma-delta look-up table. *Innovations*, 230-236. Retrieved from http://www.radioeng.cz/fulltexts/2009/09_02_230_237.pdf

Torres, J., & Paz, H. (2012). *Estudio y comparación en eficiencia espectral y probabilidad de error de los esquemas de modulación gmsk y dbpsk.* (Extraído Abril 2014)

Torres Nova, J. M., & Paz Penagos, H. (2008, 12). Estudio y comparación en eficiencia espectral y probabilidad de error de los esquemas de modulación gmsk y dbpsk. *Ingeniería e Investigación*, *28*, 75 - 80. Retrieved from http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-56092008000300010&nrm=iso

Tranter, W. (1998). *An overview of the virginiatech program in software radios implemented with reconfigurable computing.* (Instituto Politécnico de Virginia y Universidad Estatal Blacksburg)

Valero, L. (2013). *Curso audiovisual interactivo de codificación de voz.* Retrieved from http://ceres.ugr.es/~alumnos/luis/c_voz.htm (Extraído Abril 2014)

Vijamaro. (2009). *Cambiar inicio automático de ubuntu sin entorno gráfico.* Retrieved from http://vijamaroylinux.blogspot.com/2009/08/cambiar

-inicio-automatico-de-ubuntu-sin.html (Extraído Abril 2014)

Vilchez, L. (2011). *Inicio de sesión automático en ubuntu.* Retrieved from http://www.elblogdejabba.com/2011/05/inicio-de-sesion-automatico-en-ubuntu.html (Extraído Abril 2014)

VoipForo. (2013a). *Funcionamiento de un codec.* (Extraído Marzo 2014)

VoipForo. (2013b). *Tabla resumen de codecs.* Retrieved from http://www.voipforo.com/codec/codecs.php (Extraído Marzo 2014)

Wurzburg, U. (2013). *Understanding fft windows.* (Extraído Marzo 2014)

Yang, W., & Yantorno, R. (2000). Comparison of two objetctive speech quality measures: Mbsd and itu-t recommendaion p861. *Electrical and Computer Engineering*.

Youngblood, G. (2003). *A software-defined radio for the masses.* (Extraído Abril 2013)

ANEXOS A

HARDWARE Y SOFTWARE

Tabla 14: Librerías GNU Radio (Parte 1).

	Librerías GNU Radio Project
gr	Módulo principal de GNU Radio, se necesitará prácticamente en todos los diseños. Contiene bloques básicos como fuentes, sumideros, etc.
digital	Contiene librerías y archivos para funciones de modulación y demodulación digital
audio	Control de la tarjeta de audio. Permite enviar y recibir ondas sonoras.
blocks	Contiene bloques de procesamiento usados en los diagramas de flujo.
blks2	Contiene bloques adicionales escritos en Python.
trellis	Para codificaciones convolucionales.
analog	Archivos relativos a las modulaciones analógicas.
wavelet	Bloques para transformadas wavelet.
fft	Bloques para *Fast Fourier Transform*.
window	Rutinas para el diseño de ventanas.
optfir	Rutinas de diseño óptimo de filtros FIR.
filter	Bloques para operaciones de filtrado.
qtqui	Módulo conteniendo sumideros gráficos basados en QT.
qtqui	Módulo conteniendo GUI basada en WX.
grc	Módulo para interfaz gráfica gnuradio-companion.
video_sdl	Control para envío y recepción de señales de video.
vocoder	Bloques de procesamiento para implementación de vocoders.
uhd	Módulo para interfaz a la librería UHD para Tx y Rx de datos de los USRP.
How to write a block	Contiene información para crear nuevos módulos e incluirlos al proyecto GNU Radio.

Tabla 15: Librerías GNU Radio (Parte 2).

	Librerías out-of-tree
osmosdr	Soporte para el uso del Hardware osmoSDR.
baz	Agrega nuevas funcionalidades de soporte al proyecto GNU Radio.

Tabla 16: Estructura de un Módulo GNU Radio.

Carpetas	**Descripción**
Apps	Contiene ejemplos y aplicaciones de prueba del módulo.
Cmake	Contiene archivos de configuración para la instalación correcta del módulo.
GRC	Contiene distintos archivos ".xml" de los bloques para usarlos en la aplicacion GRC.
Include	
Lib	Contiene archivos fuente ".cc" de los bloques de procesado.
Python	Contiene scripts de Python.
Swig	Contiene archivos swig ".i" con la configuración del intérprete de C++ y Python.

ANEXOS B

DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

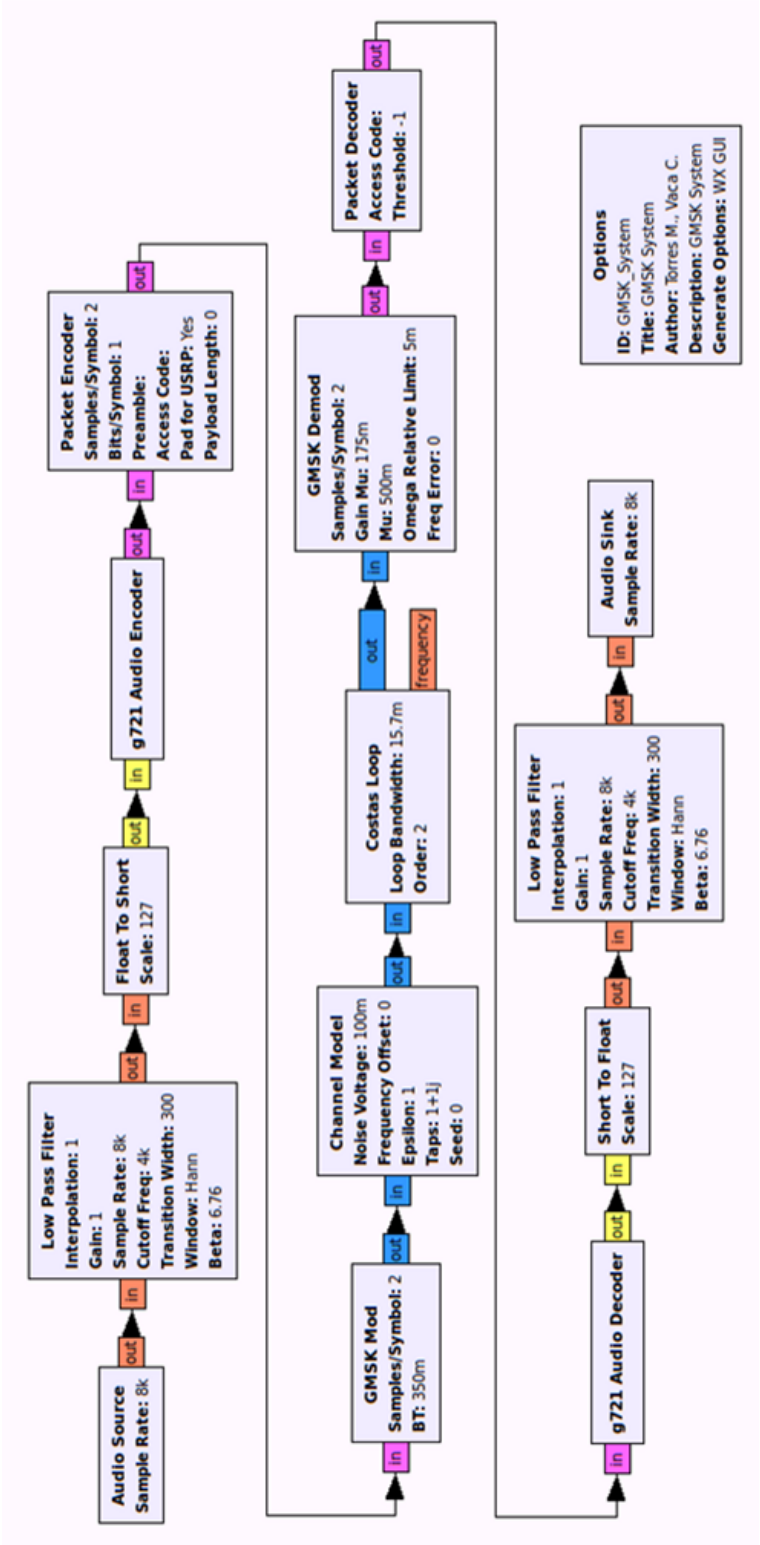


Figura 71: Sistema de comunicaciones con bloque de simulación de canal.

ANEXOS C

PRUEBAS Y RESULTADOS

Tabla 17: Modelo de encuesta intelegibilidad segmental para cambios en la consonante final.

Original	Opciones				
car	cas	car	cal	can	cam
pez	per	pet	pel	pen	pez
lir	lix	lit	lil	lir	lim
pol	pom	pot	pop	por	pol
fax	fas	fat	fax	fan	fal
pum	pun	pub	pum	pul	pur
tap	tab	tac	tat	tap	tam
sul	sur	sum	sun	sul	suc
set	sep	seb	sel	set	sec
pan	par	pam	pan	pal	pav
top	tos	toc	tol	tom	top
fed	fet	fem	feb	fed	fes
jal	jam	jas	jac	jat	jal
ben	bem	bel	bep	ben	ber
sion	siom	sion	sios	siop	siot
ref	res	ref	reb	ren	rep
quin	quid	quin	quim	quit	quil
cad	cad	cat	cap	cab	cac
wav	wap	wav	wal	wan	wad
tom	ton	tob	tom	top	tor

Tabla 18: Modelo de encuesta intelegibilidad segmental para cambios en la consonante inicial.

Original	Opciones				
car	par	car	tar	far	jar
pez	fez	mez	pez	lez	kez
lix	lix	pix	nix	mix	fix
mol	nol	pol	mol	col	bol
fax	bax	sax	fax	pax	lax
pum	bum	pum	gum	wum	tum
tap	pap	dad	bap	tap	lap
jul	ful	sul	jul	gul	pul
set	set	pet	met	fet	vet
pan	ban	dan	tan	pan	can
top	tos	toc	tol	tom	top
fed	fev	fem	feb	fed	fes
ken	ben	ken	len	pen	ten
vos	vos	pos	mos	los	ros
ver	per	mer	ner	ser	ver
ñus	nus	yus	ñus	tus	xus
ral	pal	fal	ral	gal	mal
bat	pat	bat	mat	tat	gat
lem	yem	pen	lem	rem	bem
tor	por	tor	for	mor	dor

Tabla 19: Modelo de encuesta intelegibilidad segmental para grupos consonánticos.

Original	Opciones				
tres	pres	fres	cres	gres	tres
crin	crin	prin	trin	frin	grin
plo	flo	blo	plo	clo	glo
bre	pre	fres	tre	gre	bre
drum	brum	prum	frum	prum	drum
fli	bli	pli	fli	gli	cli
fram	bram	dram	pram	fram	tram
gren	pren	tren	cren	gren	bren
glo	plo	blo	tlo	flo	glo
tri	fri	vri	tri	cri	pri
plus	glus	blus	flus	clus	plus
ble	ple	cle	gle	fle	ble
cros	bros	tros	pros	fros	cros
prac	trac	frac	brac	grac	prac
tril	tril	cril	gril	pril	bril
frac	grac	brac	frac	crac	trac
bror	pror	bror	fror	gror	tror
bles	ples	bles	fles	cles	gles
blum	plum	blum	clum	flum	glum
gli	bli	pli	cli	gli	fli

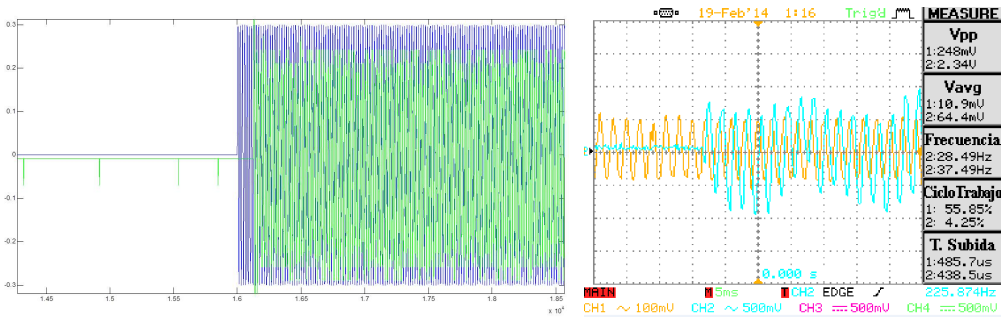
C.1 Pruebas de transmisión con dispositivos no embebidos

Los experimentos llevados a cabo se resumen en la Tabla 20.

Tabla 20: Experimentos con dispositivos USRP no embebidos.

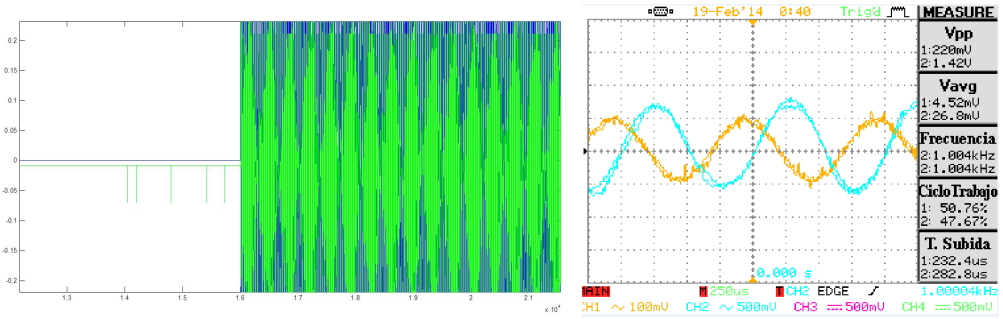
Transmisor	Receptor	Modulación
E110	N210	GMSK
E110	N210	DBPSK
E110	N210	DQPSK
N210	N210	GMSK
N210	N210	DBPSK
N210	N210	DQPSK

Las Figuras C.1 y C.1 muestran segmentos de audio y presentan gráficamente su retardo. La Tabla 21 presenta los esquemas de modulación utilizados y la función de cada dispositivo en los distintos experimentos de transmisión.



(a) Retardo obtenido en Matlab (b) Retardo obtenido en osciloscopio

Figura 72: USRP E110 como Tx y USRP N210 como Rx. Modulación GMSK.



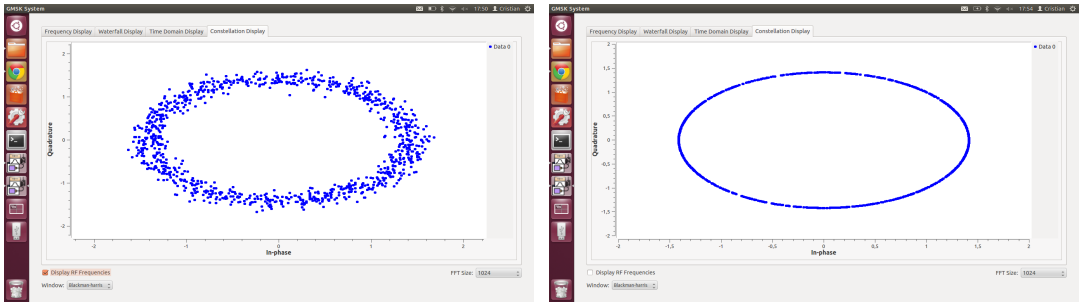
(a) Retardo obtenido en Matlab (b) Retardo obtenido en osciloscopio

Figura 73: USRP N210 como Tx y USRP N210 como Rx. Modulación GMSK.

Tabla 21: Experimentos con dispositivos USRP no embebidos.

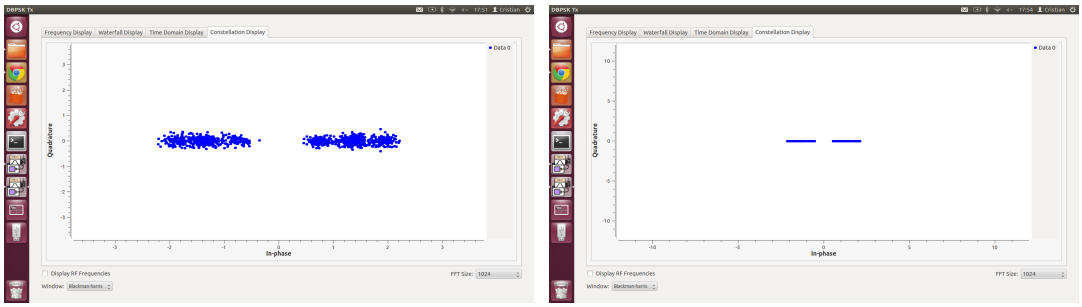
Transmisor	**Receptor**	**Modulación**	**Retardo (Muestras)**	**Retardo (ms)**
E110	E110	GMSK	330	41.025
E110	N210	GMSK	128	16
E110	N210	DBPSK	254	31.75
E110	N210	DQPSK	512	64
N210	N210	GMSK	8	1
N210	N210	DBPSK	22	2.75
N210	N210	DQPSK	47	5.875

Las Figuras C.1, C.1 y C.1 muestran las constelaciones obtenidas en la sección de Rx para los experimentos realizados.



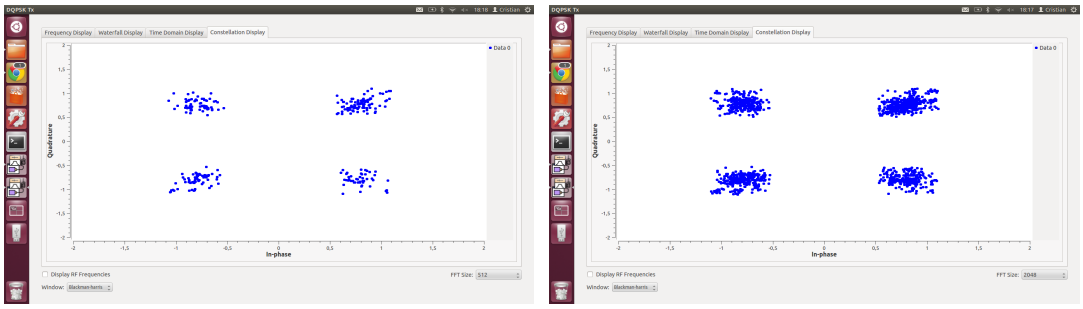
(a) USRP E110 Tx - USRP N210 Rx (b) USRP N210 Tx - USRP N210 Rx

Figura 74: Constelación GMSK obtenida en GNU Radio.



(a) USRP E110 Tx - USRP N210 Rx (b) USRP N210 Tx - USRP N210 Rx

Figura 75: Constelación DBPSK obtenida en GNU Radio.



(a) USRP E110 Tx - USRP N210 Rx (b) USRP N210 Tx - USRP N210 Rx

Figura 76: Constelación DQPSK obtenida en GNU Radio.

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

ACTA DE ENTREGA

El presente proyecto fue entregado en el Departamento de Eléctrica y Electrónica, y reposa en los archivos desde:

Sangolquí, _____ .

Elaborado por:

_____ _____

Marco David Torres Vega Cristian Santiago Vaca Gallardo

Autoridad:

_____

Ing. Paul Bernal Oñate

DIRECTOR DE LA CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES